DEPARTMENT OF PHYSICS, UNIVERSITY OF JYVÄSKYLÄ
RESEARCH REPORT No. 2/1989

0^+ STATES IN EVEN-EVEN LEAD ISOTOPES

BY
WŁADYSŁAW HENRYK TRZASKA

Academic Dissertation
for the Degree of
Doctor of Philosophy

YLIOPISTO · JYVÄSKYLÄN

AMICA VERITAS

Jyväskylä, Finland
October 1989

URN:ISBN:978-951-39-9890-5
ISBN 978-951-39-9890-5 (PDF)
ISSN 0075-465X

Jyväskylän yliopisto, 2023

ISBN 951-680-172-2
ISSN 0075-465X

DEPARTMENT OF PHYSICS, UNIVERSITY OF JYVÄSKYLÄ
RESEARCH REPORT No. 2/1989

0^+ STATES IN EVEN–EVEN LEAD ISOTOPES

BY
WŁADYSŁAW HENRYK TRZASKA

Academic Dissertation
for the Degree of
Doctor of Philosophy

To be presented, by permission of the
Faculty of Mathematics and Natural Sciences
of the University of Jyväskylä,
for public examination in Auditorium S-212 of the
University of Jyväskylä on October 14, 1989,
at 12 o'clock noon

Jyväskylä, Finland
October 1989

Preface

This work is a result of many years of research and experimental development done at the Department of Physics at the University of Jyväskylä (JYFL). On my part it covers the period of 1980 - 1984 and 1988 - 1989. I have spent the missing years 1984 - 1988 at Purdue University and at Argonne National Laboratory in the USA working towards implementation of electron spectroscopy to the heavy ion research. Even if not directly reflected in the thesis, experience gained during my stay in the States helped considerably, especially in the field of data analysis and in computer simulation of electron trajectories.

An experimental work can hardly ever be done singlehanded. This is especially true about this work. A complete list of all the people who helped is far too long to quote. Nevertheless, I would like to thank them all. My special gratitude goes to my colleagues from the 0-GROUP: to Prof. Juhani Kantele who was the guiding force of the wholc projcct and to Drs. Rauno Julin, Jaana Kumpulainen, Matti Luontama, Arto Passoja and Esa Verho who carried with me the heavy burden of experiments and data analysis.

I would also like to apologize to my wife Kristiina and to our little son Sebastian for spending too little time with them.

Wladyslaw Henryk Trzaska

Jyväskylä, June 1989

0^+ STATES IN EVEN-EVEN LEAD ISOTOPES

Abstract

A systematic investigation of excited 0^+ states in ^{202,204,206,208}Pb is carried out by means of in-beam conversion-electron and gamma-ray spectroscopy. Several new 0^+ states and E0 transitions are reported. The measured quantities include transition energies, half-lives, X- and ρ^2 values. The results are discussed in view of the published shell model calculations and intruder-states systematics. A comprehensive evaluation of the K/L ratios of E0 transitions is presented.

The experimental developments cover, among other things, high-energy electron spectroscopy, new, controversial results concerning detection efficiency of electrons by a semiconductor detector and a revised approach to data analysis of electron spectra.

A new compilation of several useful calibration sources for gamma-ray and electron spectroscopy, based partially on present measurements, is included.

CONTAINS

ERRATA

Page	line from		Is	Should be
	top	bottom		
4		6	Phys. Rev. C, in print	Phys. Rev. C, 40, 1520 (1989)
14	17		data included	data includes
19	Fig.9		BILL against REF. 38	BILL against REF. [DE79]
34	5		4Pi	4π
35		3	its value	ρ^2 value
40		1	205Tl(p,2n)204Pb	^{205}Tl(p,2n)^{204}Pb
59		1	To a good approximation, excluding..., the total E0...	To a good approximation the total E0...
61	5		[KA88,...	[KA88b,...

W.H.Trzaska *0^+ STATES IN EVEN-EVEN LEAD ISOTOPES*

1 Introduction

Positive parity, zero-spin states (0^+) represent an important part of the level scheme of an even-even nucleus. There is a large variety of excitation mechanisms leading to formation of a 0^+ state. To name only a few, there are pairing vibrations, two- and four-quasiparticle excitations, two-quadrupole- and two-octupole-phonon vibrations; each with different characteristics. Comparison of experimental and calculated strength parameters and excitation energies gives an important insight into the validity of nuclear models. Often, vital information concerning nuclear deformation or intruder states can be extracted from the decay properties of 0^+ states. Extensive discussion of various aspects of monopole excitations and E0 transitions is presented in several review articles [CH59, RE61, KA84, VO86, LA82, SP74].

Despite major advances in gamma-ray spectroscopy that resulted in an impressive growth of our knowledge even on levels at high energies and with high spin, many of the fundamentally interesting 0^+ states have remained undiscovered. One of the main reason for this is, of course, that the single-photon E0 transitions are strictly forbidden by angular-momentum conservation. A transition between two 0^+ states can only proceed through a conversion-electron emission (for transition energies exceeding 1022 keV also through internal pair formation) or through a two-photon transition. The latter is a higher-order process representing practically negligible fraction, of the order of 10^{-4}, of the total transition probability [SC84]. Also, being non-yrast states, 0^+ states are only weakly populated in most of the nuclear reactions.

To the proven ways of detecting 0^+ states belong: (i) transfer-reaction studies, and (ii) electron and internal-pair spectroscopy. Under favourable circumstances the 0^+ assignment can also be reached by other means, for example, from angular correlations of gamma-rays. The former method (transfer-reaction study) requires energy determination and angular distribution measurements of the light component of the reaction products. The data are then analyzed in terms of the angular momentum L transferred in the reaction by comparison with the DWBA calculations. From the knowledge of L, the ground-state spin of the target nucleus, and the spin of the transferred nucleon (or nucleus), one can deduce the spin of the excited state. Its energy is calculated from the energy of the light product. Despite poor energy accuracy (typically around 10-15 keV) and a certain degree of doubt originating from the fact that the spin determination is not direct but involves model calculations, nuclear transfer studies have proved to be successful in locating many of the 0^+ states. Nevertheless, only electron spectroscopy, especially when combined with traditional gamma-ray measurements, can yield vital spectroscopic characteristics, such as X-values, half-lives, and good energy determinations.

Equally important as the spectroscopic information on individual states is the systematics of these levels over a large number of nuclei. Only then there is a good chance to expose and finally understand the processes responsible for what we can measure in the first place.

The lead isotopes, including the heaviest doubly-magic nucleus ^{208}Pb, play a very important role in experimental and theoretical nuclear physics attracting considerable interest associated with the study of shape coexistence, the intruder state systematics, and the onset of deformation. The location and characteristics of the excited 0^+ states are among the crucial pieces of information needed in these studies but, until recently, data on that subject have been very scarce.

Prior to this work the only high-quality electron spectroscopy study [TA72, JU76] was the half-life and ρ^2 measurement of the 0^+_2 state in ^{206}Pb. Other data were predominantly obtained from transfer reactions [FL74, LA77, AN77, WE81, TA83]. In some cases [GO70], measurements with magnetic spectrometers were made but poor efficiency and poor resolution at electron energies above and around 1 MeV, which are characteristic of these devices, limited the credibility of the data.

To improve the situation, systematic in-beam studies of the even-even lead isotopes between mass numbers 202 and 208 have been carried out in the present work. At the same time, measurements by Van Duppen et al. [DU84, DU85] on lead isotopes below mass 202, performed using the on-line isotope separator LISOL [VE81], added a new dimension to our results. With the data on all even-even Pb isotopes between A=192 and A=208, a comprehensive base for intruder states systematics emerged [HE87], enabling us to make tentative assignments to the two-particle-two-whole proton intruder configuration.

Most of the results and experimental methods presented in this work have been published in the following papers:

1) J. Kantele, M. Luontama, W. Trzaska, R. Julin, A. Passoja, and K. Heyde, E0 TRANSITIONS IN ^{202,204}Pb AND INTRUDER-STATE SYSTEMATICS OF EVEN-EVEN LEAD ISOTOPES, Phys. Lett. B **171**, 151 (1986), https://doi.org/10.1016/0370-2693(86)91523-6

2) R. Julin, J. Kantele, J. Kumpulainen, M. Luontama, A. Passoja, W. Trzaska, E. Verho, J. Blomqvist, E0 study of 0^+ states near 5 MeV in ^{208}Pb, Phys. Rev. C **36**, 1129 (1987), https://doi.org/10.1103/PhysRevC.36.1129

3) R. Julin, M. Luontama, A. Passoja, W. Trzaska, DECAY OF 0^+_2 STATES IN EVEN-EVEN N=82 NUCLEI, International Symposium on IN-BEAM NUCLEAR SPECTROSCOPY, Debrecen, Hungary, May 1984,

4) S. W. Yates, L. G. Mann, E. A. Henry, D. J. Decman, R. A. Meyer, R. J. Estep, R. Julin, A. Passoja, J. Kantele, W. Trzaska, E0 decays of 0^+ states in ^{146}Gd: Search for two-phonon octupole excitations, Phys. Rev. C **36**, 2143 (1987), https://doi.org/10.1103/PhysRevC.36.2143

5) W. Trzaska, J. Äystö, J. Kantele, SEMICONDUCTOR TELESCOPE SPECTROMETER FOR BETA-RAY SPECTRA, Nucl. Instr. and Meth. **212**, 221 (1983),
https://doi.org/10.1016/0167-5087(83)90695-6

6) J. Kantele, M. Luontama, W. Trzaska, A. Passoja, SINGLE-DETECTOR PARTICLE IDENTIFICATION IN CONJUNCTION WITH COINCIDENCE MEASUREMENTS, Nucl. Instr. and Meth. **206**, 403 (1983),
https://doi.org/10.1016/0167-5087(83)90376-9

7) R. Julin, J. Kantele, J. Kumpulainen, M. Luontama, V. Nieminen, A. Passoja, W. Trzaska, E. Verho, A SETUP FOR SPECTROMETRY OF HIGH-ENERGY CONVERSION ELECTRONS, Nucl. Instr. and Meth. **A270**, 74 (1988).
https://doi.org/10.1016/0168-9002(88)90011-3

8) W. H. Trzaska, R. Julin, J. Kantele, and J. Kumpulainen, Solution of controversy over 1583-keV levels in ^{204}Pb, Phys. Rev. C, in print.
https://doi.org/10.1103/PhysRevC.40.1520

9) W. H. Trzaska et al., Experimental K/L ratios of E0 transitions, in preparation.
https://doi.org/10.1007/BF01290197

In preparation are also two instrumental papers: description of the new, modular spectrometer, and the results of our study of the efficiency of a Si(Li) detector together with the compilation of the calibration sources listed in appendix B.

2 Experimental

A study of a wide range of isotopes, produced in a number of different reactions, requires a flexible experimental set-up capable of handling diverse background conditions and a broad span of production yield. As the result of two decades of experimental commitment at our laboratory to development and to constant improvement of electron spectroscopy methods, the present set-up, although not ideal, fulfills many basic requirements needed in the course of such studies.

In addition to standard electron and gamma-ray spectroscopy, some special techniques had to be used, including high-energy electron spectroscopy which was developed primarily for the study of the high-energy E0 transitions in the even-even Pb isotopes. Also, a special care was taken in a precise efficiency determination over a wide range of electron energies.

2.1 Beam and target selection

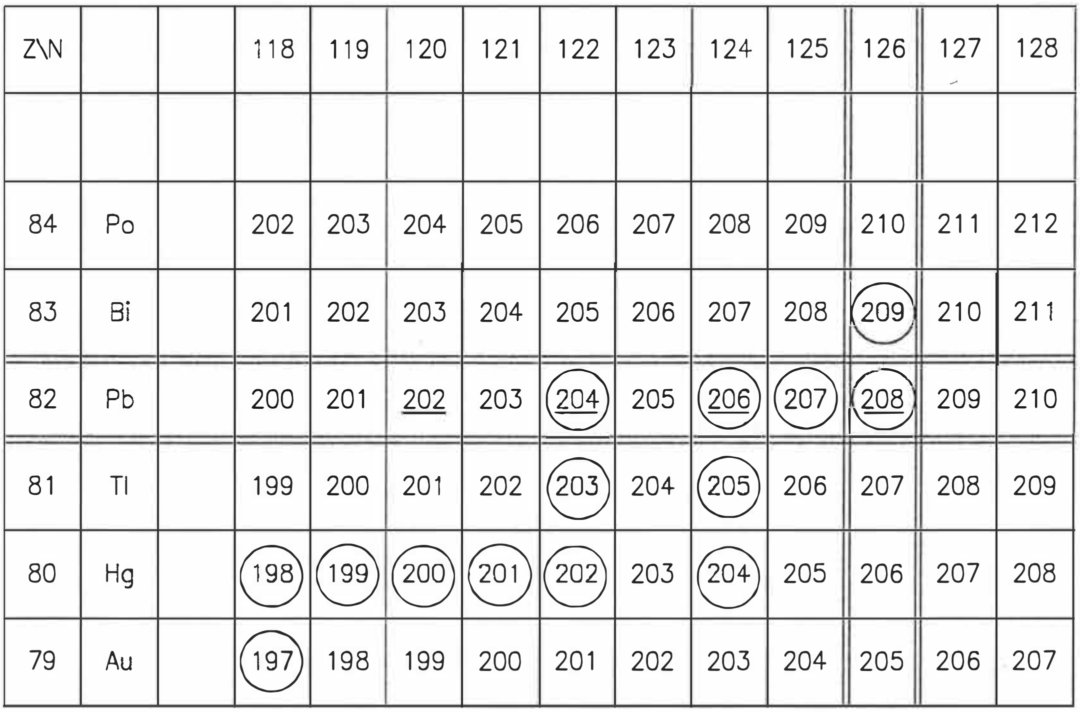
Figure 1 shows a fragment of the chart of nuclei in the lead region. It is obvious that the limited number of available target materials, together with the restrictions imposed by the JYFL cyclotron, leave little choice for the in-beam population of the excited states in the even-even lead isotopes. The stable ^{204,206,208}Pb nuclei can be exited, for example, through the inelastic scattering of protons, ^{202,204}Pb can be reached using the ^{203,205}Tl(p,2n) reaction, and ^{208}Pb - via the ^{207}Pb(d,p) reaction. All of these methods were employed.

Table 1 lists the properties of all the targets used in this study of the Pb isotopes.

Table 1. *Isotopic content (in per cent) of the targets used in this work. Cross bombardments with natural and enriched targets were an important part of the tests in identification of the new E0 transitions.*

Target	Isotope:	^{204}Pb	^{206}Pb	^{207}Pb	^{208}Pb
NAT.		1.4	24.1	22.1	52.4
204		66.5	16.1	7.5	9.9
206		-	90.4	6.7	2.9
207		0.01	2.66	89.1	8.2
208		0.1	0.7	1.4	97.8

Target	Isotope:	^{203}Tl	^{205}Tl
NAT.		29.5	70.5
203		86.3	13.8
205		3.6	96.4



Z\N			118	119	120	121	122	123	124	125	126	127	128
84	Po		202	203	204	205	206	207	208	209	210	211	212
83	Bi		201	202	203	204	205	206	207	208	209	210	211
82	Pb		200	201	202	203	204	205	206	207	208	209	210
81	Tl		199	200	201	202	203	204	205	206	207	208	209
80	Hg		198	199	200	201	202	203	204	205	206	207	208
79	Au		197	198	199	200	201	202	203	204	205	206	207

Fig. 1. *Fragment of the chart of nuclei in the lead region. Stable isotopes are marked with a circle. Isotopes investigated within this work are underlined. Double lines indicate magic numbers of protons (Z=82) and neutrons (N=126).*

2.2 Electron spectroscopy

Two generations of in-beam electron spectrometers [JU88, KA88a] were used to collect the electron data presented in this work. Both of them make use of a magnetic field to transport the electrons away from the target to a cooled semiconductor detector providing the energy information. The newer spectrometer, similar to those described in ref. [LI75, BA78, ST84, GU84], is a multimode solenoid-type unit designed to work in a number of configurations including a "traditional" lens mode (fig. 2) that differs little from the other spectrometer (fig. 3). Only the lens mode of operation was used for the purpose of this study.

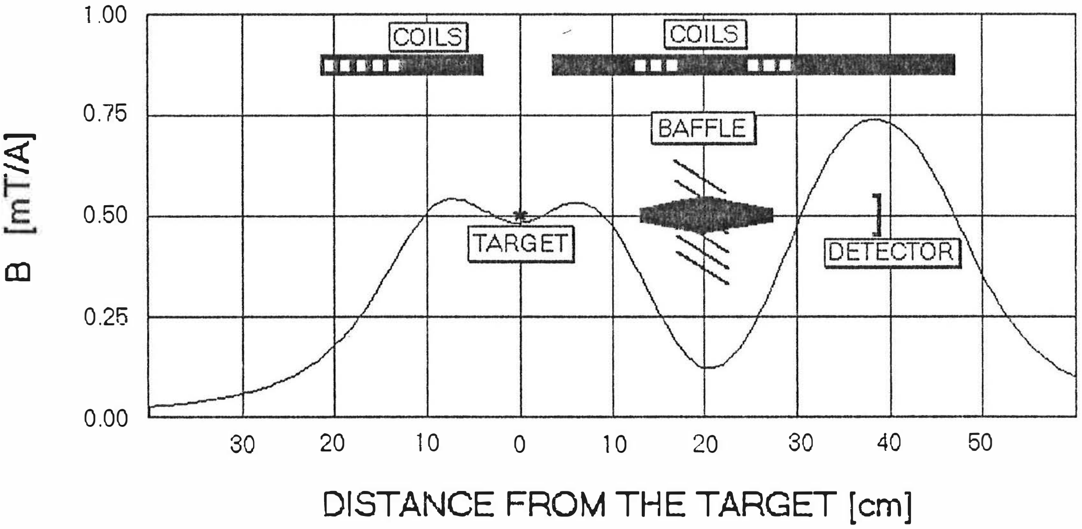


Fig. 2. Magnetic field profile of the new iron-free spectrometer operating in a lens mode. At the maximum current in the coils (700 A) the strongest magnetic field (at the detector position) is about 0.5 T. Inactive coils are marked by empty rectangles. In this configuration current in the coils surrounding the central baffle with the anti-positron "propeller" is reversed (-0.5I). Beam enters perpendicular to the plane of the drawing.

2.2.1 Lens mode of operation

Electrons in a magnetic field move along spiral trajectories with the drift defined by the electron momentum parallel to the magnetic field, and the radius of the spiral determined by the perpendicular component of the momentum. Due to the axial symmetry of the magnetic field in the spectrometers, it is convenient to describe the motion of the electrons using cylindrical coordinates. It is then easy to see that an electron emitted from the target (located at the origin of the z-axis) returns periodically to the z-axis at locations z1, z2 etc. that are the focal points for this particular trajectory. It is also easy to see that the adjoining trajectories, corresponding to small changes in electron momentum, will have focal points located close to z1, but already less close to z2 etc. (For that reason, in the lens mode one is usually interested in the first focal point.) Theoretically, each point along the symmetry axis can be chosen as a location for a detector since each point can be turned into a focal point (with some limited focussing characteristics) by adjusting the strength of the magnetic field. As a general rule, the closer the source, the better the (focal) properties and the transmission; but also, stronger magnetic field is required to focus the electrons.

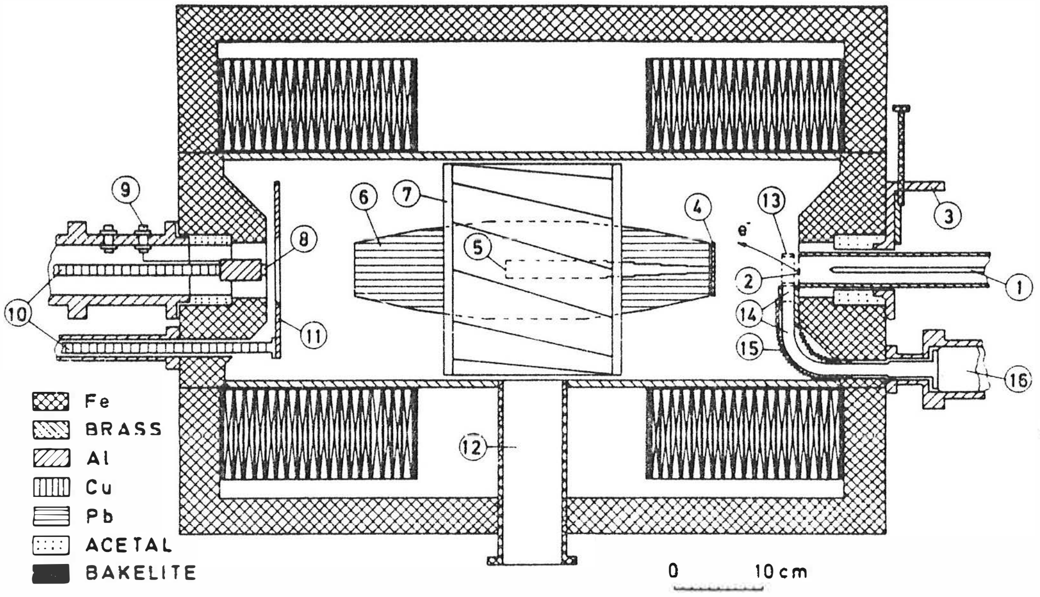


Fig. 3. *Magnetic lens + semiconductor detector conversion-electron spectrometer: (1) beam, (2) target, (3) target changing system, (4) collimator and current measurement, (5) Faraday cup, (6) lead shield, (7) anti-positron baffle, (8) detector, (9) feedthrough, (10) coldfingers, (11) coldtrap, (12) to pump, (13) cylindrical plastic scintillator, (14) light guide, (15) bellows, (16) PM tube. A Siegbahn-Slätis type magnet transports electrons from the target to the semiconductor detector (used as energy dispersion unit). Plastic scintillator ring was designed for coincidence measurements with charged particles.*

Even a uniform magnetic field has good focusing properties. They can be improved somewhat by keeping the source and the detector in a strong magnetic field and by lowering the magnetic field in-between (fig. 2). Such a configuration is referred to as an intermediate-image lens because of the additional ring focus formed by the trajectories half-way between the source and the detector, away from the z-axis. This feature of the intermediate-image lens makes it possible to construct an effective anti-positron baffle that blocks all the direct positron trajectories (some scattering is unavoidable) without significant losses in electron transmission.

The typical transmission of a magnetic lens is 8-20% resulting in a 4-10% detection efficiency, since roughly half of the electrons are back-scattered, and the momentum window $\Delta p/p$ is 15-25%. A good theoretical description of an electron spectrometer based on a magnetic solenoid is given in ref. [WE84] and also in ref. [JE49, PE49]. Calculations, some useful formulae, and certain experimental aspects concerning lens spectrometers are discussed in ref. [SI65].

2.2.2 Choice of detector

The size and the material of the detector have a crucial influence on the performance of the spectrometer by determining the final efficiency and the sensitivity to the background radiations. The detector thickness is an important parameter since it sets the useful energy range of the spectrometer. Readily available Si(Li) detectors have thicknesses between 0.5mm (corresponding to the range of 400keV electrons) and 5 mm (the range of 2.5 MeV electrons). In general, thinner detectors and those of smaller sensitive area have slightly better energy resolution. However, this is practically of no concern for in-beam studies where other effects such as the target thickness and the count rate determine the resolution. A good 5 mm by 200 mm^2 Si(Li) detector gives 1.9 keV FWHM at 1 MeV so there is seldom any reason to use thinner or smaller detectors on grounds of resolution alone.

For high-energy electron spectroscopy, one has to use Ge detectors. They are not as easily available as Si(Li) detectors and, except for the effective thickness, they are less suitable for electron measurements. Some of the main disadvantages are: high sensitivity to gamma radiation, larger efficiency loss due to back-scattering (both because of higher Z), more stringent requirements for cooling and vacuum. The only kind of germanium detector that we could acquire was a 5 mm by 80 mm^2 HPGe, where 5 mm of Ge approximates the range of 5 MeV electrons [MA68].

Contrary to the intuitive belief, the sensitive area of the detector located on the symmetry axis inside a magnetic lens or a solenoid does not play a major role in the overall detection efficiency. As a rule of thumb, the efficiency increases linearly with the radius - not with the area. Only the laboratory background increases with the area since it is proportional to the volume of the detector. For most applications a 100 to 300 mm^2 area gives the optimum performance. Smaller sizes are not advisable because of the relative importance of the detector frame obstructing electron trajectories.

2.2.3 Efficiency

The overall detection efficiency of a combination spectrometer (magnet + detector) depends on: (i) the transmission of the magnetic field which is a function of the magnetic field, the geometry of the system, and the electron momentum - p, and (ii) the intrinsic efficiency of the detector.

The transmission can be reliably calculated or measured. Measurements are very straight-forward if the absolute value of the transmission is not important. Normally, one only needs the relative properties of the transmission: the momentum window width - $\Delta p/p$, low and high momentum cut-off, etc. One measures the number of counts in a single peak - preferably (on grounds of

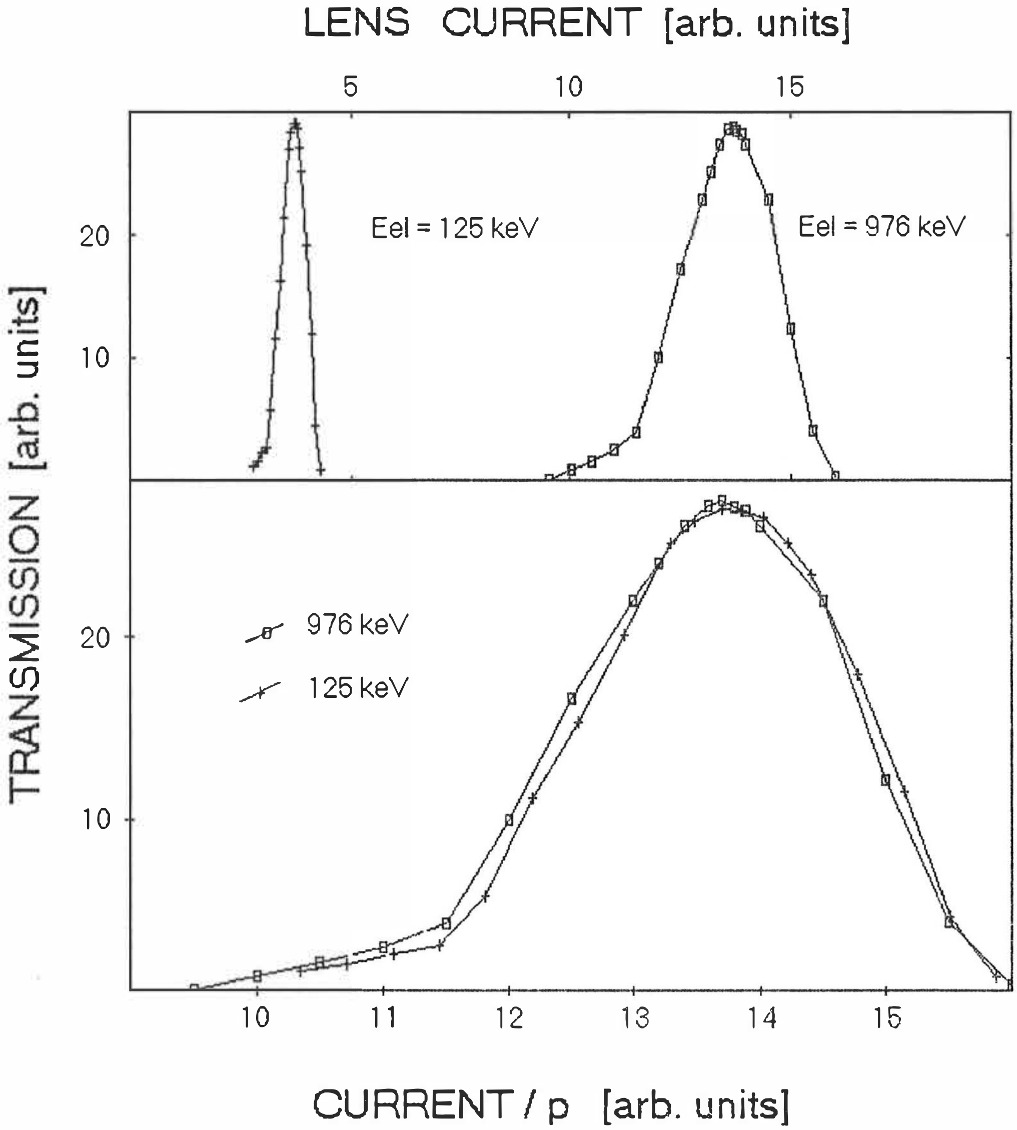


Fig. 4. ***Top:*** *transmission of a 125 keV and a 976 keV conversion-electron line as a function of the current in the lens (magnetic field).* ***Bottom:*** *the same curves plotted using energy independent units - current over momentum (~B/p). B/p is proportional to the reverse of the radius of an electron in a magnetic field - a geometric factor characterizing spectrometer's performance. Therefore 1/r (or B/p, or current/p) is a very convenient way of expressing the momentum window of a spectrometer.*

high intensity and low background) the 975.6 keV line from ^{207}Bi source - as a function of the magnetic field or, any other parameter proportional to it - usually the current in the coils. These measurements, carried out at one electron energy (momentum) and different values of the magnetic field, are equivalent to the measurements of the transmission for different momenta at a fixed magnetic field. They contain all the information needed to calculate the transmission at any field/energy value (fig. 4). Some useful formulae needed in these calculations are given in appendix A.

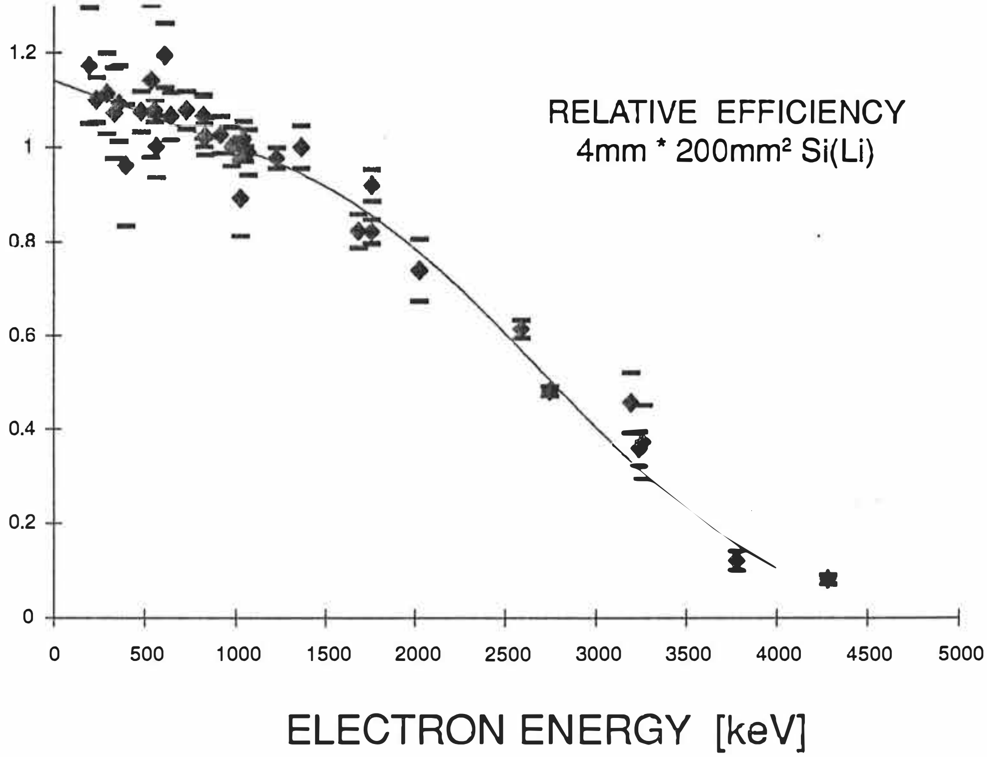


Fig. 5. *Efficiency of a 4mm by 200mm^2 Si(Li) detector operating in conjunction with a magnetic lens. The continuous line represents a fit to the data. The fitted function is a product of a linear function and a diffused step function. Five radioactive sources were used to collect the data: ^{152}Eu, ^{207}Bi, ^{133}Ba, ^{66}Ga, and ^{56}Co. To improve the clarity of the drawing, data points collected with ^{133}Ba source are not shown.*

The detector efficiency depends not only on the physical properties of the detector itself like its size and the material, but also on the conditions in which it operates. This is especially true for the magnetic field that not only affects the electrons that are to be detected but also the whole detection process (charge collection in the detector). Indeed, some detectors (mainly gamma-ray detectors) do not operate at all in a magnetic field above about 0.1 Tesla. Our experience with Si(Li) detectors shows that as long as the bias does not exceed 1000 V they can take a magnetic field as high as 2 Tesla without break-down, although some deterioration of the line shape has been observed.

The present detailed experimental study has revealed some interesting characteristics about the detector response to the electrons. A number of calibration sources were used, extending in energy up to 4800 keV. A thorough compilation of the existing experimental data on these sources was done for the purpose of this work. When possible, electron intensities were also calculated using the gamma-ray intensities and mixing ratios. The results of the compilation and calculations for the conversion-electron calibration sources are presented in appendix B. The compilation of ^{152}Eu data included our own measurements as well.

