課題研究P6
μ−PICを用いたエネルギー損失測定

上野一樹　水上拓　八登浩紀

2005/3/31

目 次

1　はじめに　～実験でやろうとしたこと～

1.1　Abstract

未知の放射線を同定するということは高エネルギー宇宙物理学だけではなく、放射線物理学、核医学、考古学など様々な分野においてもたくさんの情報を得られるという点で有用なことである。そして、この放射線を同定するにはこれまで色々な方法が試されてきた。本実験ではそのうち、荷電粒子のエネルギー損失の分布を用いて放射線を同定することを試みた。特に、今回は簡単のため、既知の放射線源（α線源）を用いて実際に同定できるのかを確かめることを目標とした。

1.2　実験の流れ

エネルギー損失を測定するには、検出器内に入ってきた荷電粒子が検出器内でどれだけ走り、そしてその間にどれだけのエネルギーを落としたかを同時に知ることができればよい。つまり、入射荷電粒子の飛跡とそのエネルギー情報を同時に得られればよい。これが可能な検出器として、μ−PICという検出器（図1）がある（詳しくは次のsectionで説明する）。ということで、まずはこの検出器を用いて実際に解析できることを確かめた。その後にα線源としてマントル（図2）を用いて実際にエネルギー損失を求めることにした。マントルとは、ランタンの発光体で少量のトリウムが添加されている。このトリウムの放射性壊変に伴いα線やβ線を放出する。トリウム系列を図3に示しておく。そして、この結果とシミュレーションを行いその結果とを比較することにより同定を行うという流れである。