Figure 5 shows the efficiency of a 4 mm by 200 mm^2 Si(Li) detector operating inside the magnetic lens. As expected, there is a clear drop in the efficiency around the electron energy corresponding to their range in silicon. What is surprising, though, is the relatively strong energy dependence for the low energy electrons - 14% difference between 0 and 1 MeV. All the calculations [BE69] as well as the measurements [JA74] that we know of indicate constant efficiency in that energy range for a detector like the one used here. In figure 6 our result is compared with calculated results for a 3 mm and a 5 mm silicon detector from ref. [BE69].

In order to understand the large and unexpected energy dependence of the efficiency curve below 1.8 MeV, a number of additional experiments were

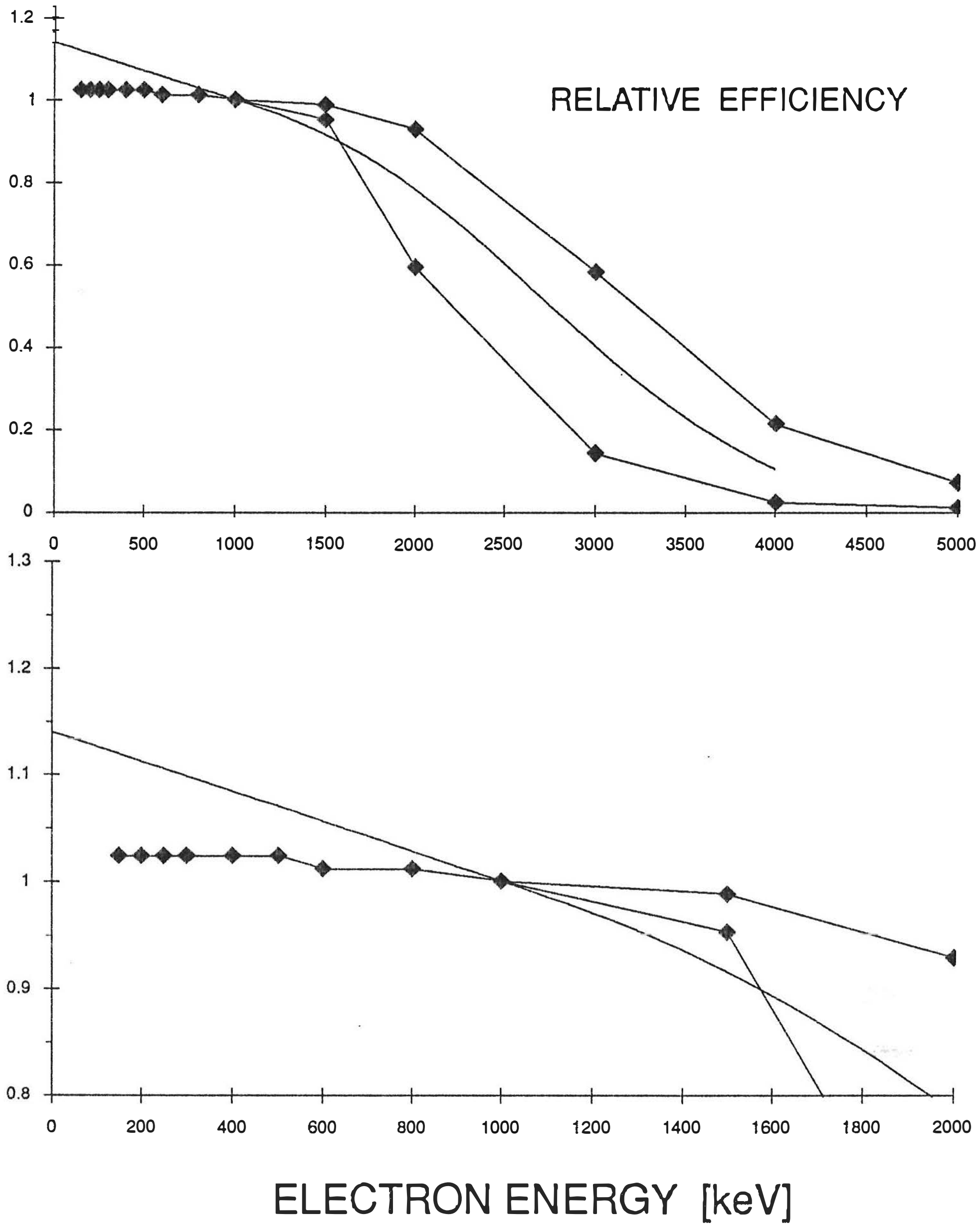


Fig. 6. *Discrepancy between measured efficiency of a 4mm thick Si(Li) detector (smooth line, cf. fig. 5) and calculated efficiency (connected rectangles) for a 3mm and a 5mm silicon detector. Although at high electron energies (top) the experimental values fall, as expected, half-way between the theoretical curves, there is considerable difference for electron energies below 1800 keV (bottom).*

carried out. The following effects were investigated:

1. Nonlinearity of the sweep. The rate of change of the magnetic field is very steady (fig. 7). The contribution of the instability of the sweep in the 0-1 MeV region was estimated to be (1.2 +- 0.5)%.

2. The edge effects. Electrons focused by a magnetic lens do not necessarily fall on the detector surface evenly; most of them may hit the active surface close to the edge, which in turn would increase the probability of electron's escape before its full energy is absorbed. Indeed, most of the spectrometers produce a ring focus [KL65] rather than a point focus but this effect is completely wiped out in the swept-mode of operation. In addition, tests with collimated and uncollimated electron sources showed that, in case of our detector, the edge effects do not play any noticeable role (0 +- 1)% for energies below 1 MeV.

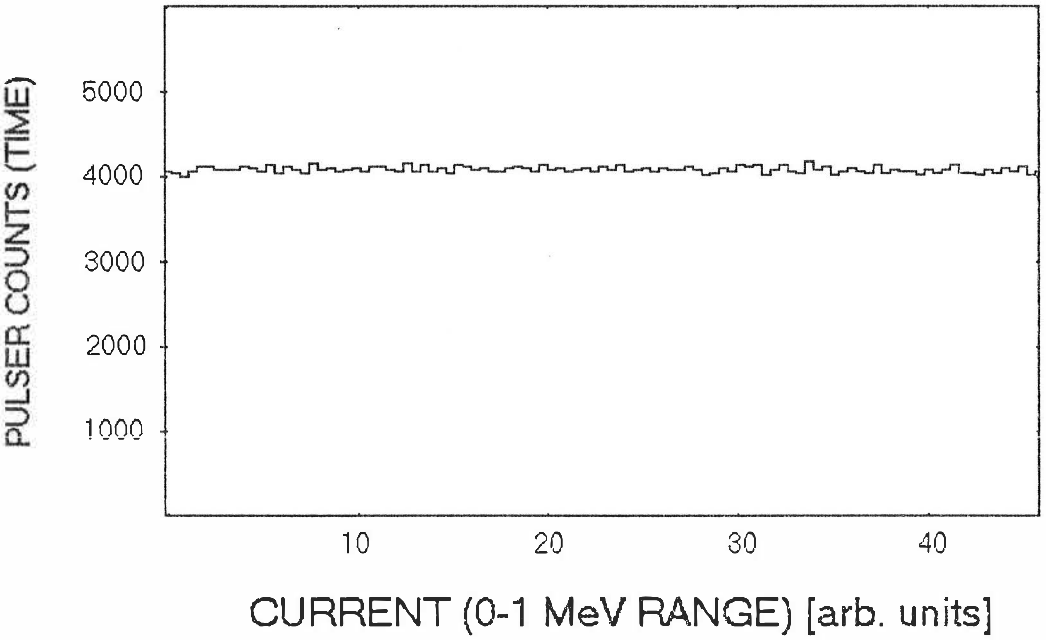


Fig. 7. Sweep stability in the low energy range. As part of the detailed efficiency determination, sweep control of the current in the magnetic lens was closely monitored and found reliably constant throughout the whole operating range.

3. Entrance angle. There is a very strong dependence between backscattering and the angle at which electrons fall on the detector's surface [TA71, KA82a]. Fortunately, this dependence is very nearly the same for all the energies and therefore does not affect the relative intensity. The present measurements indicate differences of no more than 1%.

4. Magnetic field. Obviously, operating in the lens mode, each electron energy is recorded at a different value of the magnetic field. If the detector performance is influenced by the magnetic field, then it would naturally lead to the drop in the efficiency. The experiments (fig.8) suggest a drop in efficiency

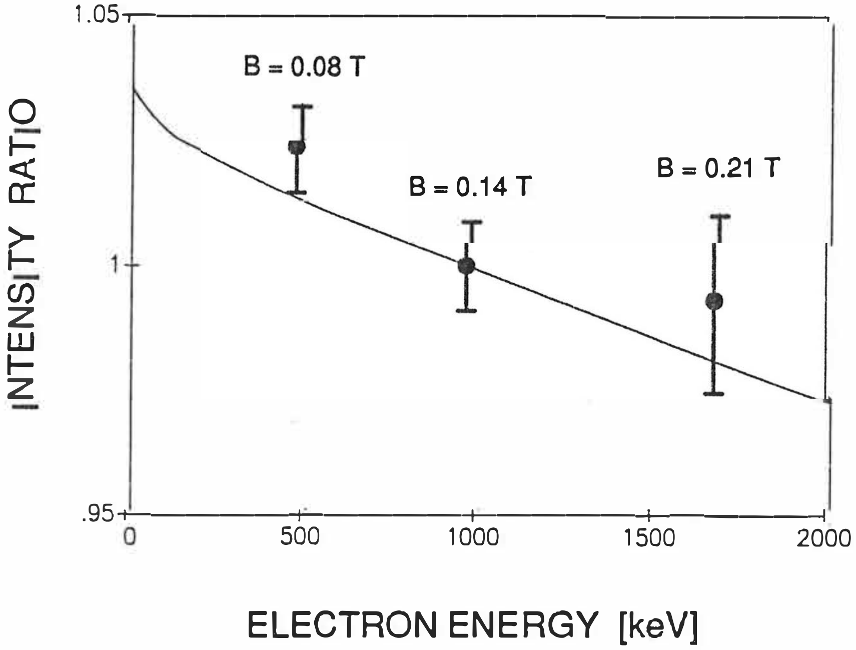


Fig. 8. *Magnetic field effect on detection efficiency of conversion-electron lines by a Si(Li) detector. Line intensities from two ^{207}Bi decay spectra were compared: with the magnet operating in a lens mode, and without the magnetic transporter. A small effect, about 3% per 0.1T, was noticed in agreement with observations made using a 2T superconducting solenoid at Argonne National Laboratory (continuous line). However, this effect alone can not account for the large efficiency discrepancy shown in fig. 6.*

of about 3% per 0.1 Tesla (less than 4.5% in the 0 - 1 MeV range). This value is consistent with the author's measurements with a superconducting solenoid spectrometer, capable of producing a uniform magnetic field up to 2 T, where a 2.5% efficiency drop per 0.1 T was noticed. It was accompanied by a deterioration of the resolution (FWHM) similar in magnitude to the efficiency drop. At present, there are no efficiency calculations for a Si(Li) detector in the magnetic field and no other experimental reports to confirm this effect. Also, our preliminary tests with gamma-ray sources do not show a similar dependence of the photopeak area on the magnetic field.

Whatever the explanation might be, the efficiency determination of our experimental set-up seems very reliable. An excellent agreement between the conversion-electron intensities from the decay of ^{152}Eu remeasured by us and the original high quality data [CO85] from the double-focusing iron-core BILL spectrometer [MA78] proves it well (fig. 9b).

It is interesting to point out that, until recently, the experimental values of conversion electron intensities from the decay of ^{152}Eu had a significant systematic error. Figure 9a compares the latest ^{152}Eu data [CO85], obtained with the BILL spectrometer to the previously published data [DE79], taken with a Si(Li) detector. There is a clear deviation for the points below 500 keV. If one would assume that the Si(Li) efficiency changes in a similar fashion as observed in the present work, instead of being constant as it was assumed in the previous ^{152}Eu publication [DE79], these differences could be accounted for.

2.2.4 Shape-analysis of electron peaks

There are certain difficulties in analyzing conversion-electron data recorded with a Si(Li) or a Ge detector; especially when the source thickness can not be neglected. Unlike the gamma-ray peaks, the electron peaks have a considerable exponential tail and a significantly higher background on the low

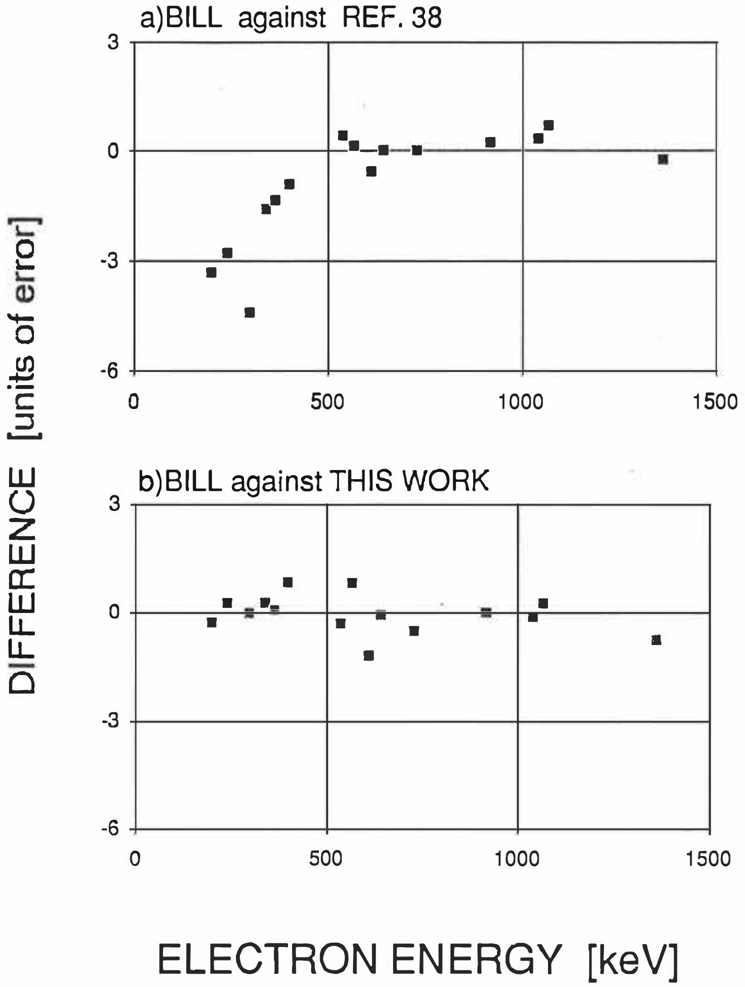


Fig. 9. *Crucial test to verify the new, controversial results concerning Si(Li) detector efficiency. For comparison, the latest conversion-electron intensity data form the decay of ^{152}Eu obtained with the BILL spectrometer, without utilizing semiconductor detectors as energy dispersion units, were used. A): intensity difference between the BILL data [CO85] and a previous study [DE79], divided by the sum of experimental errors from both works and plotted as a function of electron energy. Using such a representation, all points should fall between +1 and -1 on the y-scale. However, there is a clear, systematic deviation for electron energies below 500 keV. In the previous study a constant efficiency in that range was assumed. B): the same kind of a plot showing intensity difference between the BILL data and the present results assuming, as measured, a sloped efficiency curve even at low electron energies. Good agreement and lack of any systematic deviations prove the correctness of our experimental method. The complete set of experimental values used in preparation of this figure is given in table 2.*

Table 2. *Measured relative intensities of conversion electron lines from the decay of ^{152}Eu. To make the comparison easier, intensities of L1, L2 and L3-lines, resolved in the BILL spectrometer were added-up. The data from ref. [DE79] were normalized by a factor of 1.63 . Figure 9 shows a graphic representation of differences between the measured intensities.*

El. Energy	BILL	Si(Li)	this work
[keV]	ref. [CO85]	ref. [DE79] • 1.63	
197.9	38.9 +- 1.3	52 +- 2.63	40.61 +- 4.81
237.4	10.51 +- 0.22	12.97 +- 0.66	10.34 +- 0.44
294.0	55.1 +- 1.9	65.82 +- 0.52	55.26 +- 4.33
336.2	12.74 +- 0.28	14.35 +- 0.72	12.36 +- 1.15
360.9	2.95 +- 0.10	3.36 +- 0.20	2.93 +- 0.21
397.1	1.30 +- 0.05	1.41 +- 0.07	1.14 +- 0.14
536.1	0.69 +- 0.03	0.65 +- 0.07	0.73 +- 0.11
565.2	0.60 +- 0.02	0.59 +- 0.06	0.56 +- 0.03
609.7	0.50 +- 0.02	0.54 +- 0.05	0.56 +- 0.03
641.8	2.17 +- 0.07	2.17 +- 0.13	2.18 +- 0.08
728.7	1.46 +- 0.03	1.46 +- 0.07	1.50 +- 0.03
917.2	2.49 +- 0.09	2.44 +- 0.13	2.49 +- 0.04
1039.1	1.60 +- 0.06	1.55 +- 0.09	1.61 +- 0.02
1065.3	1.76 +- 0.06	1.65 +- 0.10	1.73 +- 0.06
1361.2	0.73 +- 0.03	0.75 +- 0.05	0.76 +- 0.01

energy side of the peak. As a result, the uncertainties in peak areas arise from the uncertainties of the analysis, as well as from the statistical errors.

When very precise measurements were called for (for instance establishing the efficiency of a detector) three different techniques were used to derive the peak areas from each spectrum: (i) integration without any shape-fitting, (ii) shape-fitting, and (iii) area-fitting using a fixed set of shape parameters. The average was then calculated and the final error was extracted from the differ-

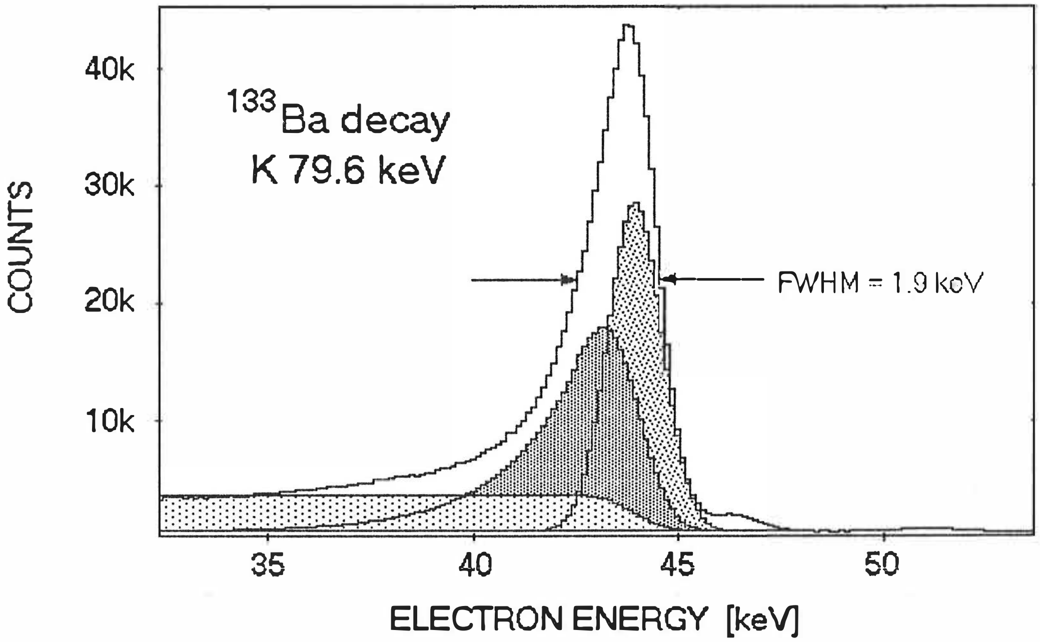


Fig. 10. *Line-shape of a conversion-electron peak. Unlike in the case of gamma-ray spectra, the Gaussian component does not necessarily dominate. In this example, most of the counts belong to the distorted Gaussian component. There is also a considerable step-like background. Majority of the available peak fitting codes can not account for such a shape. Therefore, a special computer program was created to ensure correct treatment of the data. Even if, due to a very low electron energy, this example is somewhat extreme, many of the high-energy peaks recorded using thicker targets look very similar.*

ences produced by each technique. The statistical error was used whenever it exceeded fitting differences (usually, in the case of weaker lines). This procedure gave a good protection against systematic errors in the interpretation of the data.

For the shape analysis, a special computer program was used (see appendix C) dividing each peak into a Gaussian component, a distorted Gaussian component, a step function, and background (fig. 10). The procedure was similar to that described in ref. [JO77]. The main advantages of this method are: relative simplicity (only three components plus background; all linearly dependent), good agreement with the actual line-shape (to about a few percent), and universality (the same analysis can be applied to the gamma-ray spectra although the interpretation of the non-Gaussian elements will be different). The final peak area was calculated by adding the areas of the Gaussian and the distorted Gaussian component.

2.2.5 Energy calibration of the in-beam spectra

The most accurate way of energy determination of the in-beam spectra is the use of known transitions for in-beam (internal) calibration. The in-beam calibration is usually different from the off-beam calibration due to the energy loss of the electrons in the target. This energy loss is energy dependent. Typically, a 1.5 MeV electron loses 4 keV (mean energy loss) when passing through a 4 mg/cm^2 Pb foil. The energy loss in low-Z materials is even more severe. When the in-beam calibration can not be used, one can quite accurately simulate it by measuring the electron energy from a bare calibration source and then from the same source with the actual target put in front of it. Since during the run the electrons will be emitted from within the target volume, the average energy shift will be one-half of the measured value. This assumption is, of course, not valid if the compound nucleus can recoil out from the target.

Some useful formulas to calculate the energy shift and the line broadening after the passing of an electron through an absorber are given in appendix A. These formulas are found to be quite reliable.

2.3 High-energy electron spectroscopy

It is a well established fact that in ^{208}Pb three excited 0^+ states are expected to lie in the 4 to 6 MeV region, as discussed in chapter 3.4. A similar situation (excited 0^+ states at 4 to 6 MeV) is expected in the neighboring ^{206}Pb (chapter 3.3). Interest in these states gave the motivation to develop a set-up for in-beam, high-energy conversion electron spectroscopy. This was done by modifying the designs of the previous combination-type spectrometers. At the first stage, a powerful Siegbahn-Slätis intermediate-image spectrometer [SL49], capable of focusing electrons up to 8 MeV was adopted and fitted with a thick Si(Li) or HPGe detector [JU88] (fig. 3). At the second stage, a new iron-free magnet was constructed to replace the previous one [KA88a].

If the transition energy exceeds 1022 keV (twice the electron rest mass) there is an additional mode of decay between the states involved - the internal pair formation (IPF). The IPF/IC ratio sharply increases with the transition energy. At 6 MeV for Z=80 the IPF/K-conversion ratio is about 4. In principle, IPF measurements offer an alternative to conversion-electron spectroscopy. However, since the former requires coincidences of $e^+ e^-$ pairs, the detection efficiency is very low.

2.3.1 Energy and efficiency calibration

A major difficulty in the determination of the spectrometer performance at high electron energies was the lack of readily available and reliable calibration sources. Since there are no good-quality experimental data (usually no data at all) on the conversion-electron intensities at high energies, we had to rely on the gamma-ray intensities and multipolarities, and the calculated values of the ICC. The 79 d ^{56}Co decay has relatively well established gamma-ray energies and intensities but they extend only up to 3.5 MeV. Since the 9.4 h ^{66}Ga decay has energies up to 4.8 MeV and, due to a shorter half-life, is easier to produce, it became the primary choice for energy and efficiency calibration. Also, whenever possible, internal calibration was used which considerably improved the energy determination. The ^{66}Ga activity was produced in proton bombardments on enriched ^{66}Zn targets. The highest yield in this reaction is reached at E_p=12 MeV which, for a 5 mg/cm^2 target, corresponds to the proton beam of 16 MeV [SZPC].

It can be reminded at this point that, unfortunately, the current publication [CA71] of gamma-ray intensities from the decay of ^{66}Ga has, most probably, a systematic error [GR75]. The error arises from an invalid assumption regarding the Ge(Li) detector characteristics which leads to steadily increasing errors above 2.5 MeV. The intensities given in appendix B are corrected for this error as suggested in ref. [GR75] (original data are also included).

2.3.2 Background reduction

The high background is the major obstacle for all in-beam experiments. As we have found out, neutrons produced in the beam dump and in the collimators give rise to most of the background in our high-energy electron measurements. Thermalized neutrons interact with materials surrounding the detector (coils, cold fingers, baffle etc.) via the (n,gamma) reaction. The gamma rays in turn yield Compton electrons that can spiral down to the detector. Some of these gamma rays are also registered directly in the detector - typically as escape peaks.

In our case, the background reduction went in two directions: (i) to decrease the number of neutrons (ii) to minimize their effect. For proton energies around 18-20 MeV (the maximum energy of the JYFL cyclotron is 20 MeV) a dramatic background reduction was achieved by changing the beam-dump material from lead to carbon (graphite). As it is clearly demonstrated in figure 11, for 3 MeV electrons the background went down by a factor of 10 and for higher energies even more. The design of the new spectrometer permits

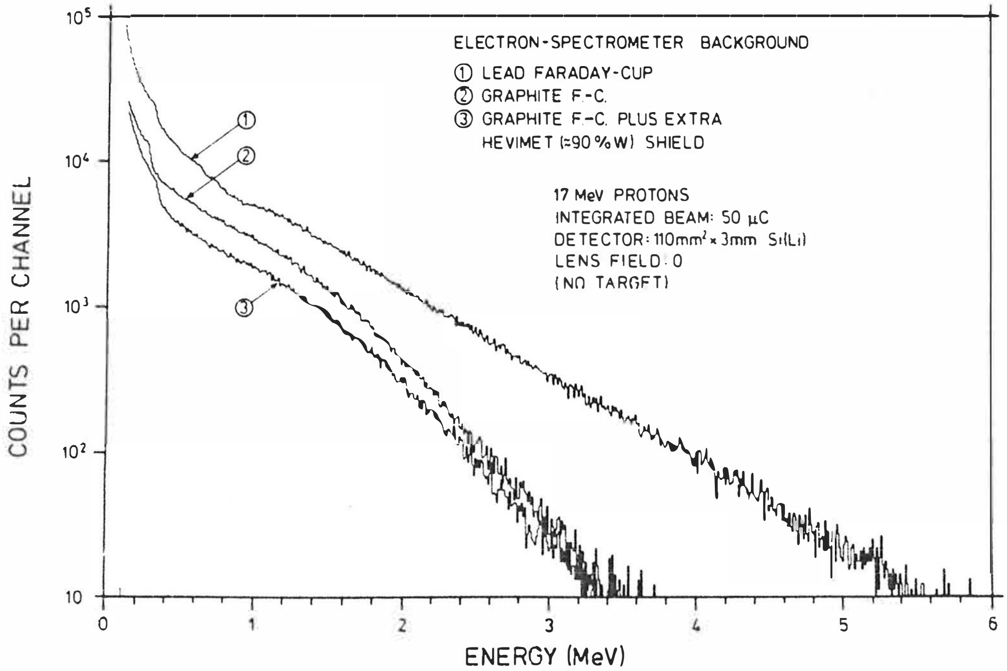


Fig. 11. Background spectra obtained using different beam-dump materials. Clearly, over an order of magnitude improvement was possible at electron energies above 3 MeV. Without this improvement most of our high-energy conversion electron studies would be impossible. The design of the new spectrometer permits dumping of the beam deep in the wall. Preliminary tests indicate that, if implemented, such a change would further reduce the background by a factor of 5.

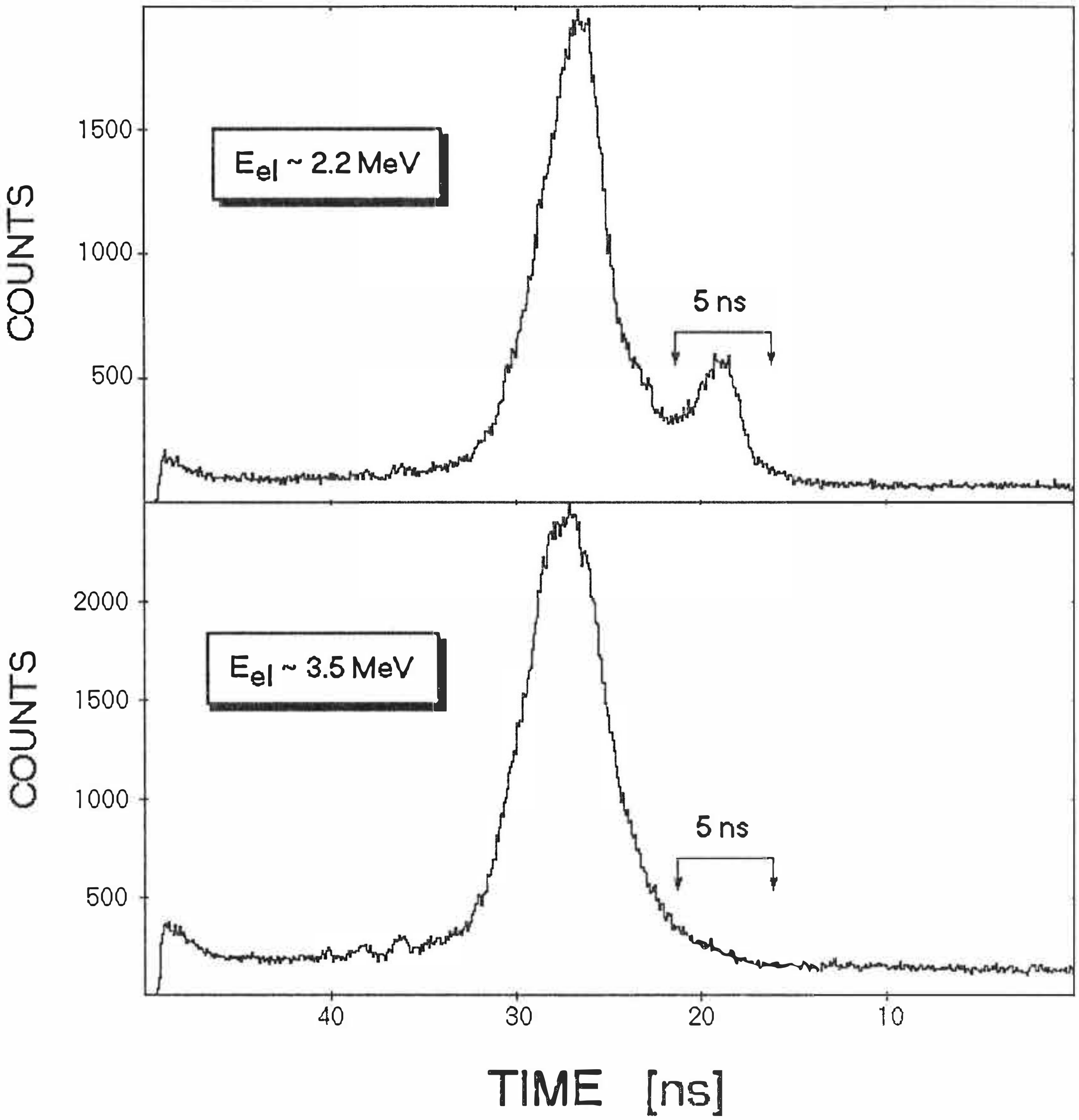


Fig.12. *Time spectra from a $^{206}Pb(p,p')$ experiment at a proton energy of 17.3 MeV. The time difference between the electron detector pulse and the RF signal of the cyclotron was measured. In both cases the transmission window of the magnet was fixed: at around 2.2 MeV (top) and 3.5 MeV (bottom). The main peaks in the spectra are related to prompt background events. With the transmission window set at the region of interest (bottom) there are so few prompt electrons coming from the target that they do not form any visible contribution to the time spectrum. Only at a lower setting of the transmission window the "true" prompt peak can be seen. A 5 ns time gate used to collect the upper spectrum in fig. 13 is also shown.*

the location of the beam dump deep in the wall - far from the detector - which will give further reduction in background. First tests indicate improvement by a factor of 5.

Many of the neutron-induced background events are slower then prompt conversion electrons produced in the target and, therefore, a narrow, prompt time gate often cuts down these undesirable events. Energy selection correlated with the momentum window in the swept mode of operation serves the same purpose [KA75].

There are certain difficulties in placing a narrow prompt time gate on a high-energy electron spectrum. There are very few prompt electrons coming from the target (as compared to the prompt background). To locate prompt electrons in the time spectrum, special techniques had to be used including tests with other targets. For example, any oxide target would produce plenty of prompt electrons with energies around 2.5 MeV. Figure 12 shows a time spectrum from an ^{206}Pb(p,p'e) experiment. Marked is the position of a 5 ns time gate used to collect the spectrum shown in figure 13 (top). For comparison, a gross (ungated) spectrum is also shown (fig. 13 - bottom).

Since electrons from oxygen, arising from intense pair production from the deexcitation of the 6 MeV 0^+ state leading to a continuous spectrum of e^+ and e^- with the end-point of 5 MeV, have the same time properties as the main events, oxygen impurities in the target are the main source of background in the spectra. Regretfully, we are not able to control target oxidization well enough.