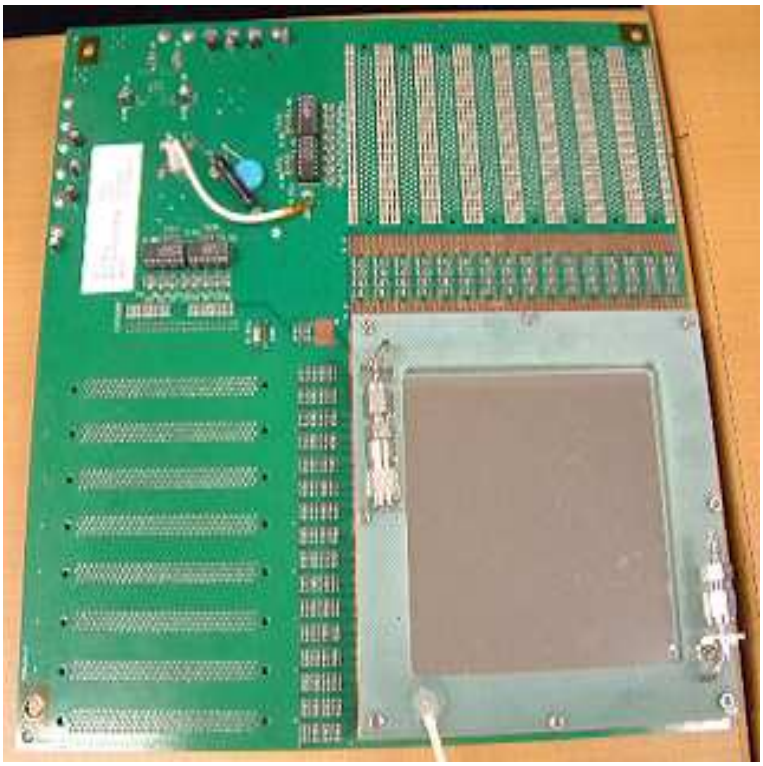

図 1: μ−PIC
大きさは 30cm × 30cm で、右下の 10cm × 10cm の部分が検出面

図 2: マントル

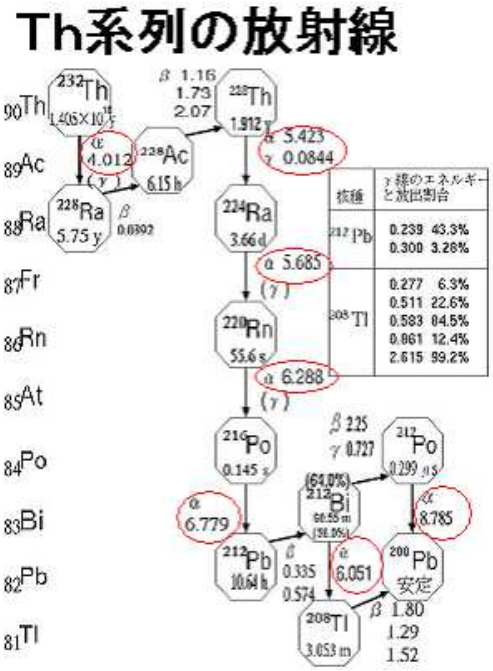

図 3: トリウム系列

2 μ−PIC

本実験で使用する検出器μ−PICは現在、京都大学宇宙線研究室で開発中のピクセル型ガス検出器である。構造は図4のようにドリフト領域と比例計数管を輪切りにして並べたような電離領域からなり、荷電粒子がドリフト領域中を走ると電離ガスを電離し多くの電子と陽イオンを作る。電子はドリフト電場によりAnodeへと動いていき、Anodeにかけられた高電圧により電子なだれを起こし増幅される。そしてAnode側、Cathode側からの信号をそれぞれ読み出す。実験で使用したμ−PICは10cm四方でデジタルではAnode側、Cathode側それぞれ256chの読み出しを、アナログではそれを32chずつsumして8chの読み出しが出来る。μ−PICの特徴として、放電に強く安定して高い増幅率を得られることと、高い位置分解能があげられる。このμ−PICを実際に動かしてみて理解を深めることが本実験のもうひとつの目的である。

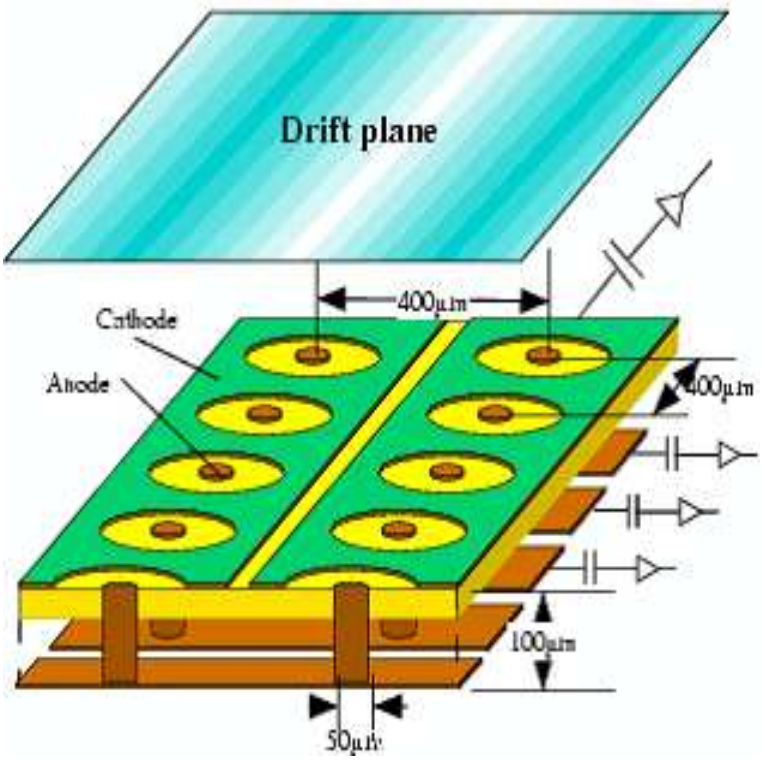

図 4: μ−PICの構造

3　μ−PIC の増幅率

検出器としての μ−PIC を使用するために、まず増幅率を測定した。

3.1　ガス比の決定

図 5 のように μ−PIC からの信号を PreAmp を通し、オシロスコープで PulseHeight を測定した。

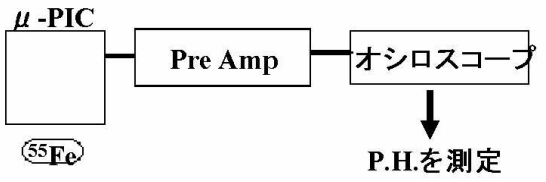

図 5: μ−PIC からの信号をオシロスコープで測定

ここで用いた PreAmp は μ−PIC からの信号を読み出す専用の ASD（Amplifier Shaper Discriminator）であり、アナログ出力は通常の Amp と同様に入力信号を増幅して出力し、またデジタル出力は、設定した閾値を超える信号が入力されたときに信号が入ったというデジタルの信号を出力する。この ASD には時定数 80nsec の CXA3653Q(新 ASD) と 16nsec の CXA3183Q(旧 ASD) の２種類がある。

混合ガスとして Ar と CO_2 を用い、ガス比を変化させ、それぞれのガス比での HV と Pulse Height の関係を調べると図 6 が得られた。

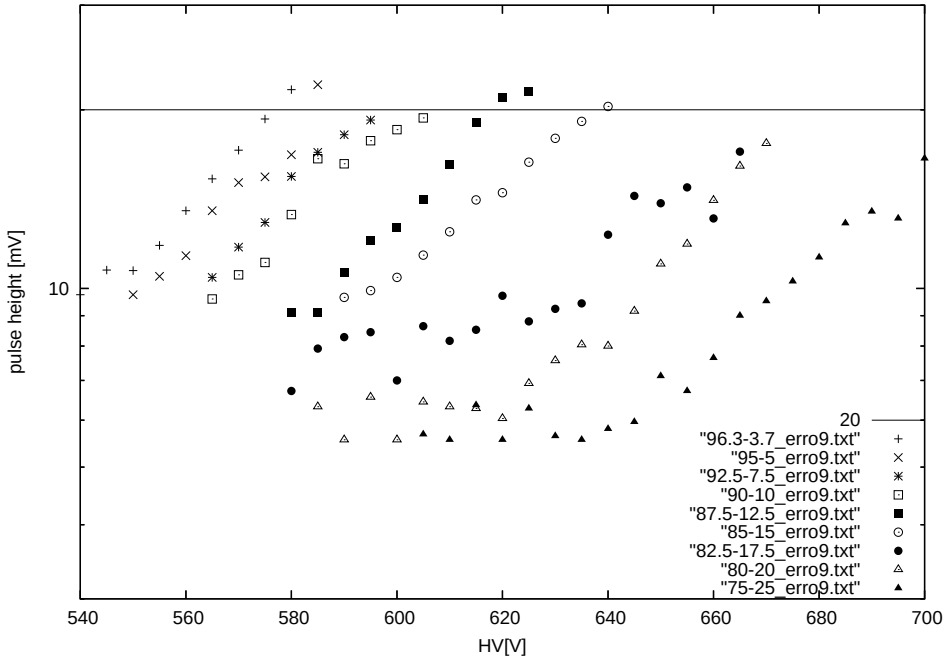

図 6: ガス比ごとの HV と Pulse Height の関係

この図から消滅ガスの CO_2 を 15 %程度まで入れても得られる最大の増幅率はそれほど変わらないが、それ以上入れると増幅率が落ちることが分かった。従ってこれ以降の増幅率測定では Ar:CO_2 =85:15 のガス比で行った。

3.2 ASD の増幅率

前節で測定した波高は μ−PIC からの信号を ASD で増幅した後に測定したものであった。従って μ−PIC 自身の検出器としての増幅率を見るためには ASD の増幅分を取り除かなければならない。ここでは ASD の増幅率を求める。

先に述べた新 ASD と旧 ASD は時定数が異なるので、入力される信号の時定数によって二つの ASD の増幅率の差も異なる。このことを利用して μ−PIC 自体から ASD に入力される信号に対する ASD の増幅率を求

める。

実験装置は図 7 のように Function Generator から矩形波を微分回路に入力して微分回路から時定数を変えたパルスを ASD に入れて増幅させたものをオシロスコープで測定する。これで入力した電荷量に対してどの位の PulseHeight を持つ信号を ASD が返すのか [mV/pC] を求めることができる。