2.4 Half-life measurements

Half-life measurements are, in general, difficult and can be done only under favorable circumstances. This is especially true in the present cases, when the 0^+ half-lives are expected in the sub-nanosecond region. A crude approxi-

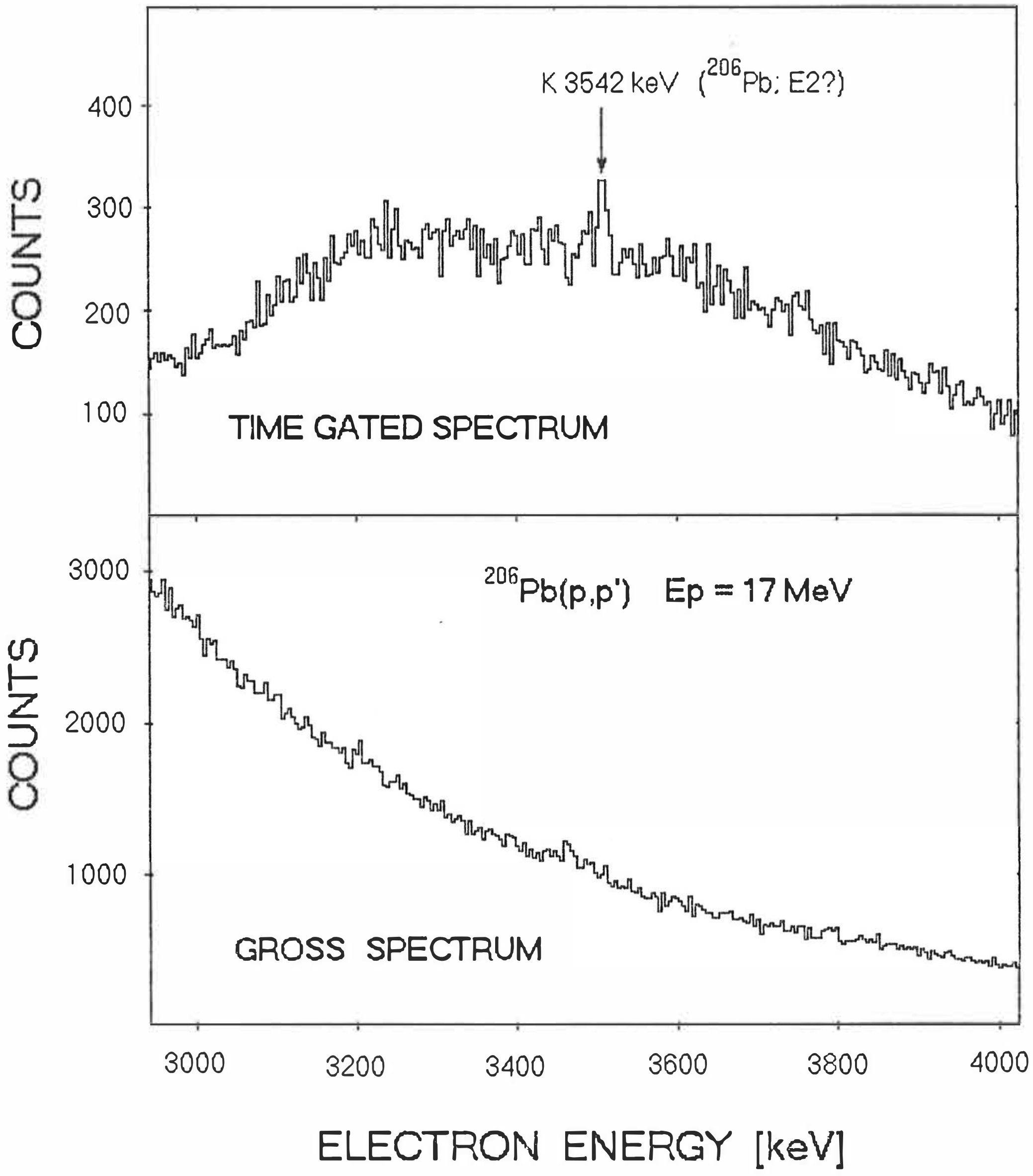


Fig. 13. Improvement in spectrum quality using a narrow time gate. Both spectra were acquired for 8 hours with the transmission window fixed at 3.5 MeV. Proton energy was 17.3 MeV at 25 nA and the target was a 3.5 mg/cm^2 ^{206}Pb. The gross spectrum (bottom) does not show any structure. Only the gated spectrum (top, see also fig. 12.) brings back the bell-shape characteristic of fixed-mode measurements. The single line in the middle is the K-component of a weak, unidentified 3542 keV (+- 2 keV) transition in ^{206}Pb. The conversion coefficient was estimated at 0.001 (+- 50%) allowing for an E2, E3, E4, M1 or M2 transition.

mation can be extracted from the tabulated [KA88b] Ω values (electronic factors). If we assume that a 0^+ state decays mostly by an E0 transition with the probability described as [CH59]

$$W(E0) = \rho^2 \sum_j \Omega_j(Z,k),$$

where ρ is the monopole strength parameter, k is the transition energy expressed in units of electron rest mass (511 keV), and the summation index refers to the atomic orbitals K,L,M,..., and to the internal pair formation - then the partial half-life for K-conversion decay of the 0^+ state is related to these numbers in the following manner:

$$t_{1/2} = \ln 2 \;/\; \rho^2 \, \Omega_K(Z,k).$$

Typical values of ρ^2 lie between 10 and 100 milli-units (10^{-3}), Ω_K for Z=82 and energy 1-3 MeV is of the order of 10^{12} so, indeed, one expects half-lives below 100 ps. These values are out of reach of conventional slope methods but could be measured using the centroid-shift technique [KA82b].

2.4.1 Centroid-shift method

The time difference between a reference signal related to the beam hitting the target and the complete energy absorption of a conversion electron in the detector is determined by a number of factors. One of them is, of course, the half-life. However, unlike in cases when the slope method can be used, the half-life makes only a small contribution to the final line shape of the time spectrum. This contribution can be detected as a small shift from the expected value of the centroid position of the time peak. Obviously, the main difficulty lies in correct interpretation of the measured shift. As a rule, the centroid

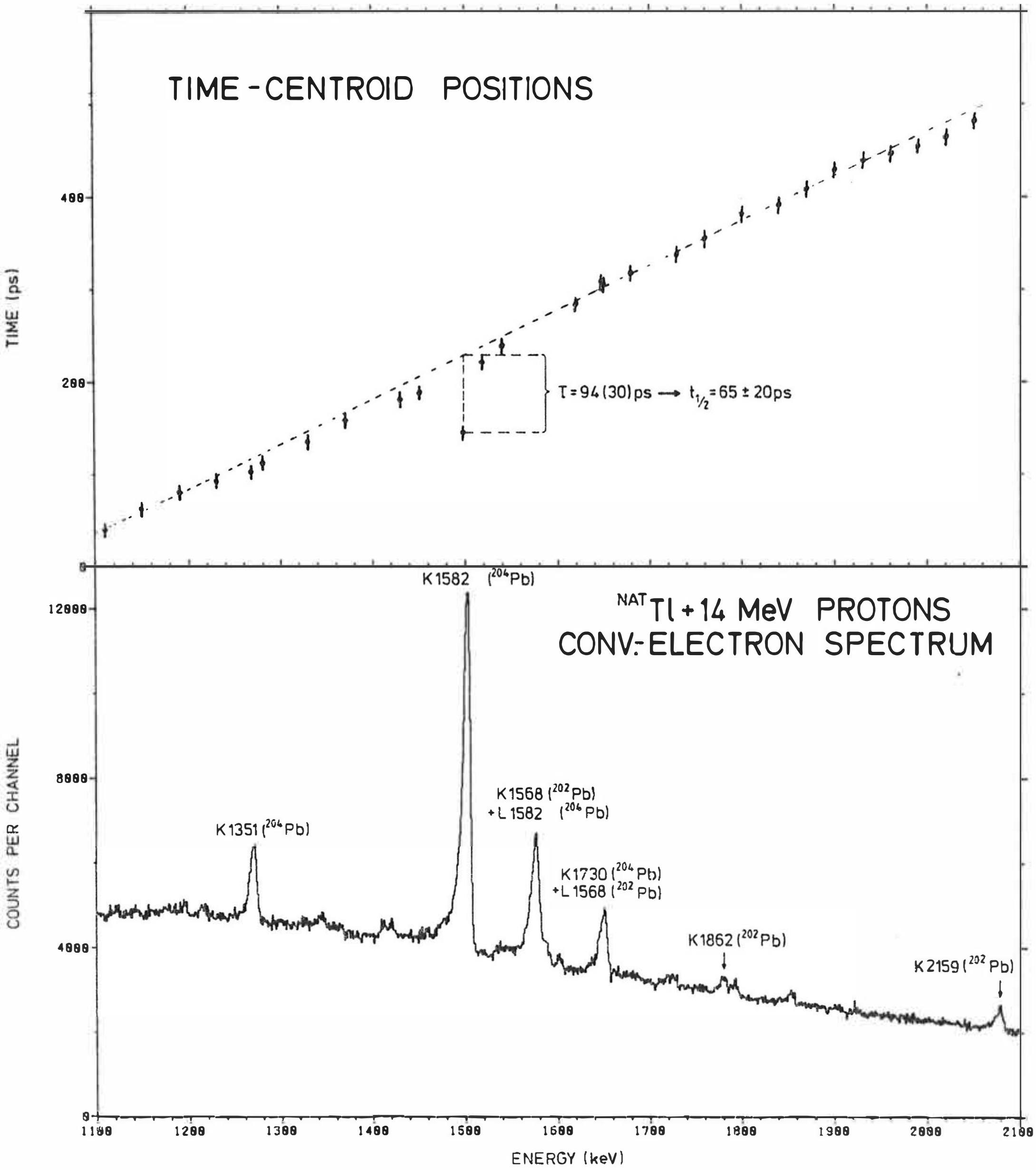


Fig. 14. The centroid-shift method. ***Bottom:*** *a 2 hours conversion-electron spectrum showing strong E0 transitions in ^{202,204}Pb following a ^{NAT}Tl(p,2n) reaction using 200 nA of 14 MeV protons and a 6 mg/cm^2 target.* ***Top:*** *time centroid positions for small sections of the spectrum shown below together with the final value of the half-life of the K 1582 keV E0 transition in ^{204}Pb.*

position is energy dependent and may be different for background events and prompt events from the target.

Figure 14 (bottom) shows a 2 hours conversion-electron spectrum from a bombardment of a 6 mg/cm^2 thick natural Tl (70% ^{205}Tl, 30% ^{203}Tl) with 200 nA of 14 MeV protons leading via the (p,2n) reaction to ^{204,202}Pb. This spectrum was recorded in coincidence with the cyclotron RF signal. Both the electron energy and the time, measured with the respect to the RF were stored on a magnetic tape. In the analysis, the energy axis was divided into small bins for which the time centroid position was determined and plotted as a function of energy (Fig. 14 - top). Clearly, the energy bin including the K 1582 peak has a significantly different centroid position, making the half-life determination possible. The half-life extraction is, however, not as straight-forward as figure 14 may imply. Beside the obvious conversion from the mean value τ to the $t_{1/2}$ ($t_{1/2} = \tau \ln 2$) one should remember that the value in the graph represents the energy bin that includes not only the peak but also a substantial amount of background events representing a different centroid shift.

The numbers given in figure 14 (top) are the final background-corrected values as obtained from a series of short measurements similar to the one described above. It was necessary to make short measurements because even small beam adjustments resulted in large time centroid shifts as compared with the measured effect.

2.5 Proton-gamma coincidence measurements

Usually, intensity limits obtained from a singles gamma-ray spectrum are sufficient to prove a correct identification of an E0 transition observed in the electron spectrum. In some cases, like in the study of ^{208}Pb, a much more selective gamma-ray technique is needed to accomplish the same task. For that

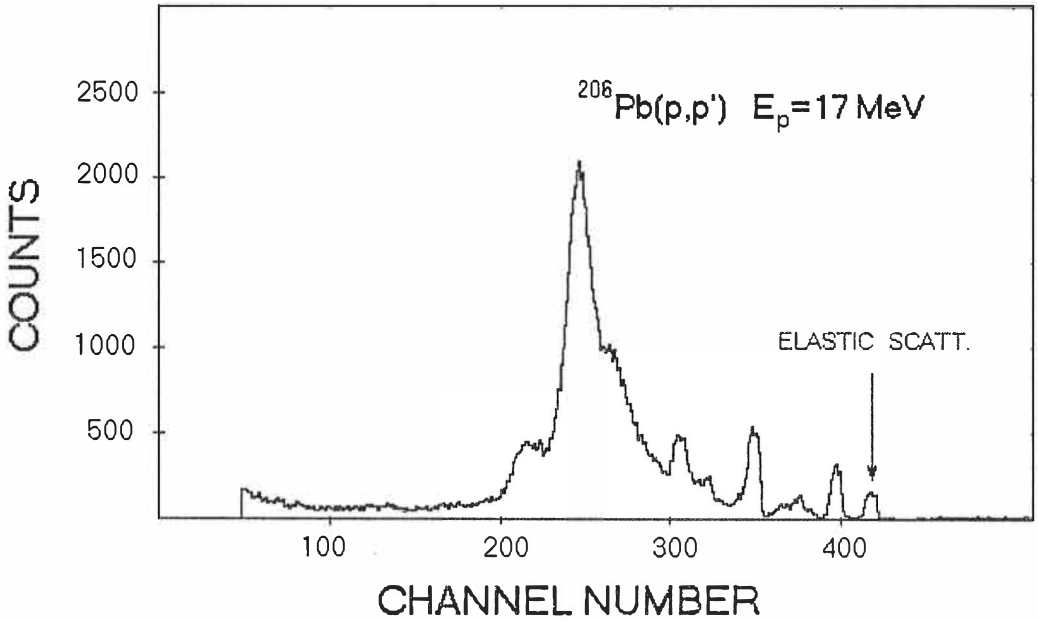


Fig. 15. *A typical particle spectrum from a 3 mm by 200 mm^2 Si(Li) operating in coincidence with a gamma-ray detector. Acquisition time was 8 hours with 1 nA of proton beam on an enriched 8 mg/cm^2 ^{206}Pb target. Position of the elastic scattering peak is marked with an arrow. Since, by definition, elastically scattered particles are not accompanied by any radiation, counts in the peak come from accidental coincidences. Rapid decline in intensity around channel number 200 comes from the fact that the low-energy protons can not exit any more through the Coulomb barrier. A thin, metallic absorber in front of the detector helped to reduce large delta-electron background that would otherwise dominate the low-energy part of the singles spectrum and would cause a high count-rate. In such a case resolution would be considerably worse and the number of accidental coincidences would increase.*

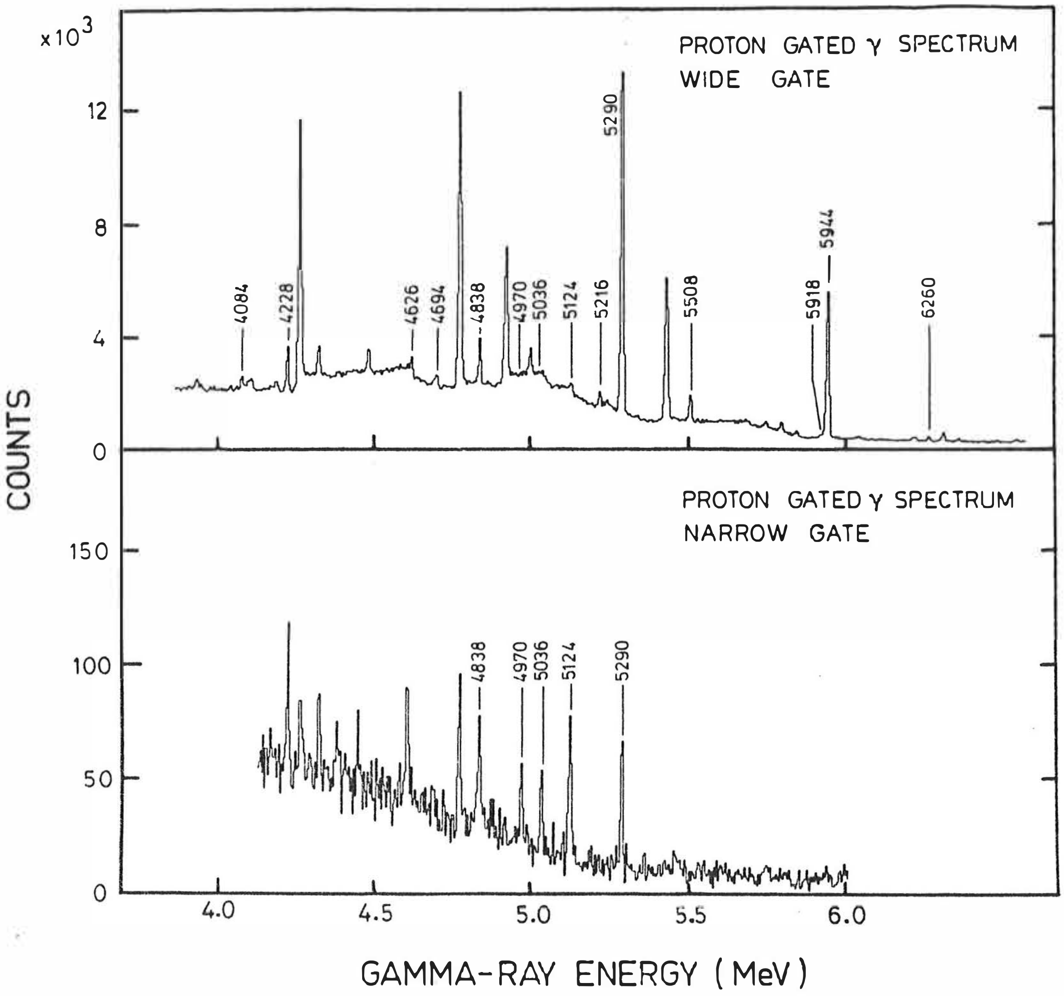


Fig. 16. *Gamma-ray spectra from a $^{207}Pb(d,p)^{208}Pb$ experiment recorded in coincidence with a particle detector.* ***Top:*** *a broad gate on the particle spectrum (similar to that in fig. 15.) produces a spectrum that does not differ much from a singles spectrum.* ***Bottom:*** *a narrow gate on the particle spectrum corresponding to about 5 MeV of excitation energy reveals a number of weak transitions and helps to associate them with nuclear levels at that energy. See fig. 22 for further details.*

purpose, a proton-gamma coincidence set-up was developed. The set-up consists of a 20% Ge detector and three 3 mm by 200 mm^2 Si(Li) detectors. The gamma-ray detector was placed at 90 degrees with the respect to the beam direction, the particle detectors were located at 135 degrees, each covering some 2.5% of the 4Pi solid angle. A thin metallic absorber in front of the Si(Li) detectors helped to reduce the very intense delta-electron flux from the target. Figure 15 shows a typical particle spectrum in coincidence with the Ge detector. The proton resolution was about 250 keV.

As has been shown earlier [KA83], a system of this kind is capable of good particle identification but for the purpose of this work, it was only used to determine the excitation energy at which the gamma-rays were emitted. Figure 16 shows the high-energy part of the gamma-ray spectrum corresponding to excitation energies in ^{208}Pb between 2.5 and 8 MeV (top) and to about 5 MeV (bottom). Clearly, the narrower gate made the identification of even weak lines possible.

3 Results and Discussion

Table 3 summarizes some of the main properties of the 0^+ states that were investigated in this work. Whenever possible, the level energies were obtained from the gamma-ray cascade rather than from the electron spectra directly. Gamma-ray measurements produce far better energy determination and are less prone to calibration errors. The competing E2 branch was also essential in the extraction of the dimensionless ratio [RA60] X of the reduced E0- to E2-transition probability:

$$X(E0/E2) = 2.56 \bullet 10^9 \; A^{4/3} \; E_\gamma^5 \; q^2 \; \alpha_K(E2) \; / \; \Omega_K.$$

The E2 energy in the above formula is expressed in MeV and the q^2 represents the intensity ratio:

$$q^2 = I_{K(E0)} \; / \; I_{K(E2)}.$$

When the half-life information was available, the square of the monopole strength parameter was calculated from the equation:

$$\rho^2 = 2.7 \bullet 10^5 \; X \;\; / \;\; T_{1/2} \; E_\gamma^5 \; A^{4/3} \; ,$$

with the gamma-ray energy expressed in keV and the $T_{1/2}$ in seconds. The strength parameter value equal to 1 is roughly equivalent to the Weisskopf estimate of gamma-ray transitions. The observed ρ^2 values are typically much smaller than 1; therefore, it is useful to express its value in milli-units (10^{-3}). Sometimes it is convenient to compare the measured ρ^2 to the single-particle unit proposed by Bohr and Mottelson [BO75]:

Table 3. Results on 0^+ states investigated in this work.

Isotope	Level en. [keV]	$t_{1/2}$ [ps]	ρ^2 [$\cdot 10^{-3}$]	X	$I_{K(E0)}/I_{K(E2)}$
^{202}Pb	1658(1)	< 30	> 4	0.08(3)	15(6)
	1862(1)	< 30	> 0.8	0.06(2)	6(2)
	2159(1)	< 30	> 2	0.5(2)	24(10)
^{204}Pb	1582.4(7)	65(20)	1.3 - 15	> 0.073	> 14
	1730(1)	< 20	> 1.3	> 0.045	> 5
	2433(2)			> 0.6	> 15
^{206}Pb	1165(2)	745(43)	0.90(16)		
^{208}Pb	4866(2)				
	5237(2)				

$$\rho_{sp} = 0.7 \ A^{-1/3}$$

hence,

$$\rho_{sp}^{\ 2} = 0.5 \ A^{-2/3},$$

where A is the atomic number. For lead isotopes it corresponds to ρ^2 between 14 and 15 milliunits. The ratio of the single-particle estimate to the measured ρ^2 gives the hindrance factor.

The ρ and X values have a direct representation in a number of nuclear models. In case of beta-vibrational transitions of deformed, spheroidal nuclei the X values have, assuming volume-conserving quadrupole surface oscillations of a uniformly-charged spheroid about an equilibrium deformation ß, a simple form [RA60]:

$$X = 4 \ ß^2 \ .$$

On the other hand, within the spherical quadrupole-phonon model, for transitions from the two-phonon 0^+ state, one gets [KU75]:

$$X = ß^2_{rms}$$

and

$$\rho = 0.151 \ Z^2 \ ß^2_{rms} \ ,$$

where $ß_{rms}$ represents a dynamic deformation.

3.1 The nucleus ^{202}Pb

In the lightest lead isotope that we were able to reach using in-beam techniques, we discovered three new [SC78, SC87] 0^+ states. In each case, the E0 transitions between these states and the ground state were observed (fig. 17). The ground state character of the E0 transitions was confirmed by additional ^{203}Tl(p,2n) excitation function measurements. For each of the states the depopulating E2 transition was found, further confirming the energy of the new state and making the X value evaluation possible. The timing centroid-shift technique [KA82b] set the half-life limits at below 30 ns. From the X values and the half-life limits the ρ^2 limits were extracted.

^{202}Pb has six neutron holes as compared with the doubly magic ^{208}Pb.

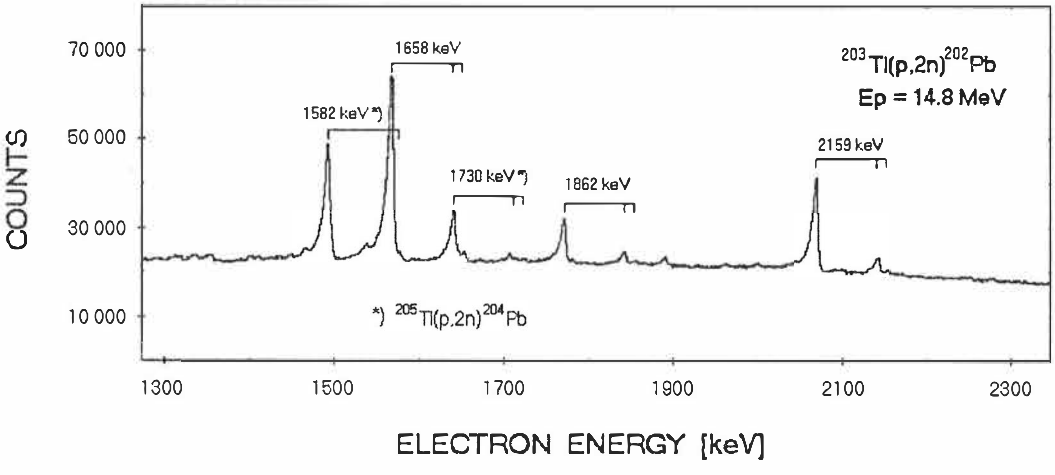


Fig. 17. A 19 hours, 25 nA, singles conversion-electron spectrum showing three new E0 transitions from three new excited 0^+ states in ^{202}Pb. The two other strong transitions (marked with an asterisk) are E0 transitions in ^{204}Pb coming from a 14% impurity in the target.

Therefore a large configuration space is required for detailed shell-model calculations for this (^{202}Pb) nucleus. Typically, in such cases, a number of simplifications is made leading to a more qualitative result. In view of that fact, the few existing shell-model calculations on ^{202}Pb present a fair picture by predicting two low-lying 0^+ quasi-particle states. Calculations by Harvey and Clement [HA71] put these 0^+ states at 1.4 and 2.3 MeV. The RPA calculations by Peltier et al. [PE74] put a 0^+ state at about 1.5 to 1.7 MeV. Even if we observe three 0^+ levels at 1658, 1862, and 2159 keV there is no major contradiction since we attribute one of them, that at 2159 keV, to the intruder states that are believed to be due to proton 2p-2h excitations. The intruder states are further discussed in chapter 3.5.

3.2 The nucleus ^{204}Pb

Also in ^{204}Pb we found three excited 0^+ states (fig. 18). The locations of the first two agree with those of previous reports [SC79]. The third excited 0^+ state at 2433 keV has been identified for the first time. We were able to measure the 65 +- 20 ps half-life of the 1582 keV state and set an upper limit of 20 ps for the 1730 keV state. The third E0 transition, depopulating the newly discovered 0^+_3 state, had too small a peak-to-background ratio for any meaningful half-life determination. Also, only a lower limit for the X value could be set.

Similarly as for ^{202}Pb, available shell-model calculations account fairly well for the 0^+_2 and 0^+_3 states. A quasi-particle model [HA71], a conventional shell model with four neutron holes [MG75], and a multi-step shell model [LI81] all predict, as observed, two 0^+ states below 2 MeV. A weak coupling model [KO73] yields only one 0^+ state below 2 MeV. However, none of these calculations could account for the 0^+_4 state.

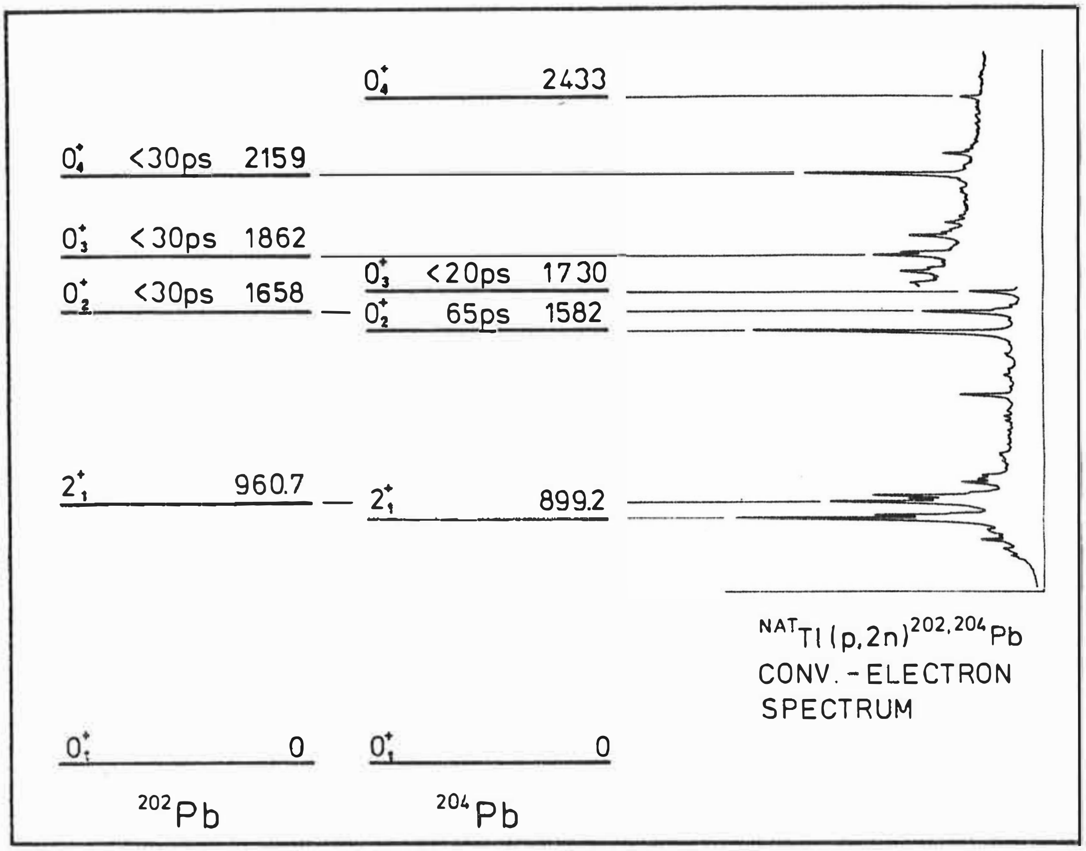


Fig. 18. Partial level scheme of ^{202,204}Pb. The energies of 0^+ states are shown as measured in this work. On the right, there is a section of a singles conversion-electron spectrum from 14.5 MeV bombardment of a natural thallium target (29.5% ^{203}Tl, 70.5% ^{205}Tl) with protons. The level positions are matched to the K lines of the corresponding ground-state transitions in the spectrum.

3.2.1 Solution of the controversy over the 1582 keV level

In a measurement by Goldman et al. [GO70], performed with an orange-type spectrometer, having an energy resolution of 18 keV in the 1.5 MeV range, a 1585 keV 0^+ state was reported from the 205Tl(p,2n)204Pb reaction at

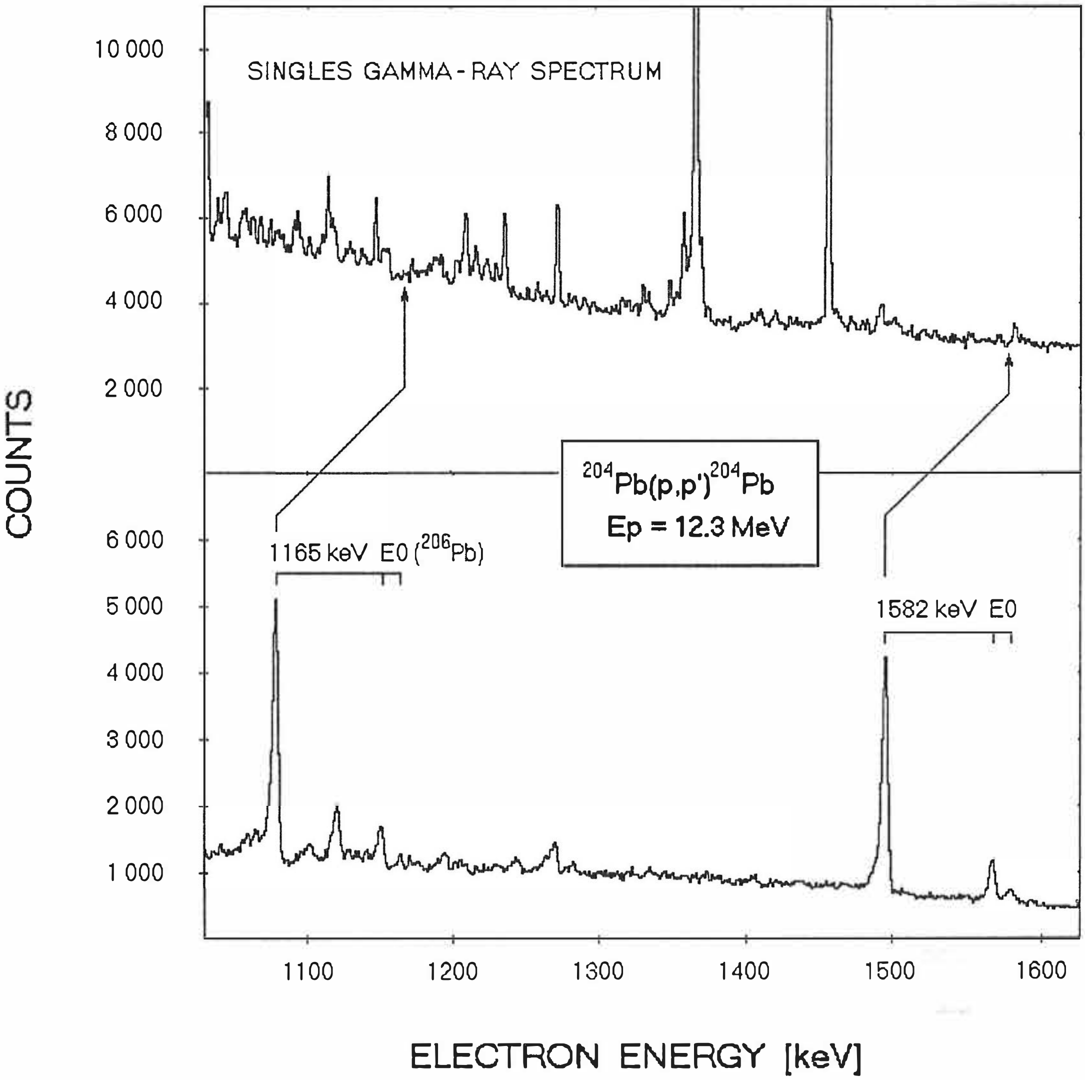


Fig. 19. *Strong E0 transitions populated in (p,p') reaction at proton energy of 12.3 MeV on a 67% enriched* ^{204}*Pb target (includes 16% of* ^{206}*Pb).* ***Top:*** *a 4 hours, singles gamma-ray spectrum from a 3 mg/cm*^{2} *target bombarded with 0.5 nA beam.* ***Bottom:*** *a 2 hours, singles conversion-electron spectrum acquired in a swept mode (150 keV - 1750 keV range) from a 1.5 mg/cm*^{2} *target bombarded wit 75 nA beam. The location of E0 transitions dominating the electron spectrum does not overlap with any visible lines in the gamma-ray spectrum.*

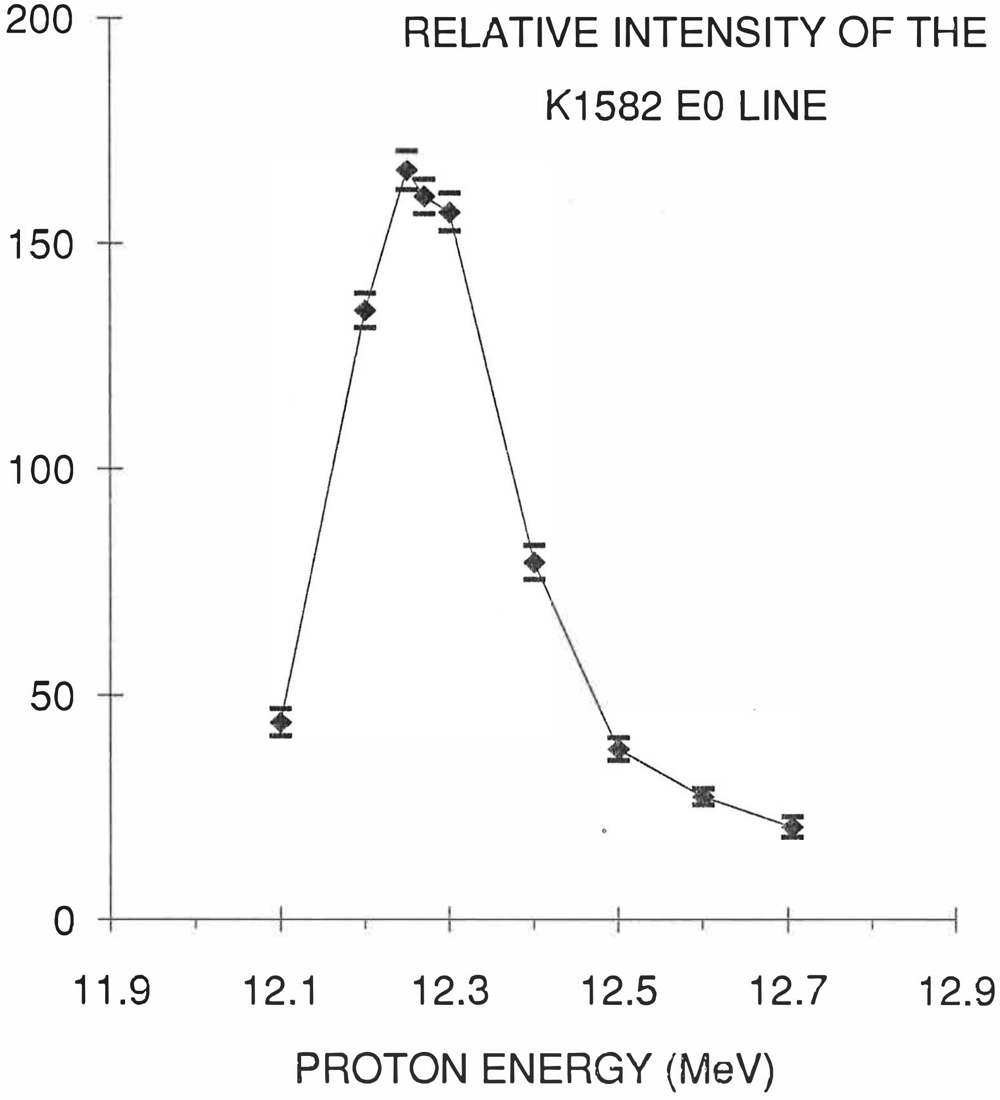


Fig. 20. *Relative intensity of the K 1582 keV E0 line from the* ^{204}*Pb(p,p')*^{204}*Pb reaction at proton energies around the 12.3 MeV IAR. Presence of such a sharp resonance excludes a possibility that the observed line comes from an other nucleus populated, for example, by a (p,t) or a (p,n) reaction. Intensity of the K 1582 keV is also directly linked with the* ^{204}*Pb enrichment of the target.*

the proton energy of 16 MeV. The poor quality of the conversion-electron data from the orange spectrometer left some doubt about the spin and energy assignment. Our measurement [KA86], using the same reaction and a similar beam energy, confirmed beyond doubt the original 0^+ assignment. In addition, the γ-γ coincidence spectra revealed the existence of a 683.5 keV gamma-ray transition depopulating a level at 1582.7 keV. This energy matched with the energy of the E0 transition from our electron spectra. Therefore, we assumed that the 683.5 keV transition depopulates the 1582 keV 0^+ level and, thus, extracted the $\rho^2_{21} = 7.4(25) \cdot 10^{-5}$ and the $X_{211} = 0.0032(3)$.

However, data from a later study [HA88] of ^{204}Pb via the (n,n'γ) reaction produced contradicting results concerning the 1582 keV level: its spin and parity were firmly established as 2^+. The authors of this experiment could not find evidence for a gamma-ray transition between a 0^+ level at an energy close to 1582 keV and the 899.2 keV 2^+_1 level. This means that, unless either of the measurements was incorrect, there should indeed be two levels at nearly identical energies, both a 0^+ and a 2^+ level. The 2^+ energy was reported at 1582.8 +- 0.1 keV.

To solve this controversy we complemented our earlier measurements by populating the levels in ^{204}Pb in the (p,p') reaction at proton energies close to and equal to the IAR energy [LE68] of 12.3 MeV. A 66% enriched ^{204}Pb target was used. The electron spectra displayed again a prominent E0 transition (fig. 19) that resonates at the IAR energy (fig. 20). This time the data indicated a decrease of the ratio of the 683.5 keV gamma-ray to the K1582 keV E0 line by a factor of 25 as compared with the results from the ^{205}Tl(p,2n)^{204}Pb experiment. This, together with the results from the (n,n') experiment [HA88], clearly proves the existence of two different decay modes and, therefore, two different levels in ^{204}Pb at the excitation energy of 1582 keV.

Table 3 (p.36) includes the current characteristics of the 0^+ state. A relatively poor energy determination of the 0^+ level (1582.4 +- 0.7 keV) comes from the fact that it was solely based on the electron measurement. There is no information as to whether the 683.5 keV gamma-ray depopulates the 0^+ or the 2^+ level, or both. This is also the reason why only limits on the I, X, and ρ^2 values can be given.

3.3 The nucleus ^{206}Pb

It is well established from the nucleon transfer studies that there are at least four excited 0^+ states in ^{206}Pb below 6 MeV [WE79]. The 1165 keV 0^+_2 has been already seen in electron measurements and its half-life has been measured as 745 +- 43 ps [TA72, JU76]. No other E0 transitions were ever observed. We made a considerable experimental effort to change this situation. The 1 - 6 MeV energy range was carefully scanned for conversion electrons from the ^{206}Pb(p,p') reaction on a 90% enriched, metallic, self-supporting target. The runs were made with proton energies around 12, 15, and 18 MeV corresponding to the isobaric analog resonance (IAR) energies. Each run lasted at least 20 hours. The sensitivity of the measurements was so good that the 4866 keV E0 transition in ^{208}Pb, discussed in the next chapter, produced from a 3% ^{208}Pb impurity in the target could be seen (fig. 21). Also an unidentified 3542(2) keV transition was resolved from the background (fig. 13) but no evidence was found for any E0 transitions in ^{206}Pb other than the 1165 keV. In principle, the 3542 keV could be a candidate for an E0 transition. However, since there is a weak gamma-ray at that energy, we are, at the moment, not able to rule out other multipolarities.

There are two simple reasons that can explain our inability to detect the E0 transitions in ^{206}Pb: (i) the 0^+ states are not sufficiently populated by inelastic scattering of protons, or (ii) there is only a weak E0 branch to the ground state from the decay of these states.

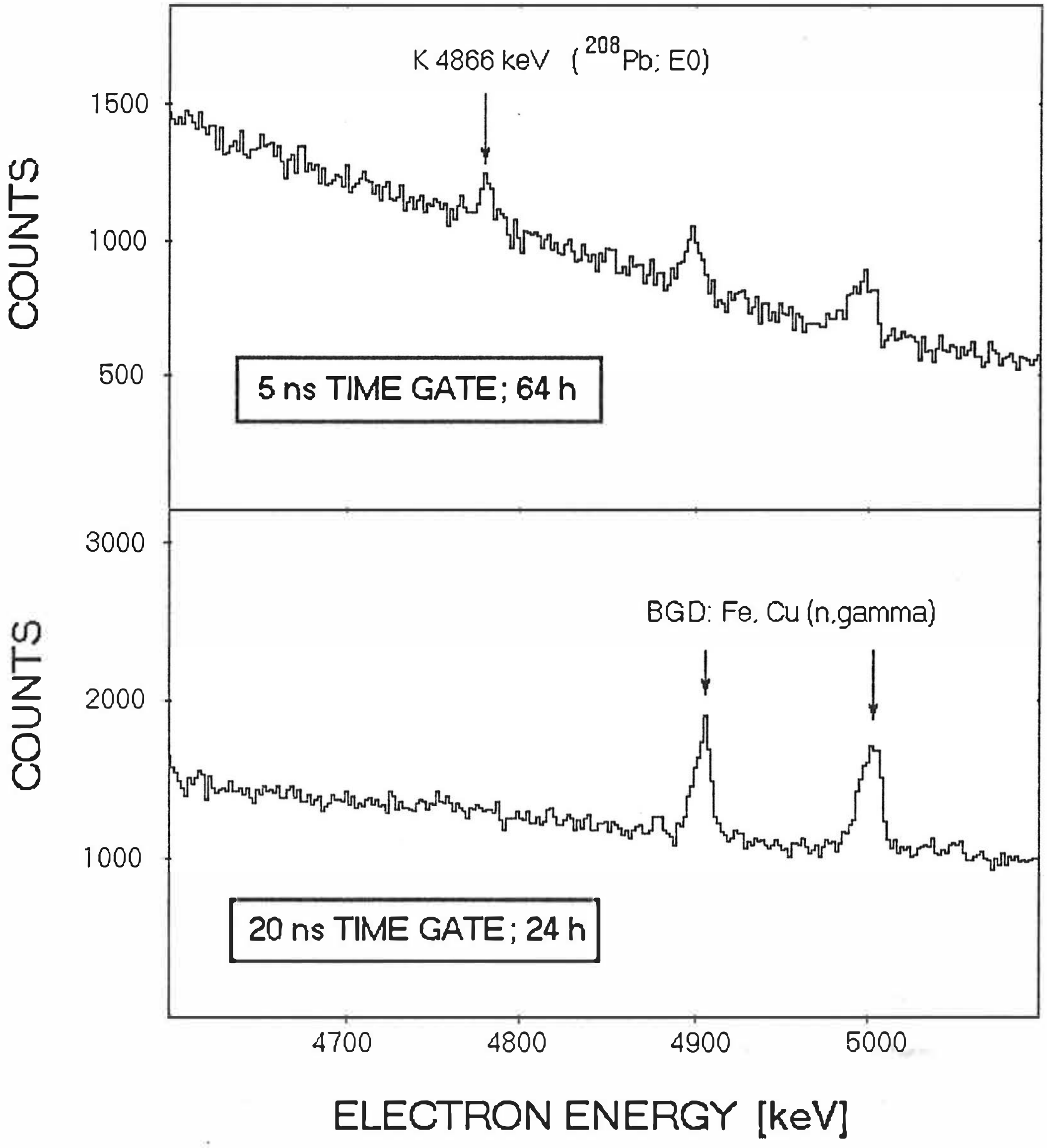


Fig. 21. Good sensitivity of the high-energy electron spectroscopy achieved by various background reducing techniques including critical time-gating of the electron spectra (top vs. bottom). Both spectra were recorded in a swept-mode of operation (1.7 - 6.3 MeV range) using a 5 mg/cm^2 90% enriched ^{206}Pb target bombarded with 50 nA of 17.3 MeV protons. The marked, weak transition is the K 4866 keV E0 in ^{208}Pb. It is populated by ^{208}Pb(p,p') on only a 3% impurity in the target. Despite such a high sensitivity no evidence for E0 transitions in ^{206}Pb in that energy range was found.

It is difficult to advocate any of these arguments without more data. There is apparently a strong direct population of the 0^+ states in (p,t) [FL67, SM70, LA77], (t,p) [FL74], (^{3}He,n) [AN77] and (p,d) [LA74] reactions. Unfortunately, all of these reactions were beyond our experimental approach. For the ^{204}Pb and ^{208}Pb the (p,p') reaction showed clear population of the 0^+ states but we can not be certain whether it was a direct population or whether they were reached via decay of the higher levels. In the latter case one should not be surprised by differences in feeding of the states below. Still, even if an excited 0^+ state is reached it does not exclusively decay to the ground-state. It is feasible to account for most of the strength diverted from the E0 branch.

3.4 The nucleus ^{208}Pb

In ^{208}Pb three excited 0^+ states are expected below 6 MeV: the neutron-pairing-vibrational state, the proton-pairing-vibrational state, and the two-octupole-phonon state. First a 4.87 MeV 0^+ state was observed [BJ66] in the (t,p) reaction. A later study [IG71] of ^{208}Pb, also involving two-neutron transfer reactions, established two 0^+ levels situated at 4859(15) keV and 5236(15) keV. The former, confirmed from the earlier experiment, was unambiguously assigned [BR67] as the neutron-pairing-vibrational state, but the latter was alternately identified as proton-pairing-vibrational state or as two-octupole-phonon state (see chapter 3.4.1).

To search for the E0 transitions we used the ^{207}Pb(d,p) reaction at 10 MeV measuring both the conversion-electron spectra and the proton-gamma coincidences (fig. 22). The gamma-ray spectrum (fig. 22 b) is dominated by escape peaks and Compton tails from the strongly populated 5292 keV and 5946 keV neutron particle-hole 1^- states in ^{208}Pb [MA86]. Fortunately, selective gating of the proton energy revealed even very weak lines in the gamma-ray

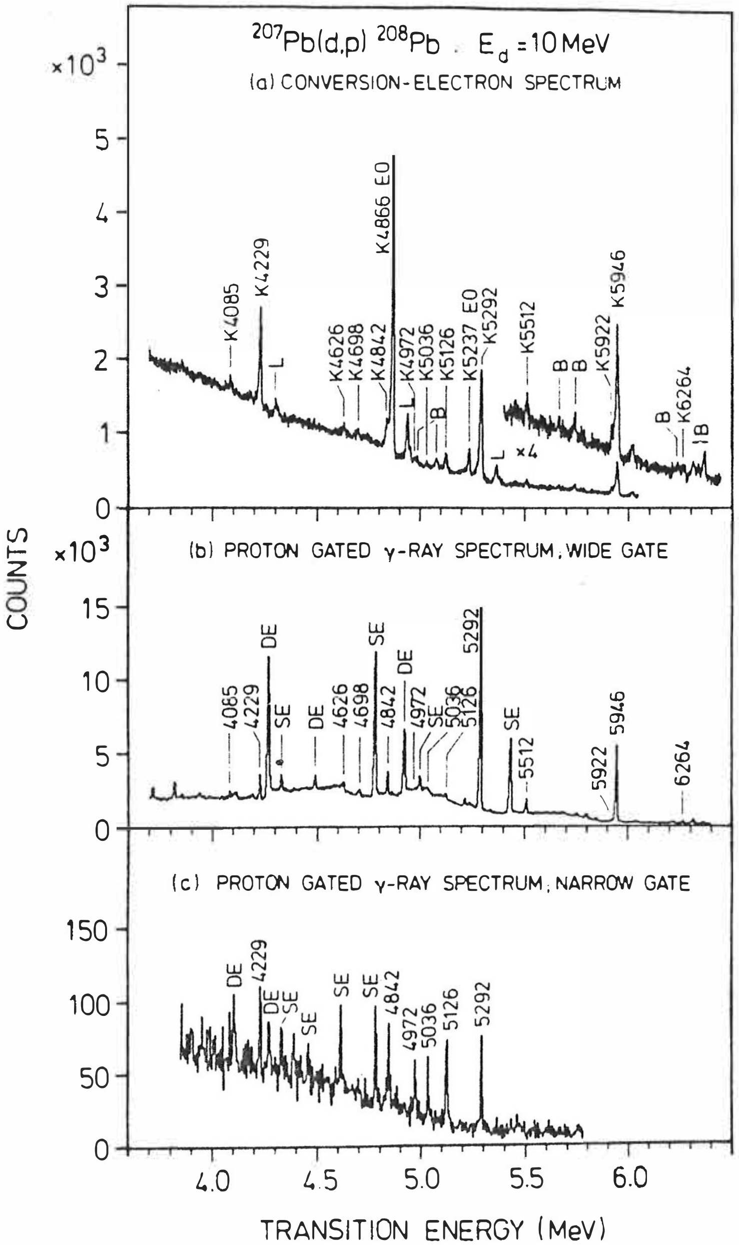


Fig. 22. *High-energy parts of conversion-electron and proton-gated gamma-ray spectra from a bombardment of* ^{207}*Pb with 10 MeV deuterons. In (a), B indicates background lines. The gate settings for (b) and (c) correspond to excitation energies of 2.5-8 MeV and about 5 MeV in* ^{208}*Pb, respectively.*

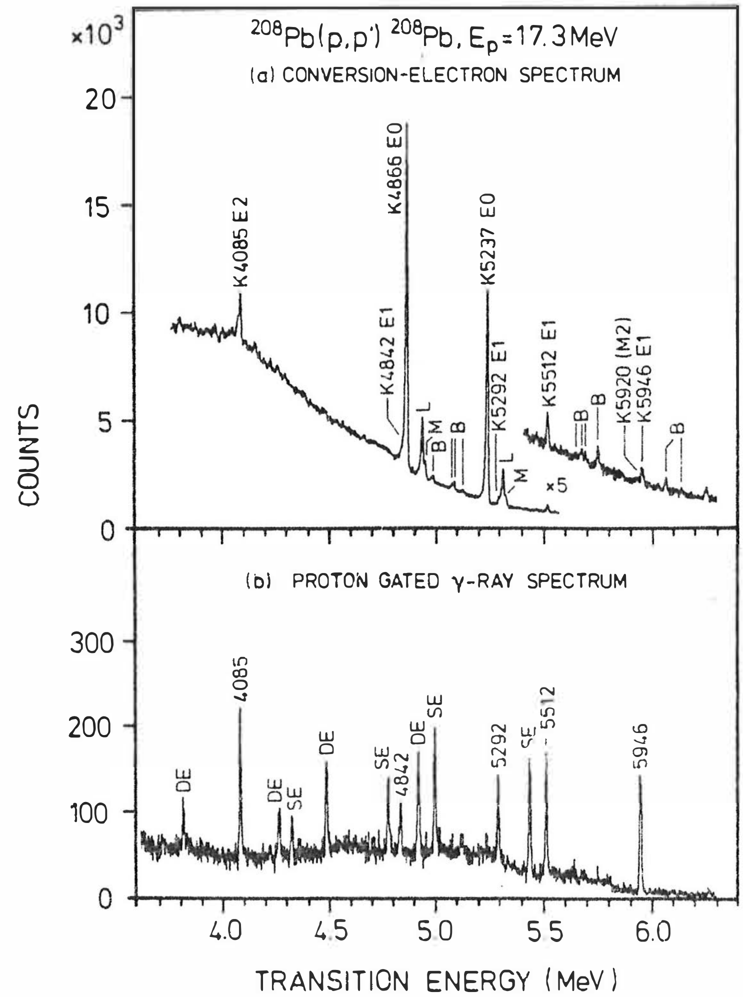


Fig. 23. *High-energy parts of conversion-electron and proton-gated gamma-ray spectra from a bombardment of* ^{208}*Pb with 17.3 MeV protons. In (a), B indicates background lines. The gate setting in (b) corresponds to excitation energies of 4-6 MeV.*

spectrum (fig. 22 c). Several known ground state transitions were identified both in the gamma-ray and in the conversion-electron spectrum: E1 from the 4842, 5292, 5512, 5946 and 6262 keV 1^- states, E2 from the 4085 keV 2^+ state, M2 from the 4229 keV 2^- state. Also the 4626 keV E1 in ^{207}Pb was present in the spectra. These transitions provided internal energy calibration good to about 2 keV. Using the internal calibration in gamma-ray and in electron spectra, together with information on proton energy, the identification of previously unobserved ground-state transitions from several known states could be done: 4698 keV 3^-, 4974 keV 3^-, 5038 keV $(2^-,3^-)$ and 5923 keV $(2)^-$.

The conversion-electron spectrum (fig. 22 a) revealed also two new E0 transitions: the 4866(2) keV, and the 5237(2) keV. We have identified the first one - the strongest line in the spectrum - as the ground-state decay of the neutron-pairing-vibrational 0^+ state in ^{208}Pb - known from the particle transfer studies as 4859(15) keV. The second one, relatively weak, we believe to be the ground-state decay from the second excited 0^+ state that was seen in the transfer studies at 5236(15) keV. Rough estimates, derived from the spectrum in figure 22 a, of the total cross section for the production of the E0 K-conversion lines are 5 and 0.5 μb, respectively, for the 4866 and the 5237 keV transitions. No evidence for the E0 decay of a third 0^+ state was found.

In another experiment, we repeated the measurements of conversion-electrons and proton-gamma coincidences using the ^{208}Pb(p,p') reaction (fig. 23). The proton energy was 17.3 MeV. This time, both above-mentioned E0 transitions were prominent in the electron spectrum (fig. 23 a) yielding the K-conversion production cross-section of about 10 and 6 μb. It should be reminded that these numbers indicate remarkable selectivity for 0^+ states in the 5 MeV region. Clearly, the 4866 and 5237 keV E0 dominate the spectrum (fig. 23 a) even if states of other spin and parity are populated with cross-sections higher by two orders of magnitude (about 1 mb).

Again, in spite of good selectivity, no evidence for the ground-state E0 decay of a third 0^+ state in the 5 MeV region was found. Competing decay-branches of such a state would probably include an E1 transition to the 4842 keV 1^- state, an E2 transition to the 4085 keV 2^+ state, and an E3 transition to the 2615 keV one-phonon state. Estimates by J. Blomqvist [BLPC] indicate that even the two-octupole-phonon state, with a large collective E3 branch to the 2615 keV 3^-, would have the ground-state E0 component of the order of 20% of the total deexcitation of such a state. Using this estimate we set about 1 μb as the upper limit for the population of the missing 0^+ state in the 4-6 MeV region in ^{208}Pb in our (p,d) and (p,p') experiments. This limit is not valid if the energy of the missing E0 coincides with one of the known transitions.

Proton-gated gamma-ray spectra from the (p,p'gamma) experiment show similar quality as the (p,d gamma) spectra. The 4842, 5292, 5512, and 5946 keV E1 transitions and the 4085 keV E2 transition were identified (fig. 23 b). Unfortunately, gamma-ray intensities from the competing transitions deexciting the 0^+ states were still below the detection limits.

Additional (p,p'e^-) measurements showed no significant intensity fluctuations of the E0 lines for proton energies between 17.0 and 17.5 MeV. The same observation followed (d,pe^-) runs with deuteron energies from 9.5 to 10.0 MeV. This indicates lack of strong resonances such as those observed in the ^{204}Pb(p,p') for the 1582 keV 0^+ state.

3.4.1 Search for two-octupole-phonon 0^+ state

In the doubly magic ^{208}Pb, the first excited state is the 2615 keV 3^- state. It is a textbook case of a collective one-phonon, octupole vibration ($\lambda = 3$). A simple liquid drop model predicts [CO71] that two-phonon vibration would produce a quartet of closely located states, one of them having spin and parity 0^+, at twice the energy of the one-phonon state. In the case of ^{208}Pb this corresponds to the unperturbed two-phonon energy of 5230 keV - strikingly close to 5237(2) keV that we report for the second excited 0^+ state. This brings up an intriguing question whether the 5237 keV 0^+ state is indeed the two-phonon-octupole state or the proton-pairing-vibrational state as it was suggested earlier.

Unfortunately, model calculations do not seem to bring enough clue to solve this problem. At first, the 5237 keV state (then 5236(15) keV) was believed to be the two-phonon state [IG71]. This picture was changed when the 0^+ member of the two-phonon quartet was predicted [BL70] to come as far down in energy as 3.7 MeV. However, new calculations [BLPC], using the improved experimental value of the quadrupole moment, put the estimate back again - close to the unperturbed energy. Predictions [WOPC] for the proton-pairing-vibrational state give the same energy range: 5.2-5.3 MeV.

On the other hand, predictions based on the known [EL78, MA83a, JO81] excitation energies of the related states in the neighboring odd-proton isotones (^{209}Bi and ^{207}Tl) suggest proton-pairing-vibrational state at around 5.5 MeV (fig. 24, top). Since similar estimates for the neutron-pairing-vibrational state using excitation energies in ^{209}Pb and ^{207}Pb agree well with the measured energy of 4866(2) keV (fig. 24, bottom) it would bc surprising to see a 0.25 MeV deviation for energy of the proton-pairing-vibrational state.

Estimates based on binding energies give comparable excitation energies for the proton- and neutron-pairing-vibrational states to those calculated in fig. 24. Using the ground-state binding energies of two-particle and two-hole nuclei around ^{208}Pb taken from The 1983 Atomic Mass Evaluation [WA85], one gets a first approximation to the excitation energies:

for neutrons (n-p-v):

$$E(^{210}\text{Pb}) + E(^{206}\text{Pb}) - 2E(^{208}\text{Pb}) = 4983 \text{ keV}$$

for protons (p-p-v):

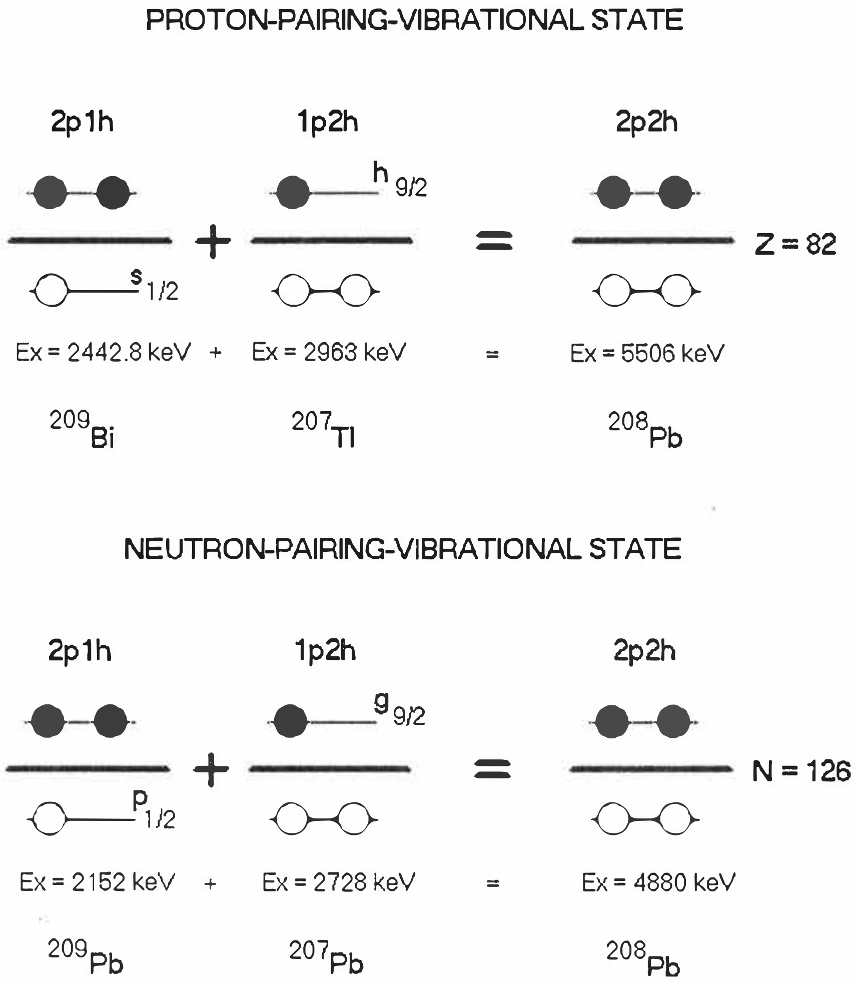

PROTON-PAIRING-VIBRATIONAL STATE
2p1h
1p2h
2p2h
h 9/2
+
=
Z = 82
s 1/2
Ex = 2442.8 keV + Ex = 2963 keV = Ex = 5506 keV
209Bi
207Tl
208Pb
NEUTRON-PAIRING-VIBRATIONAL STATE
2p1h
1p2h
2p2h
g 9/2
+
=
N = 126
p 1/2
Ex = 2152 keV + Ex = 2728 keV = Ex = 4880 keV
209Pb
207Pb
208Pb

Fig. 24. *Estimation of the energy of the proton-pairing-vibrational state (top) and the neutron-pairing-vibrational state (bottom) in ^{208}Pb. In the latter case this simple prediction agrees very well with the observed energy of 4866 keV. The other state has not yet been observed.*

$$E(^{210}Po) + E(^{206}Hg) - 2E(^{208}Pb) = 6584 \text{ keV}$$

Such an approximation does not take into account interaction between the 0^+ particle pair and the 0^+ hole pair. In the case of neutrons the interaction involves exclusively short-range nuclear forces and therefore, within the first approximation, the resulting energy correction in this fairly large nucleus can be neglected. Indeed, the estimated 4983 keV is close to the measured 4866(2) keV and to 4880 keV calculated in fig. 24. In case of protons the particle-hole interaction includes long-range Coulomb forces and therefore can not be neglected. Fortunately the Coulomb-energy correction is easy to estimate:

$$DE_{Coul} = - 4ke^2 / R$$

where k is a constant, e is electron chargc and R is the radius. Since R = $1.2A^{1/3}$ fm, the estimated Coulomb correction is:

$$DE_{Coul} = - 810 \text{ keV}$$

and the excitation energy of the p-p-v becomes:

$$6584 - 810 = 5770 \text{ keV}$$

Again, reasonably close to 5506 keV from fig. 24. One can go a step further in energy estimation by evaluating the particle-hole interaction. It can be done by comparing the binding-energy estimates with the measured excitation energies listed in fig. 24. For the $g_{9/2}$ * 0^+ (^{206}Pb, g.s.) 2728 keV state in ^{207}Pb one obtains:

$$E(^{209}Pb) + E(^{206}Pb) - E(^{208}Pb) - E(^{207}Pb) = 2792 \text{ keV}$$

and, for the $p_{1/2}^{-1}$ * 0^+ (^{210}Pb, g.s.) 2152 keV state in ^{209}Pb,

$$E(^{210}Pb) + E(^{207}Pb) - E(^{208}Pb) - E(^{209}Pb) = 2182 \text{ keV}.$$

The differences are -64 keV and -30 keV, respectively, indicating the average value of the $g_{9/2}$ particle - $p_{1/2}$ hole interaction energy of -23.5 keV. The n-p-v in ^{208}Pb has four interacting p-h pairs, therefore:

$$E(n\text{-}p\text{-}v) = 4983 - 4 \bullet 23.5 = 4889 \text{ keV},$$

as compared to the observed 4866(2) keV. The same procedure for the proton states yields:

$$E(^{209}Bi) + E(^{206}Hg) - E(^{208}Pb) - E(^{207}Tl) = 3569 \text{ keV}$$

for the $g_{9/2}$ * 0^+ (^{206}Pb, g.s.) 2963 keV state in ^{207}Tl, and

$$E(^{210}Po) + E(^{207}Tl) - E(^{208}Pb) - E(^{209}Bi) = 3029 \text{ keV}$$

for the $p_{1/2}^{-1}$ * 0^+ (^{210}Pb, g.s.) 2443 keV state in ^{209}Bi. The average particle-hole interaction is -298 keV, therefore:

$$E(p\text{-}p\text{-}v) = 6584 - 4 \bullet 298 = 5392 \text{ keV}.$$

This value is 0.1 MeV below the estimation from fig. 24, but still almost 0.2 MeV above the measured 5237(2) keV of the 0^+_3 . Similar calculations, but not based on the latest data, are reported in ref. [BL70].