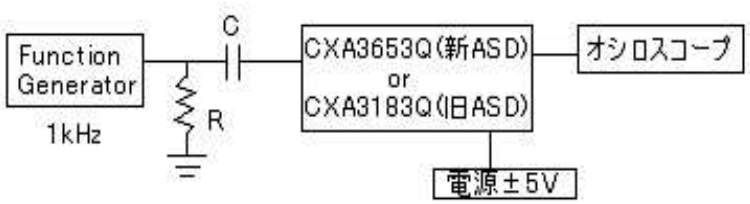

図 7: ASD の増幅率測定のための実験装置

図 8、図 9 はファンクションジェネレータ (FG) からの出力信号と、微分回路に C=330pF、R=50Ω を用いた時の微分回路からの出力信号である。

この測定で得られた新旧 ASD の増幅率がそれぞれ図 10、図 11 である。

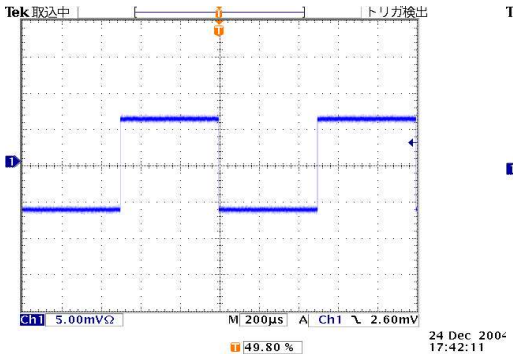

図 8: FG からの信号

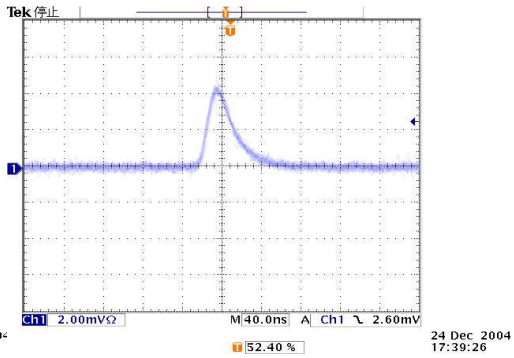

図 9: 出力信号

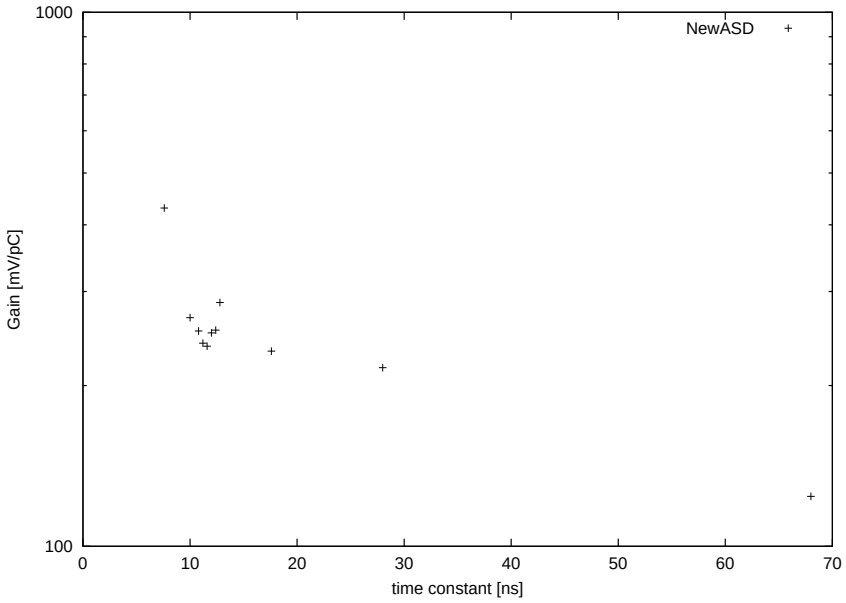

図 10: 新 ASD の増幅率

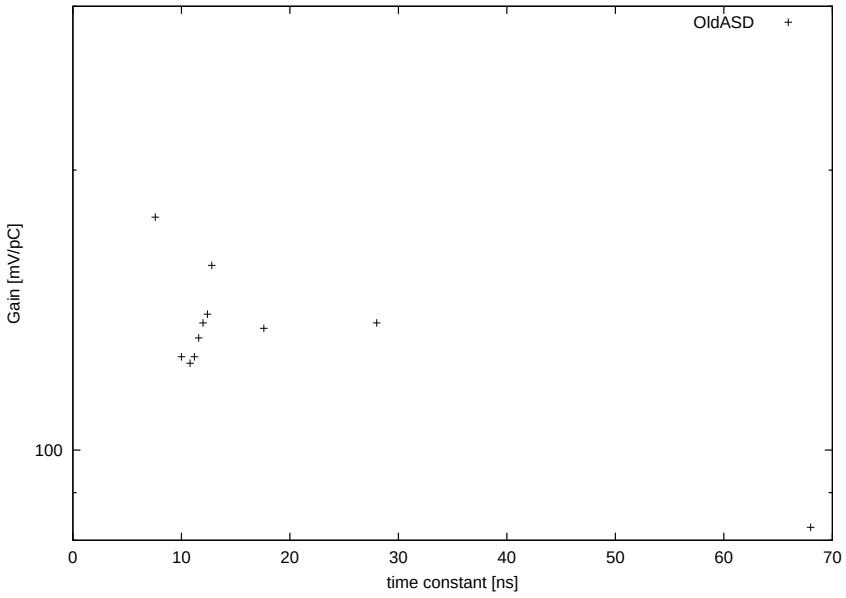

図 11: 旧 ASD の増幅率

これから新 ASD/旧 ASD の増幅率の比を取ったものが図 12 である。

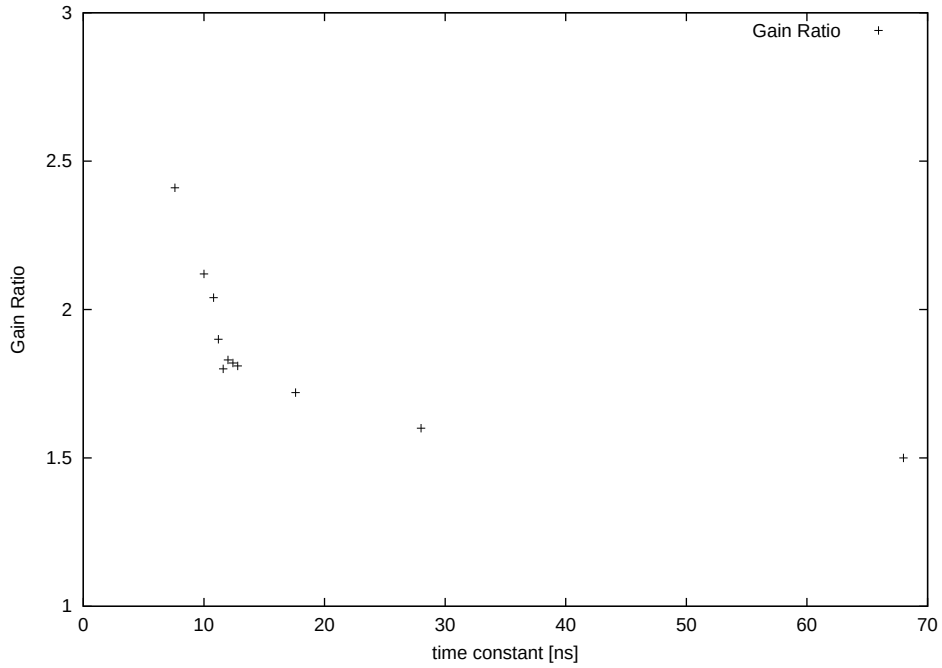

図 12: 新旧 ASD の増幅率の比（新 ASD/旧 ASD）

この図から測定した時定数の範囲では新 ASD の方が増幅率が高いことが分かる。