There is yet another qualitative argument favoring the two-octupole-phonon interpretation of the 5237 keV 0^+ state. We see a tenfold increase in intensity of the K conversion ground state E0 decay branch from this state in our (p,p') spectra as compared to the (d,p) experiments. It coincides with the expectation that the two-octupole-phonon state should be much easier to form in the (p,p') than the proton-pairing-vibrational state [BLPC].

However, none of these arguments is conclusive. Probably the discovery of the third excited 0^+ state may clarify the nature of the 5237 keV 0^+ state. It is also interesting to reflect that such a seemingly straightforward case as two-octupole-phonon excitation in a magic nucleus is so difficult to predict (calculate) and to identify. The failed attempts include a search with the BILL spectrometer [MA78] following $^{207}Pb(n_{th},\gamma)$ reaction [MA83b]. A similar situation exist in the semi-magic ^{146}Gd where two 0^+ states have been identified [YA87] at the excitation energy close to the expected two-octupole-phonon energy but no decisive assignments were made.

Now, however, it seems that the latest multistep shell-model calculations [CR88] have ended 20 years of the controversy surrounding the two-octupole-phonon state in ^{208}Pb. Motivated by our high-resolution data [JU87] that included relative intensity and cross-section estimates, the authors of this new work also performed the DWBA calculations. In conclusion they state that the two excited 0^+ states, reported both in inelastic proton scattering [JU87] and in two-neutron transfer reaction [IG71], correspond to an admixture of a two-octupole vibration and a two-neutron pairing excitation. It is claimed that the proton-pairing-vibrational 0^+ state is predicted to lie at about 5.5 MeV (it agrees with the prediction from figure 24). It is also stated that this state is virtually a pure two-proton pairing vibration and, therefore, it is neither excited by the reactions used in our experiments [JU87] nor in the (t,p) reaction [IG71].

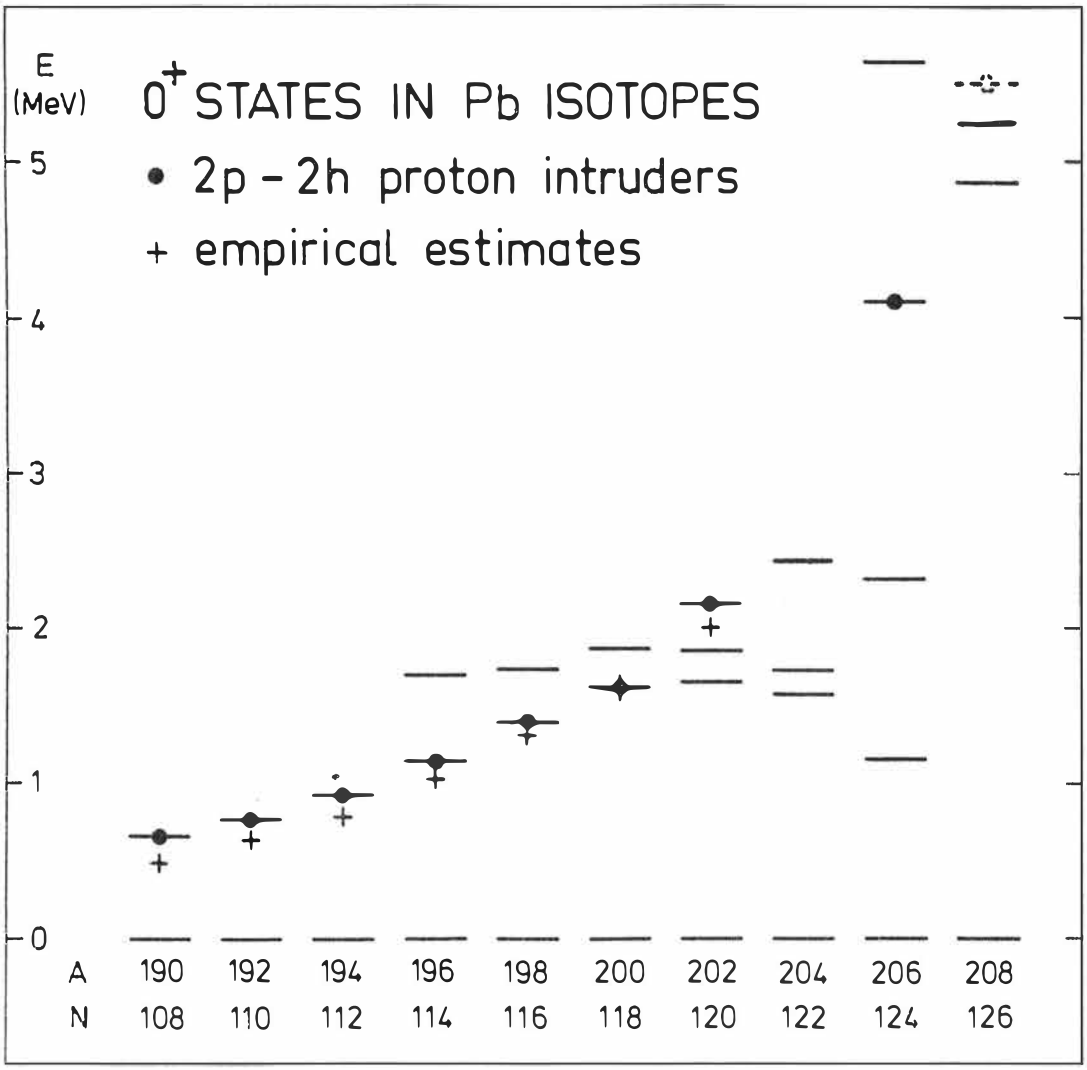


***Fig. 25.** Energies of all known 0^+ states in even-even lead isotopes $^{190\text{-}208}$Pb. The candidates for the proton 2p-2h intruder states are indicated by dots. Crosses represent empirical estimates for the intruder energies according to ref. [DU84]. Dashed line in ^{208}Pb marks the estimated location of the yet undetected proton 2p-2h state in that nucleus.*

3.5 Intruder states

All the lead isotopes have a magic number of protons representing the closed Z=82 shell. Therefore, one would expect all of the low-energy states, especially for lighter isotopes, far from the doubly magic ^{208}Pb, to have a neutron character, i.e. to be built from neutron hole terms. Any configuration involving protons would require excitations over the energy gap into the next shell. A 0^+ state would need at least two protons brought to the higher shell. This state, the proton-pairing-vibrational state, made out of two protons in the $h_{9/2}$ and two holes in the $s_{1/2}$, has in ^{208}Pb about 5.5 MeV excitation energy. In an over-simplified model one would expect this energy to remain constant for all the lead isotopes. This, however, is not true. As a result of neutron-proton interaction, leading to a deformation of the nucleus, the energy of the proton-pairing-vibrational state is pushed down. Thus it becomes an intruder state.

Judging from the spectra alone it is impossible to determine whether a particular state is an intruder. There is no universal fingerprint that could be linked with such a state. The only indication, so far, comes from the energy systematics and semi-empirical estimates. Fortunately, thanks to the measurements by Van Duppen et al. [DU84, DU85] and this work, there is an extensive base for such an evaluation. Figure 25 shows all known 0^+ states in even-even lead isotopes $^{190-208}$Pb. Crosses represent empirical estimates for the intruder energies according to ref. [DU84]. The candidates for the proton 2p-2h intruder states are indicated by dots. The dashed line in ^{208}Pb marks the estimated location of the proton 2p-2h state in ^{208}Pb. As discussed in chapter 3.4.1, the observed second excited 0^+ state is, in view of the new calculations, the two-octupole-phonon state rather than the proton 2p-2h state.

Another clue in identification of intruder states comes, if available, from the spectroscopic characteristics [KA84]. In general, proton states tend to have larger X values than neutron states. Naturally, the best indication would come from the intensity comparison obtained from a two-proton and two-neutron transfer study but such data are present only for ^{206}Pb.

In ^{202}Pb the likely candidate for the intruder state is the 0^+_4 state. It has the largest X value and its energy agrees well with the new systematics [HE87]. It is not clear how strong is the mixing between the relatively closely spaced 0^+ states in ^{202}Pb. A possible mixing could alter the unperturbed energies and, therefore, argumentation based on energy systematics alone seems weak. Nevertheless, unlike tentative assignments to the intruder states in ^{204}Pb and ^{208}Pb, the one in ^{202}Pb remains uncontested.

As discussed in paragraph 3.2, the following theoretical calculations predict, in agreement with the present observation of the 0^+_2 and 0^+_3 states in ^{204}Pb, two excited 0^+ levels below 3 MeV in the four neutron-hole space: a quasi-particle model [HA71], a conventional shell model [MG75], and a multi-step shell model [LI81]. The remaining 2433 keV 0^+_4 state is not accounted for in these calculations. Its energy agrees with a possible location of the proton $\{1/2^+[440]^{-2}, 9/2^-[514]^2\}$ intruder state, as suggested by systematics. This was therefore the tentative conclusion in our earlier work [KA86]. However, there is new evidence contradicting such an assignment. The (n,n'γ) study [HA88] showed a 751.8 keV transition exhibiting isotropic angular distribution, with an excitation threshold of 2450(50) keV, in cascade with the ground-state transition from a 1^+ level at 1681.2 keV. This cascade probably belongs to the decay of the 0^+_4 state. Thus, as already pointed out in ref. [HA88], the 0^+_4 state could belong to the four neutron-hole valence space (as apparently does the 1681.2 keV 1^+ state). There is one more argument in support of this possible new interpretation of the 0^+_4 state: from one of the two-neutron transfer studies [LA77], a state (without spin assignment) at 2430 keV is reported. This state is probably the 2433 keV 0^+ level. A proton-intruder state is not likely to be populated in two-neutron transfer (except via configuration mixing). Finally, the new, detailed fits [HE87] based on intruder-state systematics both in Pb and Bi isotopes suggest 3.2 MeV for the intruding 0^+ state energy in ^{204}Pb - considerably more than the 2.433 MeV of the 0^+_4 state.

The presence of three excited 0^+ states belonging to the neutron-hole space would not be entirely unexpected. According to early calculations by True [TR68], there should be three relatively low-lying, excited 0^+ states even in ^{206}Pb with only two quasi-particles. Clearly, improved shell-model calculations are called for to reproduce the energies.

The possible absence of the proton 2p-2h state from the known 0^+ levels in ^{204}Pb leaves a gap in the experimental intruder-state systematics in the **Z** = 82 region. A similar situation exists now in ^{208}Pb, as well: recent calculations [CR88] identify the 5237 keV 0^+ state in ^{208}Pb as a two-octupole-phonon vibrational excitation and validate 5.5 MeV as the expected energy of the unobserved two-proton pairing vibrational 0^+ state.

3.6 K/L ratios of E0 transitions

Internal-conversion processes leading to the emission of a conversion electron instead of a gamma-ray as a way of nuclear de-excitation are well understood. There are extensive tables [KA89, RO78, BA76, TR72] of calculated internal-conversion coefficients (ICC) that are in agreement with experiments to better than a few per cent. However, in the case of E0 transitions, there is no competing gamma-ray transition between the two 0^+ states and, therefore, there is no ICC to be compared with the experiment. Consequently, unlike the higher multipolarities, the theory of the electric monopole transition has not been thoroughly verified by measurements.

At the present, the only numbers that can be scrutinized are the relative emission probabilities for electrons from different atomic sub-shells: K, L1, L2, etc. To a good approximation, excluding the second order terms from the matrix

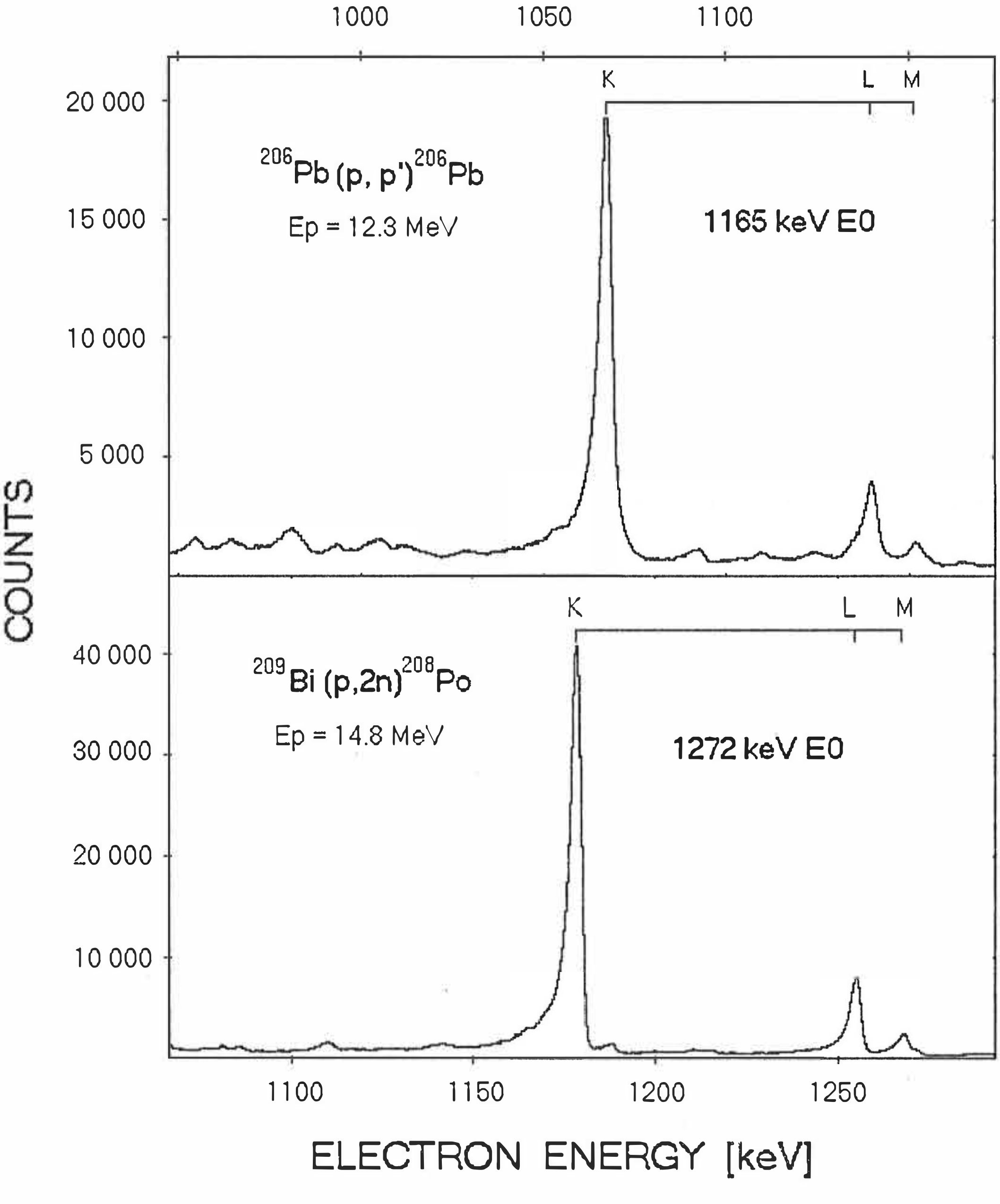


Fig. 26. *Strong E0 transitions used in experimental verification of the K/L ratios.* ***Top:*** *a 2 hours, singles conversion-electron spectrum; beam current was 25 nA, effective thickness of a tilted ^{206}Pb target was 5 mg/cm^2.* ***Bottom:*** *a 3 hour, singles conversion-electron spectrum; beam current was 20 nA, effective thickness of a tilted ^{209}Bi target was 2 mg/cm^2. Such an exceptionally high peak to background ratio is seldom encountered in in-beam measurements.*

element, the total E0 transition probability is expressed as follows [CH59]:

$$W(E0) = \rho^2(0_i \longrightarrow 0_f) \sum_j \Omega_j \ ,$$

where the summation index j refers to the atomic sub-shells K, L1, L2,.., and, if there is enough energy, to the internal pair formation. The quantities Ω [KA88, BE70, HA69] are independent of the nuclear structure and represent the electron densities in the nucleus. Although there are no reasons to doubt the validity of the above equation there are no experiments to test it either. It is not clear how good the separation of the E0-conversion probability into electronic and nuclear factors is. Actually, it is known that this separation is not as good as for the conversion of higher multipoles [CH59]. Also, the electronic factor Ω is not completely independent of nuclear properties.

One way of verifying the theory is by comparing the intensity ratios of conversion probabilities into different atomic sub-shells, typically the K/L ratio (the most intense lines) . Due to the mentioned separation of the nuclear and electronic factors in the formula for the transition probability, the dependence on nuclear matrix element ρ cancels out, leaving the ratio of the electronic factors Ω:

$$W_K \ / \ W_L = \Omega_K \ / \ \Omega_L.$$

In fact, there are three subshells labeled L: L_1, L_2 and L_3. The L_3(E0) conversion is always negligible [CH59] since $L_3/L_2 < 10^{-6}$. The L_1/L_2 ratio ranges in heavy elements between 10 and 100 (decreases with increasing Z and increasing transition energy). The K/L ratio, or more precisely the $\Omega_K \ / \ (\Omega_{L1} + \Omega_{L2})$, changes little with energy but decreases with Z. Typical value for an E0 transitions in Pb is around 6.

The present systematic study of the E0 transitions in the Pb isotopes gave, as a byproduct, a good opportunity to measure the K/L ratios. As discussed before, in the high-Z nuclei the intensity difference between K and L peaks is the smallest and therefore the easiest to compare, and there is enough energy separation between the L and the higher lines (M,N,..) to resolve them fully with a Si(Li) detector (fig. 26). Also, even if the individual measurements

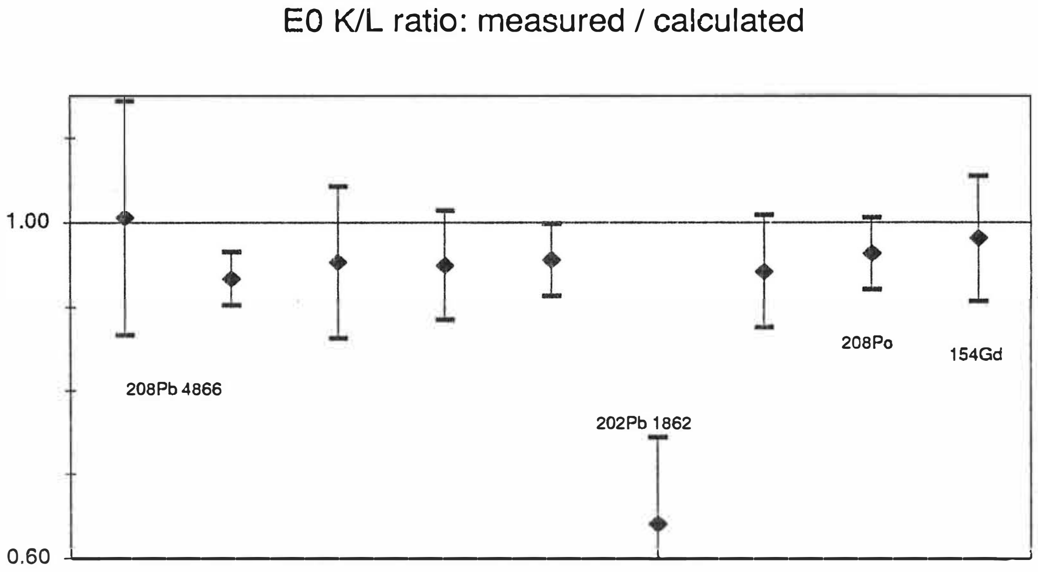


Fig. 27. *Ratio of the measured K/L ratio to the calculated value of* $\Omega_K/(\Omega_{L1} + \Omega_{L2})$ *for some strong E0 transitions in lead, polonium and gadolinium. The numerical values are summarized in tab. 4. The experimental value for* ^{154}*Gd comes from an other work (see text). With the exception of the K 1862 keV E0 transition in* ^{202}*Pb (discussed in the text) the experimental points seem to indicate a 5% deficiency over the calculated values (95% +- 2). At this stage it is difficult to say whether the deficiency should be blamed on the calculations or on a systematic error in the data.*

Table 4. *Evaluation of the K/L ratios of some strong E0 transitions. Figure 27 shows a graphic representation of this data.*

reaction	nucleus	trans. en.	calc.	measured	meas./calc.
		[keV]	K/L	K/L	[%]
(d,p)	^{208}Pb	4866	6.36	6.39(88)	100(14)
(p,p')	^{208}Pb	4866	6.36	5.94(20)	93(3)
(p,p')	^{208}Pb	5292	6.4	6.09(57)	95(9)
(p,p')	^{206}Pb	1165	5.91	5.61(38)	95(6)
(p,p')	^{204}Pb	1582	5.99	5.72(25)	96(4)
(p,2n)	^{202}Pb	1862	6.03	3.86(62)	64(10)
(p,2n)	^{202}Pb	2159	6.07	5.72(40)	94(7)
(p,2n)	^{208}Po	1272	5.81	5.59(24)	96(4)
decay	^{154}Gd	681	7.15	7.01(53)	98(7)

can not give a reliable answer, a compilation of several of them can.

Figure 27 shows the ratio of the measured K/L to the calculated value of $\Omega_K/(\Omega_{L1} + \Omega_{L2})$ for the strong E0 transitions in ^{202,204,206,208}Pb, ^{208}Po and ^{154}Gd. The data used in the plot is summarized in table 4. The experimental value for ^{154}Gd was taken from the work by M.Sakai et al. [SA87], but it was compared with a different calculated value since the original ^{154}Gd paper contained an error [KAPC]. The calculated ratios were interpolated using the tabulation by D. A. Bell et al. [BE70]. For transition energies higher than 2.5 MeV, a polynomial extrapolation was used [KA88b]. Since the electronic factors change in a very regular way, the estimated error of the extrapolation is less than 1%. According to the ref. [BE70] the original values were calculated to 0.5 % accuracy.

The measured ratios seem to be, on the average, 5% smaller than the calculated ones. However, such a difference (if real) is not too obvious to interpret. The error bars indicate only the fitting and statistical errors. Even then a 5% effect is below the sensitivity limit of each individual measurement. Typically, the L component is almost an order of magnitude less intense than the K component. A weak K-line overlapping with the E0 L-line can easily disturb the measured ratio by as much as 30%, which is a probable reason for a strong deviation for the 1862 keV E0 in ^{202}Pb. Therefore, the general trend of getting lower K/L ratios from experiments could be explained by the presence of many weak, unidentified transitions overlapping with the measured lines - a plausible explanation in the case of in-beam spectra. Our tests with radioactive sources (fig. 28) do not indicate any possible systematic errors in the analysis.

As a safe conclusion one can only say that if there is a difference between the experimental and the calculated K/L values it is probably of the order of 5% but definitely not larger than 7%. It would be interesting to extend the K/L systematics over the whole periodic table. In many cases the experimental data already exist but can not be readily compared with the calculations

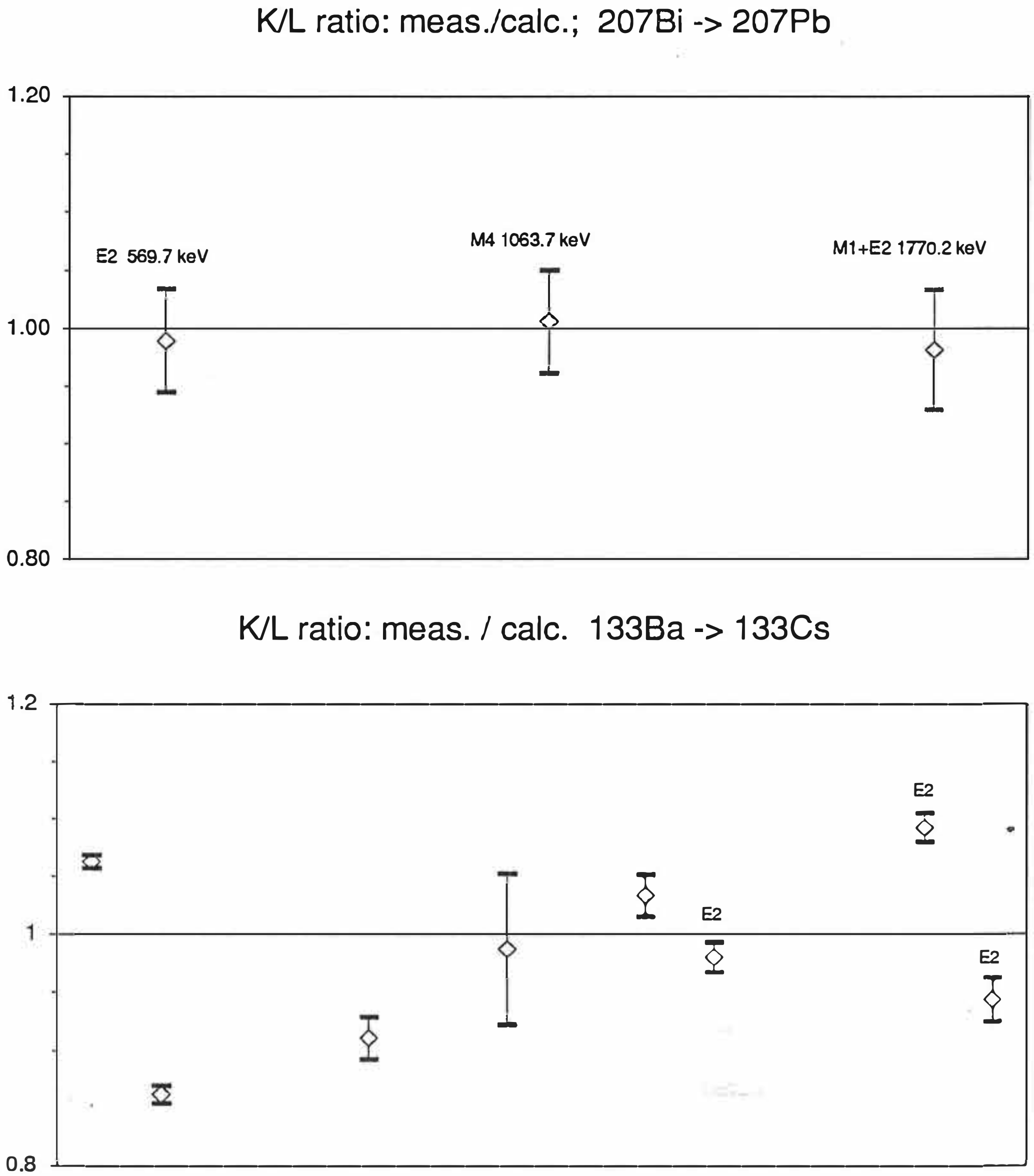


Fig. 28. Ratio of the measured K/L ratios to the calculated values of α_K/α_L evaluated for typical calibration sources. Transitions following the decay of ^{207}Bi (top) produce results consistent with the tabulated conversion coefficients (mean value: 99% +- 3). There is no indication of a systematic error in the evaluation of the data that would help to explain the deficiency from fig. 26. Data from the decay of ^{133}Ba (bottom) differs widely from the expected ratio of 1 (there are mostly low energy transitions - very sensitive to the assumed mixing ratios). However, again, there is no indication of a systematic experimental underestimation of the K/L ratio.

since the tabulation covers only the K, L1 and L2 shells whereas the L peak is not usually resolved from the M + N +.. peaks. Therefore, extended calculations are called for to cover the whole set of atomic shells and transition energies up to 5 - 6 MeV.

4 Summary and Conclusions

As the result of our systematic study of the even-even lead isotopes between mass numbers 202 and 208, four new 0^+ states were identified and several previously undetected E0 transitions were observed improving considerably the energy determination of the known states. We were also able to determine the half-life of the 1582 keV 0^+_2 level in ^{204}Pb and set the limits for four other states in ^{202}Pb and ^{204}Pb. In the case of ^{202}Pb the competing E2 branches were identified yielding the X values. Limits were set for the X values in ^{204}Pb. Using the newly developed spectrometer [JU88] first successful conversion-electron measurements in ^{208}Pb at electron energies around 6 MeV were made. The large base for the intruder states systematics enabled tentative assignments to the two-particle, two-whole proton intruder configuration. In addition, a comprehensive study of the K/L ratios of the E0 transitions was carried out.

In the course of this work, major effort was placed on improving the accuracy and scnsitivity of convcrsion-electron measurements. Λ new evaluation of the possible calibration sources was made together with electron intensity calculations based on the latest spectroscopic data. Precise efficiency determination of the combination (magnetic lens + semiconductor detector) spectrometer has been determined bringing new, unanswered questions about the interaction of electrons with silicon. Energy loss and electron line broadening after passing through an absorber were investigated to check the validity of semi-empirical formulae [CH51] useful in the analysis of the in-beam spectra. Several background reducing techniques were introduced. Area extraction from the experimental spectra was thoroughly reviewed resulting in the creation of a new computer code better suited to the actual line-shape of the electron peaks. Extensive numerical calculations were carried out to simulate the electron trajectories in the magnetic field in order to optimize the performance of the spectrometer. A new, convenient way of expressing transmission characteristics of magnetic spectrometers was suggested.

Many simple, intriguing questions concerning the 0^+ states in lead isotopes remain unanswered: the location of the proton 2p-2h state in ^{208}Pb and ^{204}Pb, E0 transitions from the states in ^{206}Pb, 0^+ states in the isotopes above A=208 and below A=192, the K/L ratios. Clearly, more experiments are needed. It would have been very interesting to employ a two-nucleon transfer reaction - for example a (^{3}He,n) on Hg isotopes - a well known favorite in population of the 0^+ states. Unfortunately, no good quality Hg targets (suitable for electron spectroscopy) were available. Another useful reaction would be (p,t) on the stable Pb isotopes. Unfortunately, this reaction channel is much weaker than the competing (p,p'), so it would require an efficient coincidence set-up with a good selectivity. We do not have such a system at present. Among the very few new experimental attempts that can be tried in the near future is the repeated search for the E0 transitions in ^{208}Pb and ^{206}Pb using the new spectrometer with the beam dump buried deep in the wall. If we succeed in maintaining the overall background reduction by a factor of five, as the preliminary tests indicate, we may stand a chance of seeing some of the missing E0 transitions.

The theoretical descriptions of the 0^+ states in the even-even $^{202\text{-}208}$Pb range from fair to good. Obviously, the doubly magic ^{208}Pb holds the focus of attention. Since the publication of our high quality in-beam data [JU87] a follow-up, revised calculations were made [CR88] ending, it seams, 20 years of search for the two-octupole-phonon state by identifying it with the observed 5237 keV level. However, this case makes one wonder about the reliability of any model interpretation including the present one: since its discovery in a nuclear transfer study [IG71] the 0^+_3 has changed its "accepted" character five times.

Our data has triggered interesting changes in the understanding of the level scheme of ^{204}Pb nucleus as well. The present, tentative interpretation links all the observed levels (0^+_1 through 0^+_4) with the neutron valence space. Therefore, there are no clear candidates for the expected intruder state. The new systematics [HE87] suggests such a state at around 3.2 MeV. Clearly improved shell-model calculations are called for to verify the present interpretation. Also in ^{202}Pb theoretical calculations lag behind the experimental evidence predicting only two low-lying, excited 0^+ states as opposed to the three observed 0^+ states. Our tentative assignment of the 0^+_4 state as the intruder state remains unchallenged and, at the same time, unconfirmed.

Only in ^{206}Pb the understanding of the 0^+ states seems good and stable. The general picture of four 0^+ states generated by two neutron holes has not changed significantly since the early calculations by True [TR68]. Also the observation of the proton-pairing vibration remains undisputed. Unfortunately, our attcmpts to mcasurc thc E0 dccay of thcsc states and their properties have failed preventing us from possibly changing that picture.

References:

[AN77] R. E. Anderson, P. A. Baty-Csorba, R. A. Emigh, E. R. Flynn, D. A. Lind, P. A. Smith, C. D. Zafiratos, and R. M. DeVries, Phys. Rev. Lett. **39**, 987 (1977).

[BA76] I. M. Band, M. B. Trzhaskovskaya, and M. Listengarten, At. Data Nucl. Data Tables **18**, 433 (1976).

[BA78] H. Backe, L. Richter, R. Willwater, E. Kankeleit, E. Kuphal, Y. Nakayama, and B. Martin, Z. Phys. A **285**, 159 (1978).

[BE69] M. J. Berger, S. M. Seltzer, S. E. Chappell, J. C. Humphreys, and J. W. Motz, Nucl. Instr. and Meth. **69**, 181 (1969).

[BE70] D. A. Bell, C. E. Aveledo, M. G. Davidson, and J. P. Davidson, Can. J. Phys. **48**, 2542 (1970).

[BJ66] J. H. Bjerregaard, O. Hansen, O. Nathan, and S. Hinds, Nucl. Phys. **89**, 337 (1966).

[BL70] J. Blomqvist, Phys. Lett. **33B**, 541 (1970).

[BLPC] J. Blomqvist, private communication.

[BO75] A. Bohr, B. R. Mottelson, “Nuclear Structure”, Vol.2, W. A. Benjamin Inc. (1975).

[BR67] R. A. Broglia, C. Riedel, Nucl. Phys. **A92**, 145 (1967).

[CA71] David C. Camp, Gerald L. Meredith, Nucl. Phys. **A166**, 349 (1971).

[CH51] J. J. Chen, and S. D. Warshaw, Phys. Rev. **84**, 355 (1951).

[CH59] E. L. Church and J. Weneser, Phys. Rev. **103**, 1035 (1959).

[CO71] B. L. Cohen, Concepts of Nuclear Physics, McGraw Hill (1971).

[CO85] G. G. Colvin, K. Schreckenbach, Nucl. Instr. and Meth. **228**, 365 (1985).

[CR88] P. Crutchet, J. Blomqvist, R. J. Liotta, G. G. Dussel, C. Pomar, and S. L. Reich, Phys. Lett. **208B**, 331 (1988).

[DE79] J. Deslauriers, S. K. Mark, Nucl. Instr. and Meth. **159**, 243 (1979).

[DU84] P. Van Duppen, E. Coenen, K. Deneffe, M. Huyse, K. Heyde, and P. Van Isacker, Phys. Rev. Lett. **52**, 1974 (1984).

[DU85] P. Van Duppen, E. Coenen, K. Deneffe, M. Huyse, and J. L. Wood, Phys. Lett. **154B**, 354 (1985).

[EL78] C. Ellegaard, R. Julin, J. Kantele, M. Luontama, and T. Poikolainen, Nucl. Phys. **A302**, 125 (1978).

[FL67] E. R. Flynn, G. J. Igo, R. Woods, P. D. Barnes, and Norman K. Glendenning, Phys. Rev. Lett. **19**, 798 (1967).

[FL74] E. R. Flynn, R. A. Broglia, R. Liotta, B. S. Nilsson, Nucl. Phys. **A221**, 509 (1974).

[GO70] L. H. Goldman, B. L. Cohen, R. A. Moyer, and R. C. Diehl, Phys. Rev. C **1** 1781 (1970).

[GR75] Graham J. McCallum, Graeme E. Coote, Nucl. Instr. and Meth. **124**, 309 (1975).

[GU84] M. Guttormsen, H. Hübel, A. v. Grumbkow, Y. K. Agarwal, J. Recht, K. H. Maier, H. Kluge, A. Maj, M. Menningen, and N. Roy, Nucl. Instr. and Meth. **227**, 489 (1984).

[HA69] R. S. Hager, and E. C. Seltzer, Nucl. Data Tables **A6**,1 (1969).

[HA71] T. F. Harvey and D. M. Clement, Nucl. Phys. **A176**, 592 (1971).

[HA88] J.M.Hanly, S.E.Hicks, M.T.McEllistrem, S.W.Yates Phys. Rev. C **37**, 1840 (1988).

[HE87] K. Heyde, J. Jolie, J. Moreau, J. Ryckebush, M. Waroquier, P. Van Duppen, M. Huyse, and J. L. Wood, Nucl. Phys. **A466**, 189 (1987).

[IG71] G. Igo, P. D. Barnes, and E. R. Flynn, Ann. Phys. **66**, 60 (1971).

[JA74] L. J. Jardine, C. M. Lederer, Nucl. Instr. and Meth. **120**, 515 (1974).

[JE49] Jesse W. M. DuMond, The Rev. of Scien. Instr. **20**, 160 (1949).

[JO77] H. H. Jorch, J. L. Campbell, Nucl. Instr. and Meth. **143**, 551 (1977). See also J. L. Campbell and H. H. Jorch, Nucl. Instr. and Meth. **159**, 163 (1979).

[JO81] B. Jonson, O. B. Nielsen, L. Westgaard, and J. Żylicz, in Proceedings of the 4th International Conference on Nuclei far from Stability, CERN Report 81-09,1981, p. 640.

[JU76] R. Julin, J. Kantele, M. Luontama, T. Poikolainen, and V. Rahkonen, Phys. Lett. **65B**, 337 (1976).

[JU87] R. Julin, J. Kantele, J. Kumpulainen, M. Luontama, A. Passoja, W. Trzaska, E. Verho, and J. Blomqvist, Phys. Rev. C **36**, 1129 (1987).

[JU88] R. Julin, J. Kantele, J. Kumpulainen, M. Luontama, V. Nieminen, A. Passoja, W. Trzaska, and E. Verho, Nucl. Instr. and Meth. **A270**, 74 (1988).

[KA75] J. Kantele, M. Luontama, A. Passoja and R. Julin, Nucl. Instr. and Meth. **130**, 467 (1975).

[KA82a] J. Kalef-Ezra, Y. S. Horowitz, and J. M. Mack, Nucl. Instr. and Meth. **195**, 587 (1982).

[KA82b] J. Kantele, R. Julin, M. Luontama, and A. Passoja, Nucl. Instr. and Meth. **200**, 253 (1982).

[KA83] J. Kantele, M. Luontama, W. Trzaska, A. Passoja, Nucl. Instr. and Meth. **206**, 403 (1983).

[KA84] J. Kantele in "Heavy Ions and Nuclear Structure" Proceedings of the XIV Summer School, Mikolajki 1984, edited by B. Sikora and Z. Wilhelmi (Harwood, Academic, New York 1984).

[KA86] J. Kantele, M. Luontama, W. Trzaska, R. Julin, A. Passoja, and K. Heyde, Phys. Lett. **171B**, 151 (1986).

[KA88a] J. Kantele, J. Kumpulainen, V. Nieminen, J. Pesonen, W. H. Trzaska, and R. Julin, JYFL Annual Report 1988.

[KA88b] J. Kantele, Nucl. Instr. and Meth. **A271**, 625 (1988).

[KA89] J. Kantele, Nucl. Instr. and Meth. **A275**, 149 (1989).

[KAPC] J. Kantele, private communication.

[KL65] P. Kleinheinz, L. Samuelsson, R. Vukanovic, and K. Siegbahn, Nucl. Instr. and Meth. **32**, 1 (1965).

[KO73] C. M. Ko, T. T. S. Kuo, and J. B. McGrory, Phys. Rev. C **8**, 2379 (1973).

[KU75] K. Kumar,in "The Electromagnetic Interaction in Nuclear Spectroscopy", edited by W. D. Hamilton, North-Holland (1975).

[LA74] W. A. Lanford, G. M. Crawley, Phys. Rev. C **9**, 646 (1974).

[LA77] W. A. Lanford, Phys. Rev. C **16**, 988 (1977).

[LA82] J. Lange, Krishna Kumar, J. H. Hamilton, E0-E2-M1 multipole admixtures of transitions in even-even nuclei, Rev. Mod. Phys. **54**, 119 (1982).

[LE68] G.H.Lenz and G.M.Temmer, Nucl. Phys. **A112**, 625 (1968).

[LI75] Th. Lindblad, C. G. Linden, Nucl. Instr. and Meth. **126**, 397 (1975).

[LI81] R. J. Liotta and C. Pomar, Nucl. Phys. **A362**, 137 (1981).

[MA78] W. Mampe, K. Schreckenbach, P. Jeuch, B. Maier, F. Braumandl, J. Larysz, and T. von Egidy, Nucl. Instr. and Meth. **159**, 243 (1978).

[MA83a] K. H. Maier, T. Nail, R. K. Sheline, W. Stöffl, J. A. Becker, J. B. Carlson, R. G. Lainer, L. G. Mann, G. L. Sruble, J. A. Cizewski, and B. H. Erkkila, Phys. Rev. C **27**, 1431 (1983).

[MA83b] M. A. J. Mariscotti, D. R. Bes, S. L. Reich, H. M. Sofia, P. Hungerford, S. A. Kerr, K. Schreckenbach, D. D. Warner, W. F. Davidson, and W. Gelletly, Nucl. Phys. **A407**, 98 (1983).

[MA86] M. J. Martin, Nucl. Data Sheets **47**, 797 (1986).

[MG75] J. B. McGrory and T. T. S. Kuo, Nucl. Phys. **A247**, 283 (1975).

[PE49] E. Persico, The Rev. of Scien. Instr. **20**, 191 (1949).

[PE74] S. M. Peltier, A. Plastino, and E. S. Hernandez, Z. Phys. **269**, 347 (1974).

[RA60] J. O. Rasmussen, Nucl. Phys. **19**, 85 (1960).

[RE61] A. S. Reiner, Nucl. Phys. **27**, 115 (1961).

[RO78] F. Rösel, H. M. Fries, K. Alder, and H. C. Pauli, At. Data Nucl. Data Tables **21**, 91 (1978).

[SL49] H. Slätis and K. Siegbahn, Ark. Fys. **1**, 339 (1949).

[SA87] M. Sakai, Y. Fujita, M. Imamura, K. Omata, K. Miyatake, T. Nomura, S. Ohya, and S. Chojnacki, Nucl. Phys. **A473**, 317 (1987).

[SC78] M. R. Schmorak, Nucl. Data Sheets **25**, 675 (1978).

[SC79] M. R. Schmorak, Nucl. Data Sheets **27**, 581 (1979).

[SC84] J. Schirmer, D. Habs, R. Kroth, N. Kwong, D. Schwalm, M. Zirnbauer, and C. Broude, Phys. Rev. Lett. **53**, 1897 (1984).

[SC87] M. R. Schmorak, Nucl. Data Sheets **50**, 669 (1987).

[SI65] K. Siegbahn,"Alpha, Betha and Gamma-ray Spectroscopy" Vol.I, North Holland (1965).

[SZPC] F. Szelecsenyi, F. Tarkanyi, Z. Kovacs, private communication and to be published.

[MA68] J. B. Marion, F. C. Young, "NUCLEAR REACTION ANALYSIS", North-Holland (1968).

[SM70] S. M. Smith, P. G. Ross, A. M. Bernstein, and Cyrus Moazed, Nucl. Phys. **A158**, 497 (1970).

[SP74] J. Speth, L. Zamick, and P. Ring, Nucl. Phys. **A232**, 1 (1974).

[ST84] W. Stöffl, E. A. Henry, Nucl. Instr. and Meth. **227**, 77 (1984).

[TA71] T. Tabata, R. Ito, and S. Okabe, Nucl. Instr. and Meth. **94**, 509 (1971).

[TA72] J. W. Tape, E. G. Adelberger, D. Burch, and Larry Zamick, Phys. Rev. Lett. **29**, 878 (1972).

[TA83] M. Takahashi, T. Murakami, S. Morita, H. Orihara, Y. Ishizaki, and H. Yamaguchi, Phys. Rev. C **27**, 1454 (1983).

[TR68] William W. True, Phys. Rev. **168**, 1388 (1968).

[TR72] V. F. Trusov, Nucl. Data Tables **10**, 477 (1972).

[VE81] J. Verplancke, B. Brijs, K. Cornelis, J. Gentens, M. Huyse, and G. Lhersonneau, Nucl. Instr. and Meth. **186**, 1977 (1981).

[VO86] N. A. Voinova-Eliseeva and I. A. Mitropolskij, Sov. J. Part. Nucl. **17**, 521 (1986).

[WA85] A. H. Wapstra, Nucl. Phys. **A432**, 1 (1985).

[WE79] M. P. Webb, Nucl. Data Sheets **26**, 145 (1979).

[WE81] R. C. Weiss, R. E. Anderson, J. J. Kraushaar, R. A. Ristinen, E. Rost, and S. Shastry, Nucl. Phys. **A355**, 45 (1981).

[WE84] H. I. West, Jr., Nucl. Instr. and Meth. **223**, 85 (1984).

[WOPC] J. Wood, J. Blomqvist, private communication.

[YA87] S. W. Yates, L. G. Mann, E. A. Henry, D. J. Decman, R. A. Meyer, R. J. Estep, R. Julin, A. Passoja, J. Kantele, W. Trzaska, Phys. Rev. C **36**, 2143 (1987).

APPENDIX A

Useful formulae for analysis of electron spectra.

Most probable[a] energy loss (ΔE) and line broadning[b] (δ) of monoenergetic electrons passing through an absorber

$$\boxed{\Delta E\,[\mathrm{kev}] \;=\; \frac{t\cdot x}{\beta^{2}}\cdot \ln\left[\frac{2\cdot t\cdot x}{\left[1-\beta^{2}\right]\cdot I^{2}\cdot \exp\left[\beta^{2}-0.37\right]}\right]\cdot 511}$$

$$\boxed{\delta\,[\mathrm{kev}] \;=\; 1.22\cdot t\cdot \frac{Z}{A}\cdot \beta^{-2}}$$

$$\beta \;=\; \sqrt{1-\frac{511\cdot 511}{(511+E)^{2}}} \qquad x \;=\; \frac{Z}{A}\cdot 300\cdot 10^{-6}$$

$$I \;=\; (1.75\cdot Z+3.1)\cdot 10^{-5}$$

E - electron energy [keV]

Z - atomic number of the absorber

A - mass number of the absorber

t - absorber thickness [mq/cm^2]

a. ΔE should not be confused with the mean energy loss. In practical terms ΔE corresponds to the peak shift measured at the maximum position and is smaller than the mean energy loss.

b. To estimate the finnal resolution δ should be added quadraticly to the resolution measured without the absorber.

Relationship between energy (E) and trajectory radius (r) for electrons moving in a magnetic field:

since

$$r = \frac{p \cdot \sin(\theta)}{q \cdot B}$$

and

$$p[\mathrm{MeV}] = \sqrt{E^2 + 1.022 \cdot E}$$

$$(q = e)$$

$$r = \frac{\sin(\theta)}{300 \cdot B} \cdot \sqrt{E^2 + 1.022 \cdot E}$$

$$E = \sqrt{0.511^2 + \left[\frac{300 \cdot B \cdot r}{\sin(\theta)}\right]^2} - 0.511$$

r - trajectory radius [m]
E - electron energy [MeV]
B - magnetic induction [T]
θ - angle between $\vec{p}$ and $\vec{B}$
p - electron momentum [MeV]

Charged ions in the magnetic field:

Relativistic formula:

$$r = \frac{\sin(\theta)}{300 \cdot B \cdot q} \cdot \sqrt{E^2 + 1864 \cdot A \cdot E}$$

Nonrelativistic:

$$r = \frac{\sin(\theta)}{300 \cdot B \cdot q} \cdot \sqrt{1864 \cdot A \cdot E}$$

A - mass number of the ion
q - charge of the ion [units of e]

APPENDIX B

Energies (in keV) and relative intensities of gamma-rays and conversion-electrons from the decay of selected radioactive sources. For convenience, in case of ^{207}Bi and ^{133}Ba the given intensities equal number of gamma-rays and conversion electrons emitted per second form a 1 micro curie source (37 000 decays per second).

^{207}**Bi** $T_{1/2}$ = 31.8(19) y

E_{gamma}	I_{gamma}	+-	E_{el}	Shell/Mult.	I_{calc}	I_{expe}	+-
569.702	36168	145	481.697	K / E2	578	564	11
			553.841	L1	85		
			554.502	L2	57		
			556.667	L3	20		
			554.4	L	162	161	5.4
1063.662	27415	145	975.657	K/M4(+E5)	2680	2590	61
			1047.801	L1	528		
			1048.462	L2	104		
			1050.627	L3	48		
			1048.1	L	680	676	29
1770.237	2541	10	1682.232	K/M1+E2	9.33	8.39	0.51
			1754.376	L1	1.39		
			1755.037	L2	0.09		
			1757.202	L3	0.01		
			1754.4	L	1.49	1.53	0.17

^{152}Eu $T_{1/2}$ = 13.51(3) y

E_{gamma}	I_{gamma}[a]	E_{el}	Isotope/Shell	I_{expe}[b]	+-
121.782	1362(16)	74.984	Sm/K	1234	75
		114.046	Sm/L1	116	7
		114.471	Sm/L2	289	17
		115.066	Sm/L3	285	17
		114.6	Sm/L	690	41
244.699	359(6)	197.865	Sm/K	38.9	1.3
		236.962	Sm/L1	4.39	0.15
		237.387	Sm/L2	3.43	0.12
		233.983	Sm/L3	2.69	0.10
		237.4	Sm/L	10.5	0.3
344.276	1279(6)	294.037	Gd/K	55.1	1.9
		335.900	Gd/L1	6.67	0.23
		336.345	Gd/L2	3.61	0.13
		337.033	Gd/L3	2.46	0.09
		336.2	Gd/L	12.7	0.4
411.115	109.0(5)	360.876	Gd/K	3.0	0.1
443.965	150.6(6)	397.131	Sm/K	1.3	0.05
586.294	21.9(8)	536.055	Gd/K	0.69	0.03
615.406	E0	565.167	Gd/K	0.59	0.03
656.484	7.1(5)	609.649	Sm/K	0.53	0.03
688.678	42.0(4)	641.843	Sm/K	2.17	0.09
778.903	621.6(22)	728.664	Gd/K	1.49	0.04
964.055	701.4(23)	917.221	Sm/K	2.48	0.06
1085.842	481.5(16)	1039.008	Sm/K	1.35	0.05
1089.700	83.5(4)	1039.461	Gd/K	0.25	0.01
1112.087	646.7(21)	1065.253	Sm/K	1.72	0.04
1408.022	1000(3)	1361.188	Sm/K	0.75	0.02

----

a. for absolute intensity per 37 000 decays multiply by 7.715

b. for absolute intensity per 37 000 decays multiply by 5.76

^{133}Ba $T_{1/2}$ = 10.57(4) y

E_{gamma}	I_{gamma}	+-	E_{el}	Shell	I_{calc}	I_{expe}	+-
53.161	814	8	17.176	K	4009	4068	814
			47.447	L1	489		
			47.802	L2	101		
			48.149	L3	90		
			47.5	L	680	473	95
79.623	969	22	43.638	K	1479	1338	107
			73.909	L1	181		
			74.264	L2	22		
			74.611	L3	13		
			74.0	L	216	217	17
80.997	12602	101	45.012	K	18378	16700	530
			75.283	L1	2239		
			75.638	L2	314		
			75.985	L3	217		
			75.3	L	2770	2720	86
160.613	239	3	124.628	K	56.44	52	1
			154.899	L1	6.18		
			155.254	L2	2.73		
			155.601	L3	2.55		
			155.1	L	11.46	10.7	0.3
223.234	166.5	1.5	187.249	K	14.20	13	1
			217.520	L1	1.73		
			217.875	L2	0.12		
			218.222	L3	0.03		
			217.6	L	1.88	1.7	0.1
276.398	2651	8	240.413	K	122.29	123	8
			270.684	L1	13.13		
			271.039	L2	5.21		
			271.386	L3	4.33		
			270.9	L	22.68	22.8	1.5
302.853	6782	20	266.868	K	258.22	251	14
			297.139	L1	31.46		
			297.494	L2	1.75		
			297.841	L3	0.41		
			297.2	L	33.62	36.4	2
356.017	24069	72	320.032	K	507.59	484	14
			350.303	L1	56.08		
			350.658	L2	16.11		
			351.005	L3	12.28		
			350.5	L	84.47	79.4	2.3
383.851	3308	10	347.866	K	55.71	57	3
			378.137	L1	6.20		
			378.492	L2	1.62		
			378.839	L3	1.20		
			378.3	L	9.02	9.3	0.5

^{66}Ga $T_{1/2}$ = 9.49(8) h

E_{gamma}	+-	I^a	+-	$I_{gamma}{}^{bc}$	$E_K{}^d$	mult.	$I_K{}^{ef}$	$I_{TOT}{}^{gf}$
833.65	0.08	15950	160	15950	824.0	E2(+M1)	24611	27483
1039.35	0.08	100000		100000	1029.7	E2	89550	100000
1333.37	0.09	3260	30	3260	1323.7	E2	1677	1873
1418.97	0.09	1680	20	1680	1409.3	?		
1918.66	0.09	5650	20	5650	1909.0	?		
2190.2	0.15	15050	150	15121	2180.5	?		
2422.5	0.15	5140	50	5213	2412.8	?		
2752.27	0.1	61100	500	63087	2742.6	M1(+E2)	8185	9140
3229.35	0.2	3920	30	4191	3219.7	M1/E1	419/284	468/317
3381.3	0.2	3730	30	4040	3371.6	M1	376	420
3422.5	0.2	2170	40	2359	3412.8	?		
3791.56	0.1	2670	30	3010	3781.9	M1	234	262
4086.45	0.15	3020	40	3516	4076.8	M1/E1	244/176	273/197
4295.5	0.2	9180	100	10955	4285.8	M1	707	789
4462.1	0.14	1870	20	2278	4452.4	M1	139	155
4806.6	0.2	3860	40	4918	4796.9	M1	270	301

----

a. original data; contains systematic error

b. corrected gamma-ray intensity

c. for absolute intensity per 37 000 decays multiply by 0.14

d. K-line electron energy [keV]

e. K-line electron intensity (calculated)

f. for absolute intensity per 37 000 decays multiply by $3.8*10^{-5}$

g. calculated total conversion-electron intensity

E_{gamma}	I_{gamma}[a]	+-	E_K	I_K(calc)[b]	I_K(expe)[b]	+-
733.516	0.193	0.012	726.402	0.499	0.49	0.1
787.742	0.305	0.013	780.628	0.82	0.85	0.08
846.769	100	0.3	839.655	260	260	0.6
977.368	1.435	0.016	970.254	2.07	1.96	0.06
1037.842	14.16	0.05	1030.728	18.09	18.77	0.39
1175.097	2.241	0.012	1167.983	2.3	2.16	0.04
1238.286	66.06	0.21	1231.172	69.3	69.3	0.9
1360.206	4.265	0.017	1353.092	3.24	3.3	0.7
1771.344	15.49	0.05	1764.230	7.26	7.32	0.21
1963.714	0.707	0.011	1956.600	0.277	0.28	0.06
2015.19	3.026	0.014	2008.076	1.16	1.24	0.06
2034.769	7.766	0.028	2027.655	2.86	3.03	0.0[illegible]
2598.459	16.96	0.06	2591.345	4.14	4.52	0.1
3009.587	1	0.01	3002.473	0.192	0.22	0.06
3201.953	3.04	0.03	3194.839	0.532	0.633	0.031
3253.428	7.41	0.07	3246.314	1.29	1.48	0.06
3273.006	1.75	0.02	3265.892	0.296	0.347	0.031
3451.148	0.875	0.01	3444.034	0.139	0.102	0.018

a. for absolute intensity per 37 000 decays multiply by 370

b. for absolute intensity per 37 000 decays multiply by 0.038

The following sources were used for compilation of the data presented in the appendix B:

^{207}Bi

- M. R. Schmorak, Nucl. Data Sheets **43**, 383 (1984)
- Zs. Nemeth, Nucl. Instr. and Meth. **A267**, 153 (1988)
- Y. Yoshizawa et al., Nucl. Instr. and Meth. **174**, 109 (1980)
- L. J. Jardine and C. M. Lederer, Nucl. Instr. and Meth. **120**, 515 (1974)

^{152}Eu

- G. C. Colvin and K. Schreckenbach, Nucl. Instr. and Meth. **228**, 365 (1985)
- W. H. Trzaska, conversion-electron measurements, unpublished
- J. Deslauriers and S. K. Mark, Nucl. Instr. and Meth. **159**, 243 (1979)
- E. K. Warburton and D. E. Alburger, Nucl. Instr. and Meth. **A235**, 38 (1986)
- C. M. Baglin, Nucl. Data Sheets **30**, 1 (1980)
- Y. Yoshizawa et al., Nucl. Instr. and Meth. **174**, 133 (1980)

^{133}Ba

- Yu. V. Sergeenkov and V. M. Sigalow, Nucl. Data Sheets **49**, 639 (1986)
- Y. Yoshizawa et al., Nucl. Instr. and Meth. **212**, 249 (1983)
- K. S. Krane, At. Data Nucl. Data Tables **19**, 363 (1977)

^{66}Ga

- N. J. Ward and F. Kearns, Nucl. Data Sheets **39**, 1 (1983)
- D. C. Camp and G. L. Meredith, Nucl. Phys. **A166**, 349 (1971)
- G. J. McCallum and G. E. Coote, Nucl. Instr. and Meth. **124**, 309 (1975)
- M. E. Phelps, D. G. Sarantities and W. G. Winn, Nucl. Phys. **A149**, 647 (1970)
- A. Schwarzschild and L. Grodzins, Phys. Rev. **119** 276 (1960)

^{56}Co

- G. Wang, E. K. Warburton, D. E. Alburger, Nucl. Instr. and Meth. **A272**, 791 (1988)
- H. Junde et al., Nucl. Data Sheets **51**, 1 (1987)
- S. Ohya et al., J. of the Phys. Soc. of Japan **53**, 538 (1984)
- N. M. Stewart and A. M. Shaban, Z. Physic A **296**, 165 (1980)
- R. G. Helmer et al., At. Data Nucl. Data Tables **24**, 39 (1979)
- G. J. McCallum and G. E. Coote, Nucl. Instr. and Meth. **124**, 309 (1975)
- H. Petterson et al., Arkiv för Fysik **29**, 423 (1965)

The half-life information (except ^{66}Ga) was taken from:

- K. Debertin and R. G. Helmer, GAMMA- AND X-RAY SPECTROMETRY WITH SEMICONDUCTOR DETECTORS, North-Holland 1988

In addition, the following tabulations of calculated conversion coefficients were used:

- F. Rösel et al., At. Data Nucl. Data Tables **21**, 91 (1978)
- I. M. Band et al., At. Data Nucl. Data Tables **18**, 433 (1976)
- V. H. Trusov, Nucl. Data Tables **10**, 477 (1972)

APPENDIX C

Program Si(Li)-FIT for shape analysis of electron spectra. The listed version is dated March 1989 and is ment to be compiled using Turbo C.

```
programs\fit\my_fit01.c
programs\fit\part01.c
programs\fit\part02.c
programs\fit\part03.c
programs\fit\part04.c
programs\fit\part05.c
programs\fit\part06.c
```

```
/*----------- MY_FIT01.C --- main program ---------*/
/*--------- W. H. Trzaska        March 1989 ------*/

#include <stdio.h>
#include <math.h>
#include <graphics.h>
#define ESC 0x1b
#define UP 72
#define FAST_UP 56
#define DOWN 80
#define FAST_DOWN 50
#define LEFT 75
#define FAST_LEFT 52
#define RIGHT 77
#define FAST_RIGHT 54
#define PAN 2
#define INCREASE 2.
#define SPECTRUM_COLOR 1
#define      FIT_COLOR 3
#define LOG_COLOR 3

int X;                    /*          horizontal pixels         */
int Y;                    /*          vertical pixels                       */
int N_MAX1;               /*          highest channel number */
int LINE ;  /*             # of lines of text          */
                          /*          program leaves space on */
           /*     top and bottom          */
                          /*          = Y/LINE + 1                          */

float *pointer_to_sp, *pointer_to_spf;

main()
{

int i, chn;
extern int _zoom, _from_x, _to_x ;
double en, c ;
void readspectrum( float sp[], int *n_max);
void display_spe(float sp[],int from_x,int *to_x,int zoom,
                       float y_low, float y_high);
void display_fit( float fit[], int fit_from, int fit_to );
void display_log( float sp[]);
void initialize();
void display_menu();
void fit();
void cursor(int *c_channel, double *c_energy);
void on_off();
float ymax( float sp[] );
extern float _y_low, _y_high;

extern float  *pointer_to_sp, *pointer_to_spf;
static float sp[4100],  spf[4100];
int n_max;

                pointer_to_sp = sp;
                pointer_to_spf = spf;

  initialize();

                readspectrum(sp,&n_max);
                N_MAX1 = n_max;
                _y_high = ymax( sp );
                _y_low = 0;
                _zoom = 1;
                _from_x = 0;

                cleardevice();
                setcolor(SPECTRUM_COLOR);
                display_spe(sp,_from_x,&_to_x,_zoom,_y_low,_y_high);

start_of_the_main_loop:

                display_menu();

/*----------------main loop------------------------------*/

while( ( c = toupper ( getch() ) ) != ESC ){

       switch (c) {
                case ’N’:       /*---------new file----------*/
                                readspectrum(sp,&n_max);
                                N_MAX1 = n_max;
                                _y_high = ymax( sp );
                                _y_low = 0;
                                _from_x = 0;
                                setcolor(SPECTRUM_COLOR);
                                clearviewport();
                                _zoom = 1;
                                display_spe(sp,_from_x,&i,_zoom,_y_low,_y_high);
                                display_menu();
                                break;
                case ’+’:       /*------zoom in---------------*/
                                if ( _zoom >= 30 ) break;
                                _zoom = _zoom + 1;
                                setcolor(SPECTRUM_COLOR);
                                clearviewport();
                                display_spe(sp,_from_x,&_to_x,_zoom,_y_low,_y_high);
                                break;
                case ’-’:       /*-------zoom out--------------*/
                                if( _zoom == 1 ) break;
                                _zoom = _zoom - 1;
                                setcolor(SPECTRUM_COLOR);
                                clearviewport();
```

```
                        display_spe(sp,_from_x,&_to_x,_zoom,_y_low,_y_high);
                        break;
        case UP:
                        _y_high = _y_high / INCREASE ;
                        if( _y_high < _y_low ) _y_high = _y_low + 10. ;
                        if (_y_high < 10.) _y_high = 10.;
                        setcolor(SPECTRUM_COLOR);
                        clearviewport();
                        display_spe(sp,_from_x,&_to_x,_zoom,_y_low,_y_high);
                        break;
            case FAST_UP:
                        _y_high = _y_high / 10. ;
                        if( _y_high < _y_low ) _y_high = _y_low + 10. ;
                        if (_y_high < 10.) _y_high = 10.;
                        setcolor(SPECTRUM_COLOR);
                        clearviewport();
                        display_spe(sp,_from_x,&_to_x,_zoom,_y_low,_y_high);
                        break;
            case DOWN:
                        _y_high = _y_high * INCREASE ;
                        setcolor(SPECTRUM_COLOR);
                        clearviewport();
                        display_spe(sp,_from_x,&_to_x,_zoom,_y_low,_y_high);
                        break;
            case FAST_DOWN:
                        _y_high = _y_high * 10 ;
                        setcolor(SPECTRUM_COLOR);
                        clearviewport();
                        display_spe(sp,_from_x,&_to_x,_zoom,_y_low,_y_high);
                        break;
            case LEFT:
                        if ( _to_x >= N_MAX1 ) break;
                        i = ( _to_x - _from_x ) ;
                        if( ( _to_x + i / PAN ) >= N_MAX1 ) _from_x = N_MAX1  - i ;
                        else _from_x = _from_x + i / PAN ;
                        setcolor(SPECTRUM_COLOR);
                        clearviewport();
                        display_spe(sp,_from_x,&_to_x,_zoom,_y_low,_y_high);
                        break;
            case FAST_LEFT:
                        if ( _to_x >= N_MAX1 ) break;
                        i =  _to_x - _from_x ;
                        if( ( _to_x + 512 ) >= N_MAX1 ) _from_x = N_MAX1 - 511 ;
                        else _from_x = _from_x + 512;
                        if ( _from_x < 0 ) _from_x = 0;
                        setcolor(SPECTRUM_COLOR);
                        clearviewport();
                        display_spe(sp,_from_x,&_to_x,_zoom,_y_low,_y_high);
                        break;
            case RIGHT:
                        if( _from_x == 0 ) break;
                        i = ( _to_x - _from_x ) / PAN ;
                        if( ( _from_x - i ) <= 0 ) _from_x = 0 ;
                        else _from_x = _from_x - i;
                        setcolor(SPECTRUM_COLOR);
                        clearviewport();
                        display_spe(sp,_from_x,&_to_x,_zoom,_y_low,_y_high);
                        break;
            case FAST_RIGHT:
                        if ( _from_x == 0 ) break;
                        i =  _to_x - _from_x ;
                        if( ( _from_x - 512 ) <= 0 ) _from_x = 0 ;
                        else _from_x = _from_x - 512;
                        setcolor(SPECTRUM_COLOR);
                        clearviewport();
                        display_spe(sp,_from_x,&_to_x,_zoom,_y_low,_y_high);
                        break;
            case ’S’:{
                        auto int zoom, from_x, to_x;
                        auto float y_low, y_high;
                        gotoxy(1,1);
                        printf("%80c",’ ’);
                        gotoxy(1,1);
                        printf("FROM, TO, Y_LOW, Y_HIGH = ");
                        scanf("%d,%d,%f,%f",&from_x,&to_x,&y_low,&y_high);
                        display_menu();
                        if( y_low >= y_high) break;
                        if( from_x >= to_x ) break;
                        if( from_x < 0 ) break;
                        if( to_x > N_MAX1 ) break;
                        _to_x = to_x;
                        _from_x = from_x;
                        _y_low = y_low;
                        _y_high = y_high;}
                        _zoom =  X / ( _to_x - _from_x ) ;
                        if( _zoom < 1 ) _zoom = 1;
                        setcolor(SPECTRUM_COLOR);
                        clearviewport();
                        display_spe(sp,_from_x,&_to_x,_zoom,_y_low,_y_high);
                        break;
            case ’C’:
                        gotoxy(1,1);
                        printf("%80c",’ ’);
                        gotoxy(1,1);
                        printf("CURSOR: arrows, +, -, =, c    ESC - Quits    ");
                        cursor(&chn,&en);
                        setcolor(SPECTRUM_COLOR);
                        display_menu();
                        break;
            case ’L’:
                        setcolor(LOG_COLOR);
                        display_log(sp);
                        setcolor(SPECTRUM_COLOR);
                        break;
            case ’F’:
                        fit();
                        display_menu();
                        break;
            case ’R’:
                        cleardevice();
                        display_menu();
/*                      clearviewport();  */
                        setcolor(SPECTRUM_COLOR);
                        display_spe(sp,_from_x,&_to_x,_zoom,_y_low,_y_high);
                        break;
            case ’Q’:
                        goto end;
            case ’O’:
                        on_off();
                        display_menu();
```

```
                                break;
    default:
                                break;

  }
/*-------------------end of loop-------------------------------*/


end:
              gotoxy(1,1);
              printf("%80c",' ');
              gotoxy(1,1);
              printf("DO YOU WANT TO QUIT ?  y/n ");
              c = toupper( getch() );
              if( c != 'Y' ) goto start_of_the_main_loop ;

              restorecrtmode();


}
/*---------------end of main--------------------*/

float ymax(float sp[])
{
              int i,n = 649;
              extern int N_MAX1;
              float y = 10.;

              if ( N_MAX1 < n ) n = N_MAX1;

              for( i = 1 ; i <= n ; i++ )
                            if ( sp[i] > y ) y = sp[i];

              return(y);
}
void display_menu()
{
              extern int STATUS;

              gotoxy(1,1);
              printf("%80c",' ');
              gotoxy(1,1);
              printf("DISPLAY: + -, arrows, Scale,
           Cursor, Fit, New_file, Log, Regen, Quit,      ");
    if( STATUS ) printf("On ");
              else      printf("Off");



/*-----------------PART01.C--- shape functions-------------*/


#include <math.h>

extern struct peak_type {
              double position;
              double sigma;
              double beta;
              double ga;
              double ba;
```

```
            double ste;}  peak[5],tmp[5];

double step( double x, double center, double sigma)

{
            double t, f, erf( double f);

    t = ( x - center ) / sigma;    /*distance in units of sigma */

            if ( t < -2. )    return( 1. );        /*check the limits */
            if ( t >  2.5 )    return( 0. );

            f = erf ( t );        /*diffused step function */
            return ( f );
}

double step2( double x, double center, double sigma)

{
            double t, f;

            t = 1.75 * ( x - center ) / sigma;
             /*distance in units of sigma/1.75 */
            f = 1. / ( 1. + exp(t) );

            return ( f );
}

double erf(double t)
{
            int static n_of_steps = 5;        /*number of integration steps */
            double static factor = 2.5066;   /* sq. root of 2Pi                */
            int i;
            double y , x , s ;

            s = 4. / (double) n_of_steps;
            y = 0.;

            for (i = 0; i < n_of_steps; i++){
          x =  -4. + ( ( double ) i + .5 ) * s - t ;
                          y += exp( - x * x / 2. );
            }
            y = s / factor * y * 1.04;
          /*1.04- correction to make smooth change */
            return( y );
}

double bump ( double x, double center, double sigma, double beta)
{
            double t, f, b, rim ( double t, double beta );

            if ( beta <= 0. ) return (0.);

            t = ( x - center ) / sigma ;

            if ( t < -2.1 ) return ( exp( ( x - center ) / beta) );
            if ( t >  2.4 ) return (    0.  );

            b = beta / sigma;
            f = rim (t, b);
```

```
                return(f);
}

double bump2 ( double x, double center, double sigma, double beta)
{
                double t, f, b;

                if ( beta <= 0. ) return (0.);

                t = 1.75 * ( x - center ) / sigma ;
                b = ( x - center ) / beta ;

                f = exp(b) / ( 1. + exp(t) ) ;

                return(f);
}

double rim( double t, double beta)
{
                int static n_of_steps = 5;
                int i;
                double static factor = 2.5066;          /*              sq. root of 2Pi                         */
                double x, y, s, v;

                s = 5. / (double) n_of_steps;
                y = 0.;

                for( i = 0; i < n_of_steps; i++){
                                v = ( (double)i + 0.5 ) * s - 5.;
                                x = v - t;
                                y += exp( v / beta - x * x / 2. );
                }

                y = s / factor * y;
        return(y);
}

double gauss( double x, double center, double sigma )
{
                double t, y ;

/*              double  factor = 2.5066;                */

                t = ( x - center ) / sigma;

                if ( t < - 3.5 ) return (0.);
                if ( t >   3.5 ) return (0.);

                y = exp ( -t * t / 2. );

/* --- Gauss normalized to hight 0 for t=0-------------*/
/* --- to normlized area to 1: y = exp(-t*t/2.)/factor/sigma  --------*/

                return (y);
}

void bump_centroid_info(int n, double *cx, double *a)
/*--------- centroid = cx/a ;  a = bump area -----------*/
{
                int n_of_st = 5 ,i ;
        double s,b,bs,c,t,step,x;
        double rim(double x, double b);

        if(peak[0].ba < 0.000001 ){
                *cx = 0.;
                *a = 0.;
                return ;
        }

        s = peak[0].sigma;
        b = peak[0].beta;
        bs = b / s ;
        c = peak[n].position ;
        step = 5.6 * s / (double)n_of_st  ;

        *a = b * exp( -2.1 * s / b );
        *cx = b * exp( -2.1 * s / b ) * ( c - 2.1 * s - b );
        x = -2.1 * s - 0.5 * step;

        for( i = 0 ; i < n_of_st ; i++ ){
                x += step;
                t = rim( x/s, bs ) * step;
                *a += t ;
                *cx += ( x + c ) * t ;
        }

}


void peak_parameters(double *xmax, double *ymax, double *fwhm )
{
        double gauss( double x, double center, double sigma );
        double bump ( double x, double center, double sigma, double beta);
        double x,ba,s,b,x1,x2,t,t1,step,c;


        x = peak[0].position ;
        ba = peak[0].ba ;
        s = peak[0].sigma ;
        b = peak[0].beta ;
        c = peak[0].position ;

        if( ba < 0.000001 ){
                *ymax = 1. ;
                *xmax = x  ;
                *fwhm = 2.35 * s ;
                return;
        }

        step = 1. ;
        *ymax = gauss(x,c,s) + ba * bump(x,c,s,b) ;

/*-------------- searching for xmax, ymax------------*/
        do{
        x -= step;
        t = gauss(x,c,s) + ba * bump(x,c,s,b) ;
        if ( t <= *ymax ) break;
        *ymax = t ;
        } while ( x > 0. );
```

```
          x += step;
          step = 0.1 ;

          do{
          x -= step;
          t = gauss(x,c,s) + ba * bump(x,c,s,b) ;
          if ( t <= *ymax ) break;
          *ymax = t ;
          } while ( x < c );

          x += step;
          step = 0.01;

          do{
          x -= step;
          t = gauss(x,c,s) + ba * bump(x,c,s,b) ;
          if ( t <= *ymax ) break;
          *ymax = t ;
          } while ( x > 0. );

          *xmax = x + step;

/*-------end of search------------------------------*/

/*---------search for fwhm--------------------------*/

          x = *xmax + 1.17 * s ;

          step = 1. ;
    do{
          t = gauss(x,c,s) + ba * bump(x,c,s,b) ;
          if ( 2. * t <= *ymax ) break;
          x += step;
          } while( x < ( c + 2. * s ) );

          t1 = gauss(x - step,c,s) + ba * bump(x - step,c,s,b) - t ;
          if( t1 == 0. ) x2 = x - 0.5 * step;
                    else
                              x2 = x - step * ( *ymax / 2. - t ) /  t1

          x = *xmax * 2. - x2 ;

    do{
          t = gauss(x,c,s) + ba * bump(x,c,s,b) ;
          if ( 2. * t <= *ymax ) break;
          x -= step;
          } while( x > 0 );

          t1 = gauss(x + step,c,s) + ba * bump(x + step,c,s,b) - t;
          if ( t1 == 0. ) x1 = x + 0.5 * step ;
                    else
                              x1 = x + step * ( *ymax / 2. - t ) / t1

          *fwhm = x2 - x1 ;


/*-------------PART02.C------- display #1 -----------*/
```

```
#include <graphics.h>
#include <conio.h>

void DrawBorder()
{
        struct viewporttype vp;

        getviewsettings( &vp );
        rectangle( 0,0, vp.right-vp.left, vp.bottom-vp.top );
}

void initialize()
{
        int graph_driver, g_m;
        struct viewporttype vp;
        extern int X, Y;
        extern int LINE;
        struct text_info t_i;


        detectgraph(&graph_driver, &g_m);
        if ( graph_driver < 0 ) exit(1);

/*----PROBLEMS with high resolution mode: gotoxy() doesn't work------
------for y > 25 (it should work for y <= 30 in graphic mode---------*/
        if ( g_m == VGAHI ) g_m = VGAMED;
/*-----------------------------------------------------------------*/

        initgraph( &graph_driver, &g_m ,"");


        gettextinfo(&t_i);
        LINE = t_i.winbottom;
        if( g_m == VGAHI ) LINE = 30;


        setcolor(3);
        setlinestyle( SOLID_LINE, 0, NORM_WIDTH );
        getviewsettings( &vp );
        X = vp.right;
        Y = vp.bottom;
        cleardevice();

/*---------only for the initial display------------*/
{
        auto int l,r,t,b,h,w;
        l = vp.left + 20;
        r = vp.right - 20;
        t = vp.top + Y/LINE + 21;
        b = vp.bottom - Y/LINE - 21;
        setviewport( l, t, r, b, 1);

        settextstyle(TRIPLEX_FONT, HORIZ_DIR,10);
        settextjustify(CENTER_TEXT,TOP_TEXT);
        outtextxy( (r-l) / 2, 0, "Si(Li)-Fit");

        settextstyle(TRIPLEX_FONT, HORIZ_DIR,6);
        h = textheight("H");
        settextjustify(RIGHT_TEXT,BOTTOM_TEXT);
        outtextxy( r - h , b - h, "W. H. Trzaska 1989");
```

```
                DrawBorder();

}
/*-----------------------------------------------------*/


}

/*------------------------------not used in fit-----------------------*/

void display ( double x[], double y[], int size )
{
                struct viewporttype vp;
      double x_low, x_high, y_low, y_high, f_x, f_y;
                int i, xpix, ypix;
                void initialize();
                void DrawBorder();
                void getlimits( double x[], int size, double *x_low, double *x_high);

                initialize();
                getviewsettings( &vp );
                setviewport( vp.left + 1, vp.top + 50, vp.right - 1,
                        vp.bottom - 15, 1);
                getviewsettings( &vp );
                getlimits( x, size, &x_low, &x_high);
                getlimits( y, size, &y_low, &y_high);
                DrawBorder();
                f_x = (double) (vp.right - vp.left) / ( x_high - x_low);
                f_y = (double) (vp.top - vp.bottom) / ( y_high - y_low);

                for (i = 0; i < size; i++){
                                xpix = (int) ( ( x[i] - x_low) * f_x +.5 ) ;
                                ypix = (int) ( ( y[i] - y_low) * f_y +.5 )
                                 + vp.bottom - vp.top;
/*                              putpixel( xpix, ypix, 3);*/
                                circle( xpix, ypix, 2);
                }
}
/*-------------------------------------------------------------------*/

void getlimits( double x[], int size, double *x_low, double *x_high)
{
                int i;
                double t, low, high;

                low = x[0];
                high = x[0];

                for( i = 1; i < size; i++){
                                t = x[i];
                                if( t < low ) low = t;
                                                else if( t > high) high = t;
                }
                if( low == high) high += 1.;
                *x_low = low;
                *x_high = high;
}

/*-----------------PATR03.C----- readspectrum-----------*/
```

```
#include <conio.h>
#include <stdio.h>
#include <math.h>

double a_calib = 0.1245, b_calib = 5.6;

double ds_minimum = 0.006;
double dc_minimum = 0.006;
double db_minimum = 0.02;

FILE *csv, *areas;
int STATUS = 1;      /* says if en calib is on == 1 or off == 0  */

extern struct peak_type {
            double position ;
            double sigma ;
            double beta ;
            double ga ;
            double ba ;
            double ste ; }     peak[5], tmp[5]



void readspectrum( float sp[], int * n_max )
{
            FILE *fp, *fopen();
            static char file_name[30];
            char new_name[30], c, tmps[200] ;
            int i, j = 0 , k = 0, l;
            static int first_time = 1;
            extern int LINE, STATUS;

            void write_label();
            void write_label_areas();
            void on_off();

                        if(STATUS == 0) on_off();
          /* restore dx_minimum values into keV:*/
                        ds_minimum *= a_calib;
                        db_minimum *= a_calib;
                        dc_minimum *= a_calib;

            if(first_time == 1){
                        first_time = 0 ;
                        strcpy(file_name , "c:\\tmp\\file.asc");
                        csv = fopen("c:\\tmp\\fit.csv","a");
                        areas = fopen("c:\\tmp\\areas.csv","a");
                        gotoxy(10,10);
                        if (csv == NULL) printf("problems with csv");


get_the_name_of_the_file:

            do{
                 gotoxy( 1,1 );
                        printf("%80c",' ');
                        gotoxy( 1,1 );
                        printf("%s   Read an other file: y/n? ",file_name);
                        c = getch();
```

```
                if (c == ’y’ || c == ’Y’){
                gotoxy(1,1);
                printf("%80c",’ ’);
                gotoxy(1,1);
                printf("new file = ");
      scanf("%s",new_name);
                strcpy(file_name,new_name);
                }
                gotoxy( 1,1 );
                printf("%80c",’ ’);
        }while( (fp = fopen(file_name,"r")) == NULL );
/*
        fp = fopen("c:\\tmp\\133ba.asc","r");
*/
        gotoxy(1,1);
        printf("WAIT! Reading the file              \n");

/*-----------------READING THE FILE-----------------------------*/

        if (( fgets(tmps, 80, fp ) ) == NULL ){
                printf("empty file             \n");
                goto get_the_name_of_the_file;
                }

        printf("%s",tmps);
        for(i = 0; i < 6; i++){
        if (( fgets( tmps,80,fp ) ) == NULL ) goto koniec;
        printf("%s",tmps);
        }
        fscanf(fp,"%s",tmps);
        j = atoi(tmps);
        if( j > 0 && j < 8000)
                for(i=0; i < j; i++) sp[i] = 0.;
        fscanf( fp, "%s", tmps);
        k = atoi(tmps);
        printf("%d - %d\n",j,k);
        if(j > k){
                printf("wrong channel limits\n");
                goto get_the_name_of_the_file;
                }
        for(i = j; i <= k; i++){
            if ( (fscanf(fp, "%s", tmps ) == EOF)) goto koniec;
                sp[ i ] = atof( tmps );
                }


koniec:
        *n_max = i;

/*----------end of reading the file-------------------*/

        if(LINE <= 25)gotoxy(1,LINE);
                else {
                        gotoxy(1,25);
                        for(j = 25; j < LINE ; j++) printf("\n");
                }
        printf("%79s",file_name);

        fprintf(csv,"%s \n",file_name);
```

```
                fprintf(areas,"%s \n",file_name);

get_the_energy_calibration:



                gotoxy(1,1);
                printf("%80c",' ');
                gotoxy(1,1);
                printf("EN. CALIB: %f keV/ch %f keV  Change it ? y/n  "
                                         , a_calib,b_calib);
                c = getch();

                if( c == 'y' || c == 'Y' ){

                                gotoxy(1,1);
                                printf("%80c",' ');
                                gotoxy(1,1);
                                printf("keV/ch = ");
                                scanf("%lf",&a_calib);
                                if( a_calib == 0. ) a_calib = 1. ;

                                gotoxy(1,1);
                                printf("%80c",' ');
                                gotoxy(1,1);
                                printf("keV = ");
                                scanf("%lf",&b_calib);

                                STATUS = 1 ;   /*  en. calibration on */

                                goto get_the_energy_calibration ;

                }
                else
                fprintf(csv,"%.6f, keV/ch,% .2f, keV \n", a_calib, b_calib );
                fprintf(areas,"%.6f, keV/ch,% .2f, keV \n", a_calib, b_calib );
       write_label();
                write_label_areas();

/*     change dx_minimum values from keV into chn.   */

                                ds_minimum /= a_calib;
                                db_minimum /= a_calib;
                                dc_minimum /= a_calib;
}

void on_off()
/*----- switches energy calibration on/off ------*/
{
                extern int STATUS;
                extern double a_calib, b_calib;
                static double a = 1. , b = 0. ;

                if( STATUS ) {
                                a = a_calib  ;
                                b = b_calib  ;
                                a_calib = 1. ;
                                b_calib = 0. ;
                                STATUS = 0   ;
                }
```

```
          else{
                    a_calib = a ;
                    b_calib = b ;
                    STATUS = 1

return;
}


void write_present_parameters()
{
          extern int LINE, n_of_p, STATUS ;
          extern double a_calib, b_calib, CHI2;
          double en, gauss_fwhm, s,beta, ga, ba, ste,
                    c_of_g, cx, ab, ag, as, a, c;
          double fwhm, ymax, xmax;
          double factor = 2.5066; /*--- sq. root of 2Pi ---*/
          int i;
          void peak_parameters(double *xmxa, double *ymax, double *fwhm );

          s = peak[0].sigma;
          beta = peak[0].beta;
          ba = peak[0].ba;
          ste = peak[0].ste;
          gauss_fwhm = 2.35 * s ;

          peak_parameters(&xmax,&ymax,&fwhm);

          fwhm *= a_calib;
          gauss_fwhm *= a_calib;
          beta *= a_calib;

          gotoxy(1,10);
          if (STATUS) printf("[keV] ");
                    else  printf("[Chn] ");

          for( i = 0; i < n_of_p ; i++ ){

                    if (STATUS) fprintf(csv,"[keV], ");
                              else fprintf(csv,"[Chn], ");

                    printf("chi2=%.1f  \n",CHI2);
                    fprintf(csv,"%.1f, ",CHI2);

                    c = peak[i].position;
                    en = c * a_calib + b_calib;
                    bump_centroid_info(i, &cx, &ab );
                    ga = peak[i].ga;
                    cx *= ga * ba ;
                    ab *= ga * ba ;
                    ag = ga * factor * s ;
                    a = ag + ab  ;
                    c_of_g = ( cx + c * ag ) / a * a_calib + b_calib;
                    as = c * ste * ga ;
                    printf(" %1d, %.2f, %.2f, %.2f, %.2f, %.2f, %.2f,
             %.2f, %.0f, %.0f, %.0f\n",i+1,en,gauss_fwhm,
             beta,ba * 100.,ste * 100.,c_of_g,fwhm,a,ag/a*100.,
                                        a/(a+as)*100.);
                    fprintf(csv," %1d, %.2f, %.2f, %.2f, %.2f, %.2f,
                %.2f, %.2f, %.0f, %.0f, %.0f\n",
                                i+1,en,gauss_fwhm,beta,ba * 100.,ste * 100.,
                        c_of_g,fwhm,a,ag/a*100.,a/(a+as)*100.);
                }
}
void write_label()
{
                extern FILE *csv;

                fprintf(csv,"units,chi2,peak,c-gauss,fw-gauss,beta,bump,
              step,centroid,FWHM,area,g/a[%%],a/total[%%]\n");


}
void write_label_areas()
{
                extern FILE *areas;

                fprintf(areas,"centroid,area,total,from,to,b0,b1,b2\n");


}
void centroid()
{
                int i;
                double x, t, T = 0., A = 0., C, c = 0.  ;
                extern float *pointer_to_sp;
                extern double b0, b1, b2, a_calib, b_calib ;
                extern int fit_from, fit_to, N_MAX1;
                extern FILE *areas;



                if( fit_from > fit_to)return;
                if( fit_from < 0 )return;
                if( fit_to > N_MAX1 )return;

                for ( i = fit_from ; i <= fit_to; i++ ){
                        x = (double)i + 0.5 ;
                        t = *(pointer_to_sp + i);
                        T += t;
                        t = t - b0 - b1 * x - b2 * x * x ;
                        A += t;
                        c += t * x;
                }
                if( A <= 0. )return;
                C = c / A;
                C = C * a_calib + b_calib;

                gotoxy(1,2);
                printf("Centroid = %.1f Area = %.0f Total = %.0f", C, A, T);
                fprintf(areas,"%.1f, %.0f, %.0f, %d, %d, %.1f, %.3f, %.5f\n",
                        C,A,T,fit_from,fit_to,b0,b1,b2);

}

/*------------PART04.C----- display -------------------*/

#include <graphics.h>
#include <math.h>
#include <stdlib.h>
```

```
#define ESC 27
#define LEFT 75
#define RIGHT 77
#define FAST_LEFT 52
#define FAST_RIGHT 54
#define SPACE 32
#define CURSOR_COLOR 3

int _zoom, _from_x, _to_x, top,bottom, left, right;
float  _y_low, _y_high, scalefactor;


void display_spe ( float y[], int from_x, int *to_x, int zoom,
                        float y_low, float y_high )
{
            int i ;
            void DrawBorder();
            void add_ticks();
            void gotoxy_wt( int x, int y);
            extern int X, Y, N_MAX1, LINE ;
            extern double a_calib, b_calib;

            double ei, ef;

            if ( from_x > N_MAX1 ) return;

            _zoom = zoom;
            _from_x = from_x;
            _y_low = y_low;
            _y_high = y_high;

            left = 0    ;
            right = X   ;
            top = 0  + Y/LINE + 1 ;
            bottom = Y - Y/LINE - 1 ;
            setviewport( left  , top , right , bottom , 1);
            DrawBorder();

       *to_x = from_x + ( right - left ) / zoom;
            if ( *to_x > N_MAX1 ) *to_x = N_MAX1;
            _to_x = *to_x ;
            scalefactor = ( y_high - y_low ) / (float) (- top + bottom );

            moveto(0,bottom);
            for ( i = from_x; i <= *to_x; i++ ){

                        auto int xpix, ypix;

                        if ( y[i] <= y_high )
                                    ypix = bottom - top - ( int ) ( (y[i] - y_low)
                                          / scalefactor );
                        else
                                    ypix = 0 ;

                        xpix = (i - from_x) * zoom - 1 ;
                        lineto(xpix,ypix);
                        linerel(zoom,0);
            }

            ei = (double) (_from_x) * a_calib + b_calib;
```

```
        ef = (double) (_to_x) * a_calib + b_calib;

gotoxy_wt(1,LINE);
printf("                                        ");
gotoxy_wt(1,LINE);
printf("X: %d-%d (%.0f-%.0f keV) Y: %.0f-%.0f z: %d ",
from_x,_to_x,ei,ef,y_low,y_high,zoom);

        add_ticks();
}


void display_fit ( float fit[], int fit_from, int fit_to )
/*
display_fit uses variables that are defined in display_spe
and therefore it can't be called first
*/
{
        int  tmpi,xi,yi,i ;
        extern int _zoom, _from_x,top,bottom;
        extern float _y_low,_y_high, scalefactor;
        extern int X, Y, N_MAX1, LINE ;

        if ( fit_from > N_MAX1 ) return;

        tmpi =  bottom - top;
        xi = ( fit_from - _from_x ) * _zoom - 1;
        if ( xi > X ) return;
        yi = tmpi - (int) (( fit[fit_from] -_y_low ) / scalefactor);
        moveto(xi,yi);
        for ( i = fit_from  ; i <= fit_to; i++ ){


                if ( fit[i] <= _y_high )
                        yi = tmpi - ( int ) ( (fit[i] - _y_low)
                    / scalefactor );
                else
                        yi = 0 ;

                xi = (i - _from_x) * _zoom - 1 ;
                lineto(xi,yi);
                linerel(_zoom,0);
        }
}
void display_log ( float sp[] )
/*
display_log uses variables that are defined in display_spe
and therefore it can't be called first
*/
{
        int  xi,yi,i ;
        extern int _zoom, _from_x, _to_x, top, bottom;
        extern float _y_low,_y_high;
        extern int X, Y, N_MAX1, LINE ;
        double ll,lh,scale;


        if( _y_high <= 1. ) return;
        lh = log ( _y_high );
        ll = 0. ;
```

```
            if ( _y_low > 1. ) ll = log ( _y_low );
            scale = ( lh - ll ) / (double) (bottom - top) ;


            moveto(0,bottom);

            for ( i = _from_x  ; i <= _to_x; i++ ){


                        if ( sp[i] <= _y_high )
                                    yi = (sp[i] > 1.) ? bottom - top
          - ( int ) ( (log ( sp[i] ) - ll ) / scale ) : bottom - top ;
                        else
                                    yi = 0 ;

                        xi = (i - _from_x) * _zoom - 1 ;
                        lineto(xi,yi);
                        linerel(_zoom,0);
            }
}

void cursor(int *c_channel, double *c_energy)
{
            static int  first_time = 1;
            static int size;
            static void *Cursor;
            double energy, counts;
            extern double a_calib , b_calib ;
            extern int _zoom, _from_x, _to_x, top,bottom;
            extern float _y_low,_y_high, scalefactor;
            extern int X, Y, N_MAX1, LINE ;
            extern float *pointer_to_sp;
            void display_spe(float sp[], int form_x, int *to_x,
               int zoom, float y_low, float y_high);
            int c, speed, x, channel, l, xx;
            static int m = 1 ;

if( first_time ){
auto int y;
            first_time = 0;
            y = bottom - top;
            setlinestyle(DOTTED_LINE,0,NORM_WIDTH);
            setcolor(CURSOR_COLOR);
            line(1,0,1, y);
            setlinestyle(SOLID_LINE,0,NORM_WIDTH);
            size = imagesize(1,0,1, y);
            Cursor = malloc( size );
            getimage(1,0,1,y,Cursor);
            putimage(1,0,Cursor,XOR_PUT);
}
            xx = right - left;
            speed = _zoom;
            x = xx / 2 / _zoom * _zoom + _zoom / 2 ;
            putimage(x,0,Cursor,XOR_PUT);
            gotoxy(42,1);
            channel = _from_x + x / _zoom;
            energy = b_calib + a_calib * ( (float) _from_x + (float) x
                                    / (float) _zoom );
            *c_channel = channel;
            *c_energy = energy;
```

```
        counts = * ( pointer_to_sp + channel );

        printf("%4d ch = %d en = %.1f c = %.0f
                           speed,channel,energy,counts);



/*-----------cursor loop-----------------*/

while( ( c = toupper ( getch() ) ) != ESC ){

        switch (c) {
            case’+’:    /*------increase cursor speed-------------*/
                        if ( speed >= 30 ) break;
                        speed = speed * 2;
                        break;
            case’-’:    /*------decrease cursor speed------------*/
                        if( speed == 1 ) break;
                        speed = speed / 2 ;
                        break;
            case’=’:    /*------display from here----------------*/
                        _from_x = channel;
                        clearviewport();
                        display_spe(pointer_to_sp,_from_x,&_to_x,_zoom,_y_low,_y_high);
                        return;
            case LEFT:
                        if ( x <= 0 ) break;
                        putimage(x,0,Cursor,XOR_PUT);
                        x = x - speed ;
                        if ( x < 0 ) x = 0 + _zoom / 2 ;
                        putimage(x,0,Cursor,XOR_PUT);
                        break;
            case FAST_LEFT:
                        if ( x <= 0 ) break;
                        putimage(x,0,Cursor,XOR_PUT);
                        x = x - 10 * _zoom ;
                        if ( x < 0 ) x = 0 + _zoom / 2 ;
                        putimage(x,0,Cursor,XOR_PUT);
                        break;
            case RIGHT:
                        if ( x >= xx ) break;
                        putimage(x,0,Cursor,XOR_PUT);
                        x = x + speed ;
                        if ( x > xx ) x = xx - _zoom / 2 ;
                        putimage(x,0,Cursor,XOR_PUT);
                        break;
            case FAST_RIGHT:
                        if ( x >= xx ) break;
                        putimage(x,0,Cursor,XOR_PUT);
                        x = x + 10 * _zoom ;
                        if ( x > xx ) x = xx - _zoom / 2 ;
                        putimage(x,0,Cursor,XOR_PUT);
                        break;
            case SPACE:
                        l = ( 80 * (unsigned) (x + left) ) / X - 1;
                        m += 1;
                        if ( m >= ( LINE / 3 ) ) m = 2;
                        if ( l < 1 ) l = 1;
                        if ( l > 75 ) l = 75;
                        gotoxy( l , m);
```

```
                                printf("%5.1f",energy);
                                return;
                case 'C':
                                l = ( 80 * (unsigned) (x + left) ) / X - 1;
                                m += 1;
                                if ( m >= ( LINE / 3 ) ) m = 2;
                                if ( l < 1 ) l = 1;
                                if ( l > 75 ) l = 75;
                                gotoxy( l , m);
                                printf("%5.1f",energy);
                                x = X / 2 / _zoom * _zoom + _zoom / 2 ;
                                putimage(x,0,Cursor,XOR_PUT);
                                break;
                default:
                                break;
                }
gotoxy(42,1);
channel = _from_x + x / _zoom;
energy = b_calib + a_calib * ( (float) _from_x + (float) x / (float) _zoom );
counts = * ( pointer_to_sp + channel );

*c_channel = channel;
*c_energy = energy;

printf("%4d ch = %d en = %.1f c = %.0f      ",speed,channel,energy,counts);

}
/*------------end of loop----------------------------*/


putimage(x,0,Cursor,XOR_PUT);

}



void gotoxy_wt(int x, int y)
{
                int i;

                if(y <= 25) gotoxy(x,y);
                                else{
                                                gotoxy(x,25);
                                                for(i = 25; i < y; i++) printf("\n");
                                }
}

void add_ticks()
{
                extern double a_calib, b_calib;
                extern int _from_x, _to_x, top, bottom, left, right;
                extern float _y_low, _y_high, scalefactor;

                int d, xpix, ypix;
                double step_y, step_x, diff_y, diff_x, y_start, x_start, x0, xf;
                double tmp;

                diff_y = _y_high - _y_low;
                d = (int) ( log10( diff_y ) ) ;
                step_y = pow(10., (double) d);
```

```
        if( diff_y / step_y <= 3.) step_y /= 2.;
        if( diff_y / step_y <= 3.) step_y /= 2.5;

        tmp = _y_high / step_y;
        if ( tmp > 16000.) return;

        d = (int) tmp;
        y_start = (double) ( d * step_y ) ;

        while( y_start > _y_low ){
                        ypix = bottom - top - (int) ( ( y_start - _y_low )
                                / scalefactor );
                        xpix = 1;
                        putpixel(xpix,ypix,3);
                        xpix = right - left - 1 ;
                        putpixel(xpix,ypix,3);
                        y_start -= step_y;
        }


        x0 = (double) (_from_x) * a_calib + b_calib - a_calib / 2.;
        xf = (double) (_to_x) * a_calib + b_calib + a_calib / 2.;

        diff_x = xf - x0 ;
        d = (int) ( log10( diff_x ) ) ;
        step_x = pow(10., (double) d);

        if( diff_x / step_x <= 3.) step_x /= 2.;
        if( diff_x / step_x <= 3.) step_x /= 2.5;


        tmp = xf / step_x;
        if ( tmp > 16000.) return;

        d = (int) tmp;
        x_start = (double) ( d ) * step_x  ;

        while( x_start > x0 ){
                        ypix = bottom - top - 1 ;
                        xpix = (int) ( ( x_start - x0 ) * (double) ( right - left )
            / diff_x );
                        putpixel(xpix,ypix,3);
                        ypix = 1;
                        putpixel(xpix,ypix,3);
                        x_start -= step_x;
        }



        return;
}

/*-------------------PART05.C-------FIT---------------*/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <graphics.h>
```

```
#define ESC 0x1b
#define UP 72
#define DOWN 80
#define LEFT 75
#define FAST_LEFT 52
#define RIGHT 77
#define FAST_RIGHT 54
#define SPECTRUM_COLOR 1
#define BGD_COLOR 1
#define FIT_COLOR 3


struct peak_type {
            double position ;
            double sigma ;
            double beta ;
            double ga ;
            double ba ;
            double ste ; }     peak[5], tmp[5]  ;

int     n_of_p = 0, fit_from = 10000, fit_to = 0 ;
double  b2 = 0. , b1 = 0. , b0 = 0. ;

void fit()
{

            int c, channel, i, j ;

            double peak_value( int peak_number, int channel );
            void display_spe(float[],int from_x,int *to_x,int zoom,
                        float y_low, float y_high);
            void display_fit( float fit[], int fit_from, int fit_to );
            void display_log( float sp[]);
            void display_fit_menu();
            void peak_fit();
            void bgd();
            void cursor(int *c_channel, double *c_energy);
            void show_present_parameters();
            void write_present_parameters();
            void centroid();
            void display_components();
            int check_everything();



            extern int    N_MAX1, X, Y, LINE ;
            extern float *pointer_to_sp, *pointer_to_spf;
            extern int _from_x, _to_x , _zoom, fit_from, fit_to;
            extern float _y_low, _y_high ;

            float *sp, *spf, y ;
            double x;

            sp = pointer_to_sp;
            spf = pointer_to_spf;

            display_fit_menu();


/*----------------main loop------------------------------*/
```

```
while( ( c = toupper ( getch() ) ) != ESC ){

switch (c) {
            case ’B’:
                        bgd();
                        for ( i = _from_x; i <= _to_x ; i++ )
                 *(spf + i) = b0 + b1 * i + b2 * i * i;
                        setcolor(BGD_COLOR);
                        display_fit( spf, _from_x, _to_x );
gotoxy(1,LINE);
printf(" %10f %10f %10f ",b0,b1,b2);
                        break;

     case ’R’:
                        clearviewport();
          setcolor(SPECTRUM_COLOR);
                        display_spe(sp, _from_x, &_to_x, _zoom, _y_low, _y_high);
                        break;
            case ’Q’:
                        return;
            case ’P’:
                        peak_fit();
                        break;
            case ’C’:
                        if( check_everything() == 0 ) break;
                        display_components();
                        break;
            case ’S’:
                        if( check_everything() == 0 ) break;
                        for ( i = _from_x; i <= _to_x ; i++ ){
                                    x = (double)i + 0.5 ;
                                    *(spf + i) = b0 + b1 * x + b2 * x * x ;
                                    for ( j = 0; j < n_of_p ; j++)
                                                *(spf + i) += peak_value(j,i);
                        }
                        setcolor( FIT_COLOR ) ;
                        display_fit( spf, fit_from, fit_to );
                        break;
            case ’D’:
                        if( check_everything() == 0 ) break;
                        y = ( _y_high + _y_low ) / 2. ;
                        for ( i = fit_from; i <= fit_to ; i++ )
                *(spf + i) = *(sp + i) - *(spf + i) + y;
                        setcolor(BGD_COLOR);
                        display_fit( spf, fit_from, fit_to );
                        for ( i = fit_from; i <= fit_to ; i++ ) *(spf + i) = y;
                        display_fit( spf, fit_from, fit_to );
                        break;
            case ’V’:
                        if( check_everything() == 0 ) break;
          show_present_parameters();
                        break;
            case ’W’:
                        if( check_everything() == 0 ) break;
                                    write_present_parameters();
                        break;
            case ’I’:
                        centroid();
                        break;
```

```
  default:
                        break;
            }
}
/*------------------end of loop-------------------------------*/

}

void display_fit_menu()
{
            gotoxy(1,1);
            printf("%80c",' ');
            gotoxy(1,1);
            printf("FIT: Bgd, Peak    Regen, Show, Diff,
           Values, Comp., Int, Write     ESC - quits");
}


void bgd()
{
            int tmp, ch, c ;
            static int from, to;
            double en ;
            void cursor(int *c_channel, double *c_energy);
            void display_fit_menu();
            void fit_bgd( int from, int to );

            gotoxy(1,1);
            printf("%80c",' ');
            gotoxy(1,1);
            printf("BGD: Null, Manual, Fit, Repeat,         ESC - quits");

            tmp = toupper( getch() ) ;

  switch(tmp){
            case ESC:
                        break;
            case 'N':
                        b2 = 0.;
                        b1 = 0.;
                        b0 = 0.;
                        break;
            case 'M':
                        gotoxy(1,1);
                        printf("                                   ");
                        gotoxy(1,1);
                        printf("b0,b1,b2 = ");
                        scanf("%lf,%lf,%lf",&b0,&b1,&b2);
                        break;
            case 'F':
                        gotoxy(1,1);
                        printf("%80c",' ');
                        gotoxy(1,1);
                        printf("FITTING BGD: Enter limits  ");
                        cursor(&to, &en);
                        cursor(&from, &en);
                        if( from > to ){ c = to; to = from; from = c;}
                        if( ( to - from ) < 3 ) break;
                        fit_bgd( from, to );
                        break;
```

```
        case ’R’:
                if( from > to ){ c = to; to = from; from = c;}
                if( ( to - from ) < 3 ) break;
                fit_bgd( from, to );
                break;

        default:
                break;
  }

        display_fit_menu();
}

void fit_bgd( int from, int to )
{
        extern float *pointer_to_sp;
        extern double b2, b1, b0 ;
        int i, tmp;
        double Sy = 0., Sx = 0., Sxy = 0., Sx2 = 0., n, D2 ;
        double ii, Sx3 = 0., Sx4 = 0., Sx2y = 0., D3;

        float *sp;

        sp = pointer_to_sp;


        gotoxy(1,1);
        printf("%80c",’ ’);
        gotoxy(1,1);
        printf("BGD SHAPE: Flat, Linear, Parabolic          ESC - quits");

        tmp = toupper( getch() ) ;

  switch(tmp){
        case ESC:
                break;
        case ’F’:
                b2 = 0.;
                b1 = 0.;
                b0 = 0.;
                for( i = from; i <= to; i++)  b0 += (double) *(sp + i);
                b0 /= (double) ( to - from + 1);
                break;
        case ’L’:
                b2 = 0.;
                n = (double) ( to - from + 1 );

                for( i = from; i <= to; i++ ){
                        Sy += *(sp + i);
                        ii = (double) i;
                        Sx += (double) i;
                        Sxy += ii * *(sp + i);
                        Sx2 += ii * ii;
                }

                D2 = n * Sx2 - Sx * Sx ;

                if ( abs( D2 ) < pow( 10., -6.) ) {
                        b1 = 0.;
                        b0 = 0.;
```

```
                  }
                else {
                                b1 = ( n * Sxy - Sx * Sy ) / D2;
                                b0 = ( Sy * Sx2 - Sxy * Sx ) / D2;
                   }

                break;
        case ’P’:
                n = (double) ( to - from + 1 );

                for( i = from; i <= to; i++ ){

                                ii = (double) i;
                                Sy += *(sp + i);
                                Sx += ii;
                                Sxy += ii * *(sp + i);
                                Sx2 += ii * ii;
                                Sx3 += ii * ii * ii;
                                Sx4 += ii * ii * ii * ii;
                                Sx2y += ii * ii * *(sp + i);
                }

                D3 = n * Sx2 * Sx4 + Sx * Sx3 * Sx2 * 2.
                    - Sx2 * Sx2 * Sx2 - Sx * Sx * Sx4 - n * Sx3 * Sx3 ;

                if (  D3 * D3 < pow( 10., -16.) ) {
                                b2 = 0.;
                                b1 = 0.;
                                b0 = 0.;
                  }
                else {
                                b2 = (n * Sx2 * Sx2y + Sx * Sxy * Sx2
                                    + Sx * Sx3 * Sy - Sx2 * Sx2 * Sy
                                    - Sx * Sx * Sx2y - Sx3 * Sxy * n  ) / D3;
                                b1 = (n * Sxy * Sx4 + Sy * Sx3 * Sx2
                                    + Sx2 * Sx * Sx2y - Sx2 * Sxy * Sx2
                                    - Sx * Sy * Sx4 - Sx2y * Sx3 * n  ) / D3;
                                b0 = (Sy * Sx2 * Sx4 + Sx * Sx3 * Sx2y
                                    + Sxy * Sx3 * Sx2 - Sx2y * Sx2 * Sx2
                                    - Sxy * Sx * Sx4 - Sx3 * Sx3 * Sy  ) / D3;
                   }
                break;

        default:
                break;
  }
}
void peak_fit()
{
/*
        double step( double x, double center, double sigma);
        double bump ( double x, double center, double sigma, double beta);
        double gauss( double x, double center, double sigma );
*/
        void display_spe(float[],int from_x,int *to_x,int zoom,
                     float y_low, float y_high);
        void display_fit( float fit[], int fit_from, int fit_to );
        void display_log( float sp[]);
        void cursor(int *c_channel, double *c_energy);
        void display_fit_menu();
        int check_everyting();
        void peak_fit_fit();

        extern double a_calib, b_calib;
        extern double b2, b1, b0 ;
        extern int STATUS;

        int tmp,  c, i;
        extern int fit_from, fit_to;
        double en , fwhm, t;
        char ca;

        gotoxy(1,1);
        printf("%80c",' ');
        gotoxy(1,1);
        printf("PEAK FITTING:  Limits  Peak_pos.
         Shape_par.     Fit       ESC - quits");

/*-----------------loop---------------*/

while( ( tmp = toupper( getch() ) ) != ESC ){

        switch(tmp){
        case ESC:
                break;
        case 'L':
                gotoxy(1,1);
                printf("%80c",' ');
                gotoxy(1,1);
                printf("USE THE SPACE BAR TO MARK THE LIMITS  ");
                cursor( &fit_from, &en);
                cursor( &fit_to, &en);
                if ( fit_from > fit_to )
         { c = fit_from; fit_from = fit_to; fit_to = c ;}
                break;
        case 'P':
                gotoxy(1,1);
                printf("%80c",' ');
                gotoxy(1,1);
                printf("HOW MANY PEAKS (1-3) ? ");
                ca = getche();
                if( ca < '1' || ca > '3' ){
                        gotoxy(1,1);
                        printf("Must be 1 - 3. Press P (twice) to continue,
                     ESC - to quit");
                        break;
                }
                n_of_p = atoi( &ca );
                gotoxy(1,1);
                printf("%80c",' ');
                gotoxy(1,1);
                printf("USE THE SPACE BAR TO MARK PEAK POSITION(S) ");
      for ( i = 0; i < n_of_p ; i++ ){
                        cursor(&c, &en);
                        peak[i].position = ( en - b_calib ) / a_calib;
                }
                break;
        case 'S':
read_parameters:
                gotoxy(1,1);
                printf("%80c",' ');
                gotoxy(1,1);
                if (STATUS) printf("[keV] ");
                        else printf("[Chn] ");
                fwhm = peak[0].sigma * 2.35;
                printf("1)fwhm = %.2f 2)beta = %.1f
      3)bump%% = %.1f 4)step%% = %.1f ",
                fwhm * a_calib ,peak[0].beta * a_calib, peak[0].ba * 100. ,
                peak[0].ste * 100.);
                c = getch();
                if( c == '1' ){
                        printf("fwhm = ");
                        scanf("%lf",&fwhm);
                        peak[0].sigma = fwhm / 2.35 / a_calib;
                        goto read_parameters;
                }
                if( c == '2' ){
                        printf("beta = ");
                        scanf("%lf",&peak[0].beta);
                        peak[0].beta /= a_calib;
                        goto read_parameters;
                }
                if( c == '3' ){
                        printf("bump%% = ");
                        scanf("%lf",&t);
                        peak[0].ba = t / 100. ;
                        goto read_parameters;
                }
                if( c == '4' ){
                        printf("step%% = ");
                        scanf("%lf",&t);
                        peak[0].ste = t / 100.;
                        goto read_parameters;
                }
/*----------------------------------------------------------------*/

                gotoxy(1,1);
                printf("%80c",' ');
                gotoxy(1,1);
                printf("Enter peak position manualy: y/n");
                ca = getche();
                if( ca != 'y' && ca != 'Y' )break;

                gotoxy(1,1);
                printf("%80c",' ');
                gotoxy(1,1);
                printf("HOW MANY PEAKS (1-3) ? ");
                ca = getche();
                if( ca < '1' || ca > '3' ){
                        gotoxy(1,1);
                        printf("Must be 1 - 3. Press P (twice) to continue,
          ESC - to quit");
                        break;
                }
                n_of_p = atoi( &ca );

read_peak_position:
                gotoxy(1,1);
                printf("%80c",' ');
                gotoxy(1,1);
```

```
                    for(i = 0; i < n_of_p; i++ ){
                                t = peak[i].position * a_calib + b_calib;
                                printf(" %1d)=%.2f",i+1,t);

                    }
        ca = getch();
                    if( ca < '1' || ca > '5' ) break;
                    c = atoi(&ca);
                    if ( c > n_of_p ) break;
                    printf( " %1d) = ",c);
                    scanf("%lf",&t);
                    peak[c-1].position = (t - b_calib) / a_calib ;
                    goto read_peak_position;
/*---------------------------------------------------------------------*/
/*                  break;                                          */
            case 'F':
                    if ( check_everything() == 0 ) break;
                    peak_fit_fit();
                    break;
            default:
                    break;
            }
            gotoxy(1,1);
            printf("%80c",' ');
            gotoxy(1,1);
            printf("PEAK FITTING:  Limits  Peak_pos.  Shape_par.
                    Fit       ESC - quits");
}

/*--------------------------------------------*/

display_fit_menu();
}

double peak_value( int peak_number, int channel )
{
            double step( double x, double c, double s);
            double bump ( double x, double c, double s, double b);
            double gauss( double x, double center, double sigma );

            double y,x,c,s,b,g,u,e;

            x = (double) channel + 0.5 ;

            s = peak[0].sigma;
            b = peak[0].beta;
            u = peak[0].ba ;
            e = peak[0].ste ;
            g = peak[peak_number].ga;
            c = peak[peak_number].position;

            y = g * ( gauss(x,c,s) + u * bump(x,c,s,b) + e * step(x,c,s) ) ;

            return(y);

}

int check_everything()
{
            extern struct peak_type peak[];
        extern int n_of_p, fit_from, fit_to, N_MAX1,
                _from_x, _to_x;
        extern double b2, b1, b0;

        int i;

        if( peak[0].sigma <= 1. ){
                gotoxy(2,3);
       printf(" FWHM must be > 2.35           ");
                return(0);
        }

        if( n_of_p < 1 || n_of_p > 5 ){
                gotoxy(2,3);
                printf(" Number of peaks must be 1 - 5  ");
                return(0);
        }

        for( i = 0; i < n_of_p ; i++ )
                if( peak[i].position < fit_from || peak[i].position > fit_to){
                        gotoxy(2,3);
                        printf(" Wrong fit limits !!!!!!        ");
                        return(0);
                }

return(1);
}
void display_components()
{
        extern float *pointer_to_spf;
        extern double b0,b1,b2;
        extern int fit_from, fit_to;
        double step( double x, double c, double s);
        double bump ( double x, double c, double s, double b);
        double gauss( double x, double center, double sigma );
        void display_fit( float fit[], int fit_from, int fit_to );

        double A,B,C,c,x,s,b;
        int i,j;
        float *spf;

        spf = pointer_to_spf;

        for( i = fit_from ; i <= fit_to ; i++ ){
                x = (double)i + 0.5 ;
                *(spf + i) = b0 + b1 * x + b2 * x * x ;
        }

        setcolor(FIT_COLOR);
        display_fit(spf,fit_from,fit_to);

if( n_of_p == 1 ){

                c = peak[0].position;
                s = peak[0].sigma;
                b = peak[0].beta;
                A = peak[0].ga;
                B = peak[0].ba;
                C = peak[0].ste;
                for( i = fit_from ; i <= fit_to ; i++ ){
                        x = (double)i + 0.5 ;
                        *(spf + i) = b0 + b1 * x + b2 * x * x + A * gauss(x,c,s);
                }

                display_fit(spf,fit_from,fit_to);

                for( i = fit_from ; i <= fit_to ; i++ ){
                        x = (double)i + 0.5 ;
                        *(spf + i) = b0 + b1 * x + b2 * x * x + A * B * bump(x,c,s,b);
                }

                display_fit(spf,fit_from,fit_to);

                for( i = fit_from ; i <= fit_to ; i++ ){
                        x = (double)i + 0.5 ;
                        *(spf + i) = b0 + b1 * x + b2 * x * x + A * C * step(x,c,s);
                }

                display_fit(spf,fit_from,fit_to);
}
else{

                for (j = 0; j < n_of_p ; j++ ){

                        for( i = fit_from ; i <= fit_to ; i++ ){
                     x = (double)i + 0.5;
               *(spf + i) = b0 + b1 * x + b2 * x * x + peak_value(j,i)  ;
                        }
                        display_fit(spf,fit_from,fit_to);
                }

}

}
/*----------------------------PART06.C-------------------------------*/


#include <math.h>
#include <stdio.h>
#define ESC 27

extern struct peak_type {
                double position ;
                double sigma ;
                double beta ;
                double ga ;
                double ba ;
                double ste ; }     peak[5], tmp[5]  ;

double CHI2;

int message = 5;

void peak_fit_fit()
{
                extern int n_of_p, message;

                void peak_fit_fit_menu();
                void parallel_shift();
```

```
        void lin_fit_1();
        void lin_fit_n();
        void non_lin_fit_1();
        void non_lin_fit_n();
        void show_present_parameters();
        void save_parameters();
        void restore_parameters();
        int c,d;
        double tmp;


        peak_fit_fit_menu();
        lin_fit_n();
        show_present_parameters();

while( ( c = toupper( getch() ) ) != ESC ){

        switch(c){
        case 'T':
                if( n_of_p != 1) break;
                tmp = CHI2;
                save_parameters();
                lin_fit_1();
                if( CHI2 > (tmp * 1.001) ){
                        printf("try other limits or bgd");
                        CHI2 = tmp;
                        restore_parameters();
                        break;
                }
                non_lin_fit_1();
                break;
        case 'C':
                non_lin_fit_n();
                lin_fit_n();
                break;
        case 'R':
                gotoxy(1,2);
                printf("                    ");
                gotoxy(1,2);
                printf("sensitivity level = ");
                d = getche();
                if( d >= '0' && d <= '9') message = atoi(&d);
                gotoxy(1,2);
                printf("                    ");
                break;
        case 'P':
                parallel_shift();
                break;
        default:
                break;
        }

        peak_fit_fit_menu();
        show_present_parameters();
}
/*-----end of loop---*/
}

void show_present_parameters()
{
        extern int LINE, n_of_p, STATUS ;
        extern double a_calib, b_calib, CHI2;
        double en, gauss_fwhm, s, ga, ba, ste, c_of_g, cx, ab, ag, as, a, c;
        double fwhm, ymax, xmax;
        double factor = 2.5066; /*--- sq. root of 2Pi ---*/
        int i;
        void peak_parameters(double *xmxa, double *ymax, double *fwhm );
        void gotoxy_wt( int x, int y );

        s = peak[0].sigma;
        ba = peak[0].ba;
        ste = peak[0].ste;
        gauss_fwhm = 2.35 * s ;

        gotoxy_wt(1,LINE);
        if (STATUS) printf("[keV] ");
                else printf("[Chn] ");
        printf("fwhm=%.1f beta=%.1f ", gauss_fwhm * a_calib,
        peak[0].beta * a_calib);
        printf("bump=%.1f%% step=%.2f%%  ",100. * ba, 100. * ste);

   peak_parameters(&xmax,&ymax,&fwhm);
        xmax = xmax * a_calib + b_calib ;
        gotoxy(1,4);
        printf("chi2=%.1f  \n",CHI2);
        printf("%.1f %.0f %.2f",xmax,ymax * peak[0].ga,fwhm * a_calib);

        for( i = 0; i < n_of_p ; i++ ){
                gotoxy_wt(1,LINE - n_of_p + i);
                c = peak[i].position;
                en = c * a_calib + b_calib;
                printf("c%1d=%.1f ",i+1,en);
                bump_centroid_info(i, &cx, &ab );
                ga = peak[i].ga;
                cx *= ga * ba ;
                ab *= ga * ba ;
                ag = ga * factor * s ;
                a = ag + ab  ;
                c_of_g = ( cx + c * ag ) / a * a_calib + b_calib;
                as = c * ste * ga ;
                gotoxy(1,6 + i);
                printf("%1d)%.1f a=%.0f p/t=%.0f%%",
          i+1,c_of_g,a,a/(a+as)*100.);
        }
}

void peak_fit_fit_menu()
{
        extern int message;

        gotoxy(1,1);
        printf("%80c",' ');
        gotoxy(1,1);
        printf("FITTING MODES: Parallel, Centroid, Train, Restore(%d)
                 ESC - quits", message);
}

void lin_fit_1()
{
        extern float *pointer_to_sp;
```

```
            float *sp;
            extern int fit_from, fit_to;
            extern double b0,b1,b2;
            double S[3][4], tmp[3] ;
            double dim34( double S[][] );
            double gauss( double x, double c, double s );
            double bump( double x, double c, double s, double b );
            double step( double x, double c, double s );
            double chi2();

            double x,y,c,s,b,g,u,e,t,A,B,C,D,a;
            int i,j;

            gotoxy(1,2);
            printf("Wait !!!    ");

            sp = pointer_to_sp;

            s = peak[0].sigma;
/*---------------------------------------*/
if( peak[0].beta <= 0. ) peak[0].beta = 3. * s ;
/*---------------------------------------*/
            b = peak[0].beta;
            c = peak[0].position;


            for( i = 0; i < 3 ; i++){
                    S[i][3] = 0.;
                    for( j = i; j < 3 ; j++ )
                            S[i][j] = 0.;
            }


            for( i = fit_from; i <= fit_to; i++ ){

            x = (double)i + 0.5;
            y = (double)( *(sp + i) );
            t = y - ( b0 + b1 * x + b2 * x * x ) ;
            g = gauss(x,c,s);
            u = bump(x,c,s,b);
            e = step(x,c,s);

            S[0][0] += g * g ;
            S[0][1] += u * g ;
            S[0][2] += e * g ;
            S[0][3] += t * g ;

            S[1][1] += u * u ;
            S[1][2] += e * u ;
            S[1][3] += t * u ;

            S[2][2] += e * e ;
            S[2][3] += t * e ;
     }

            S[1][0] = S[0][1];
            S[2][0] = S[0][2];
            S[2][1] = S[1][2];

     D = dim34(S);
```

```
if( D*D < .0000001 ){
printf(" D = 0 ");
CHI2 += 1000.;
return;}

                for( i = 0; i < 3; i++ ) {
                tmp[i] = S[i][0];
                S[i][0] = S[i][3];
                }

                A = dim34(S);
if( A*A < .0000001 ){
printf(" A = 0 ");
CHI2 += 1000.;
return;}

                for( i = 0; i < 3; i++ ) {
                S[i][0] = tmp[i];
                tmp[i] = S[i][1];
                S[i][1] = S[i][3];
                }

                B = dim34(S);

                for( i = 0; i < 3; i++ ) {
                S[i][1] = tmp[i];
                S[i][2] = S[i][3];
                }

                C = dim34(S);

                a = A / D;
                if ( a <= 0. ) {
printf("Negative peak");
CHI2 += 1000.;
return;
                }

                b = B / A;
                if(b < 0.){
printf("Negative bump");
CHI2 += 1000.;
return;
                }

                c = C / A;
                if(c < 0.){
printf("Negative step");
c = 0.;
                }

                peak[0].ga = a ;
                peak[0].ba = b ;
                peak[0].ste = c ;

                gotoxy(1,2);
                CHI2 = chi2();
                printf("Chi2 = %.1f \n",CHI2 );
}
```

```
double dim34( double S[3][4] )
{
                double y ;

                y = S[0][0] * S[1][1] * S[2][2]
                  + S[0][1] * S[1][2] * S[2][0]
        + S[1][0] * S[2][1] * S[0][2]
                  - S[2][0] * S[1][1] * S[0][2]
                  - S[1][0] * S[0][1] * S[2][2]
                  - S[2][1] * S[1][2] * S[0][0] ;

                return(y);

}

double chi2()
{
                extern int fit_from, fit_to, n_of_p;
                extern double b0, b1, b2 ;
                extern float *pointer_to_sp;
                int i,j;
                double c = 0., t, x, y ;
                double peak_value( int peak_number, int channel );


                for(i = fit_from; i <= fit_to; i++){
                                x = (double)i + 0.5;
                                y = (double)( *(pointer_to_sp + i) ) ;
             t = b0 + b1 * x + b2 * x * x ;
                      for(j = 0; j < n_of_p; j++ )
                                               t += peak_value(j,i);
                                t = y - t ;
                                c +=  t * t / ( y + 1 ) ;    /* y=sqrt(n)^2; +1 to avoid/0 */
                }
                c = c / (double)( fit_to - fit_from + 1 );

return(c);
}

void non_lin_fit_1()
{

                void search_1( double *what, double *how );
                static double ds , db , dc ;
                extern int message;
                extern double ds_minimum, db_minimum, dc_minimum, CHI2;
                int n,i;
                double factor, prog1, prog2, prog3;

                if(message){
                                factor = pow(2.,(double) message);
                                ds = ds_minimum * factor;
                                db = db_minimum * factor;
                                dc = dc_minimum * factor;
                }
gotoxy(1,2);
printf("            ");
gotoxy(1,2);
printf("n = ");
scanf("%d",&n);
```

```
            for( i = 0; i < n ; i++ ){
                        prog1 = CHI2;
                        if ( dc * dc > dc_minimum * dc_minimum * 0.999  )
                                                search_1(&peak[0].position, &dc);
                        prog1 /= CHI2;

                        prog2 = CHI2;
                        if ( ds * ds > ds_minimum * ds_minimum * 0.999 )
                                                search_1(&peak[0].sigma, &ds);
                        prog2 /= CHI2;

                        prog3 = CHI2;
                        if ( db * db > db_minimum * db_minimum * 0.999   )
                                                search_1(&peak[0].beta, &db);
                        prog3 /= CHI2;

gotoxy(1,3);
printf("dc*%.0f ds*%.0f db*%.0f %.2f %.2f %.2f ",
dc/dc_minimum,ds/ds_minimum,db/db_minimum,prog1,prog2,prog3);
            }

}

void lin_fit_n()
{
            extern float *pointer_to_sp;
            float *sp;
            extern int fit_from, fit_to, n_of_p;
            extern double b0,b1,b2;
            double S[30], tmp[5] ;
            double det( double S[] );
            double gauss( double x, double c, double s );
            double bump( double x, double c, double s, double b );
            double step( double x, double c, double s );
            double peak_value(int peak_number, int channel );
            double chi2();

            double x,t,z,D;
            int i,j,k;

    gotoxy(1,2);
            printf("Wait !!!    ");

            sp = pointer_to_sp;

            for( i = 0 ; i <  n_of_p  ; i++ ){
                        peak[i].ga = 1. ;
                        for ( j = i ; j <= n_of_p ; j++ )
                                    S[ i * ( n_of_p + 1 ) + j ] = 0. ;
            }

            for( k = fit_from ; k <= fit_to ; k++ ){
                        x = (double) k + 0.5 ;
                        t = (double) *(sp + k) - b0 - b1 * x - b2 * x * x ;
                        for( i = 0 ; i <  n_of_p  ; i++ ){
                                    z = peak_value(i,k);
                                    for ( j = i ; j < n_of_p ; j++ )
                                    S[ i * ( n_of_p + 1 ) + j ] += z * peak_value(j,k) ;
                                    S[ i * ( n_of_p + 1 ) + n_of_p ] += z * t ;
```

```
                }
        }

        for( i = 1 ; i <  n_of_p  ; i++ )
                for ( j = 0 ; j < i ; j++ )
                S[ i * ( n_of_p + 1 ) + j ] = S[ j * ( n_of_p + 1 ) + i ] ;

/*----------------------------------------------------------------
for(i = 0 ; i < n_of_p ; i++){
gotoxy(1,2+i);
for(j = 0; j <= n_of_p ; j++)printf(" %.1f ",S[i*(n_of_p + 1) + j]);
}
getch();
----------------------------------------------------------------*/

        D = det(S);

if( D*D < 0.00000000001 ){
        printf(" D = 0 ");
        return;
}

        for ( j = 0 ; j < n_of_p ; j++ ){
                for( i = 0 ; i < n_of_p ; i++ ){
                tmp[i] = S[ i * ( n_of_p + 1) + j ];
                S[ i * ( n_of_p + 1) + j ] = S[ i * ( n_of_p + 1 ) + n_of_p ];
                }
                peak[j].ga = det(S) / D;
                for( i = 0 ; i < n_of_p ; i++ )
                        S[ i * ( n_of_p + 1) + j ] = tmp[i] ;
        }

        gotoxy(1,2);
        CHI2 = chi2();
        printf("     Chi2 = %.1f  \n",CHI2 );

}

double det(double s[])
{
        extern int n_of_p;
        double y;

   switch(n_of_p){
        case 1 :
                return( *s );
        case 2 :
                y = *(s + 0) * *(s + 4) - *(s + 3) * *(s + 1);
                return(y);
        case 3 :
                y = *(s + 0) * *(s + 5) * *(s + 10)
                  + *(s + 1) * *(s + 6) * *(s + 8)
                  + *(s + 4) * *(s + 9) * *(s + 2)
                  - *(s + 8) * *(s + 5) * *(s + 2)
                  - *(s + 4) * *(s + 1) * *(s + 10)
                  - *(s + 9) * *(s + 6) * *(s + 0);
                return(y);
        default:
                return(0.);
   }
```

```
}

void non_lin_fit_n()
{
            void lin_fit_n();
            void search_n( double *what, double *how );
            static double dc[5] = {1.,1.,1.,1.,1.} ;
            extern int message,n_of_p;
            extern double dc_minimum;
            int n,i,j;
            double factor;

            if(message){
                        factor = pow(2.,(double) message);
                        dc[0] = dc_minimum * factor;
                        dc[1] = dc_minimum * factor;
                        dc[2] = dc_minimum * factor;
                        dc[3] = dc_minimum * factor;
                        dc[4] = dc_minimum * factor;
            }
gotoxy(1,2);
printf("            ");
gotoxy(1,2);
printf("n = ");
scanf("%d",&n);
if( n > 20 )  n = 20 ;

            for( i = 0; i < n ; i++ ){
                        for(j = 0 ; j < n_of_p ; j++)
                                    if( dc[j] * dc[j] > dc_minimum * dc_minimum * 0.99999 )
                                                search_n(&peak[j].position, &dc[j]);

gotoxy(1,3);
for(j = 0 ; j < n_of_p ; j++ )
printf(" %.3f ",dc[j]);

            }

}


void parallel_shift()
{
            void lin_fit_n();
            void search_n( double *what, double *how );
            void search_parallel(double *how);
            static double dc = 1. ;
            extern int message,n_of_p;
            extern double dc_minimum;
            int n,i,j;
            double factor;

            if(message){
                        factor = pow(2.,(double) message);
                        dc = dc_minimum * factor;
            }
gotoxy(1,2);
printf("            ");
gotoxy(1,2);
printf("n = ");
```

```
scanf("%d",&n);
if( n > 20 )  n = 20 ;

        for( i = 0; i < n ; i++ ){
                                if( dc * dc > dc_minimum * dc_minimum * 0.99999 )
                                        search_parallel( &dc);

gotoxy(1,3);
printf(" %.3f ",dc);

        }

}


void search_1( double *what, double *how )
{
        extern double CHI2;
        double c1,c2,c3,a,b,x;
        void save_parameters();
        void restore_parameters();

        c1 = CHI2;
        save_parameters();
        *what += *how;
        lin_fit_1();
        c2 = CHI2;
        if( c2 <= c1 ) return;
        if( c2 > c1 ){
                *how *= -1. ;
                *what += 2. * *how;
                lin_fit_1() ;
      c3 = CHI2 ;
                if( c3 <= c1 ) return;
                else            {
                        *what -= *how ;
                        *how *= -1.;
                        a = c2 / 2. + c3 / 2. - c1 ;
                        b = ( c2 - c3 - 4.* *how * a * *what ) / 2. / *how;
                        x = *what;
                        *what = -b / 2. / a ;
                        lin_fit_1();
                        if( CHI2 > c1 ){
                                *what = x;
                                restore_parameters();
                                CHI2 = c1;
                        }
                        *how /= 2. ;
                        return;
                }
   }

}

void search_n( double *what, double *how )
{
        extern double CHI2;
        double c1,c2,c3,a,b,x;
        void save_parameters();
        void restore_parameters();
        c1 = CHI2;
        save_parameters();
        *what += *how;
        lin_fit_n();
        c2 = CHI2;
        if( c2 <= c1 ) return;
        if( c2 > c1 ){
                *how *= -1. ;
                *what += 2. * *how;
                lin_fit_n() ;
       c3 = CHI2 ;
                if( c3 <= c1 ) return;
                else            {
                        *what -= *how ;
                        *how *= -1.;
                        a = c2 / 2. + c3 / 2. - c1 ;
                        b = ( c2 - c3 - 4.* *how * a * *what ) / 2. / *how;
                        x = *what;
                        *what = -b / 2. / a ;
                        lin_fit_n();
                        if( CHI2 > c1 ){
                                *what = x;
                                restore_parameters();
                                CHI2 = c1;
                        }
                        *how /= 2. ;
                        return;
                }
    }

}


void search_parallel( double *how )
{
        extern double CHI2;
        double c1,c2,c3,a,b,x[5];
        void save_parameters();
        void restore_parameters();
        int i;

        c1 = CHI2;
        save_parameters();
for(i = 0; i < n_of_p; i++) peak[i].position += *how;
        lin_fit_n();
        c2 = CHI2;
        if( c2 <= c1 ) return;

                *how *= -1. ;
for(i = 0; i < n_of_p; i++) peak[i].position += 2. * *how;
                lin_fit_n() ;
       c3 = CHI2 ;
                if( c3 <= c1 ) return;
                else            {
                                restore_parameters();
                                CHI2 = c1;
                                *how /= 2. ;
                        }
```

```
                                    return;

}


void save_parameters()
{
    extern int n_of_p;
            int i;

    tmp[0].sigma = peak[0].sigma;
            tmp[0].beta = peak[0].beta;
            tmp[0].ga = peak[0].ga;
            tmp[0].ba = peak[0].ba;
            tmp[0].ste = peak[0].ste;


            for(i = 0; i < n_of_p; i++)
                        tmp[i].position = peak[i].position;

}

void restore_parameters()
{
    extern int n_of_p;
            int i;

            peak[0].sigma = tmp[0].sigma;
            peak[0].beta = tmp[0].beta;
            peak[0].ga = tmp[0].ga;
            peak[0].ba = tmp[0].ba;
            peak[0].ste = tmp[0].ste;


            for(i = 0; i < n_of_p; i++)
                        peak[i].position = tmp[i].position;

}
```