次に μ−PIC に新旧両 ASD を接続し出てくる信号の波高を測定しその比を取る。ASD に入る信号の電荷量は同じなので、この比はそのまま増幅率の比となるので図から μ−PIC から出てきた信号の時定数を知ることができる。測定から増幅率の差は 1.89 であることが分かった。図から μ−PIC からのパルスの時定数は 11.6nsec となり、この時定数のパルスに対する新旧 ASD の増幅率はそれぞれ 232.6[mV/pC]、123.1[mV/pC] と決定できた。

3.3 μ−PIC の増幅率

前節までで ASD の増幅率が求められたので、

$\mu-$PIC の増幅率 = ($\mu-$PIC+ASD) の増幅率 / ASD の増幅率
によって $\mu-$PIC の増幅率が求められる。用いた線源の ^{55}Fe からの主な放射線の Energy は 5.9keV であり、測定に用いた Ar:CO_2 =85:15 の混合ガスの W 値は 27eV であるので、電離される電子の個数はおよそ 220 個である。つまり、最初に電離される電荷量は 3.52×10^{-5}[pC] である。すると、Pulse Height が 20mV の時の ($\mu-$PIC+ASD) の増幅率は、5.7×10^{5}[mV/pC] である。新 ASD の増幅率は 232.6[mV/pC] なので、$\mu-$PIC 自身の増幅率は、$5.7\times10^{5}/232.6 = 2450$ となる。

HV と増幅率の関係を描くと図 13 のようになる。

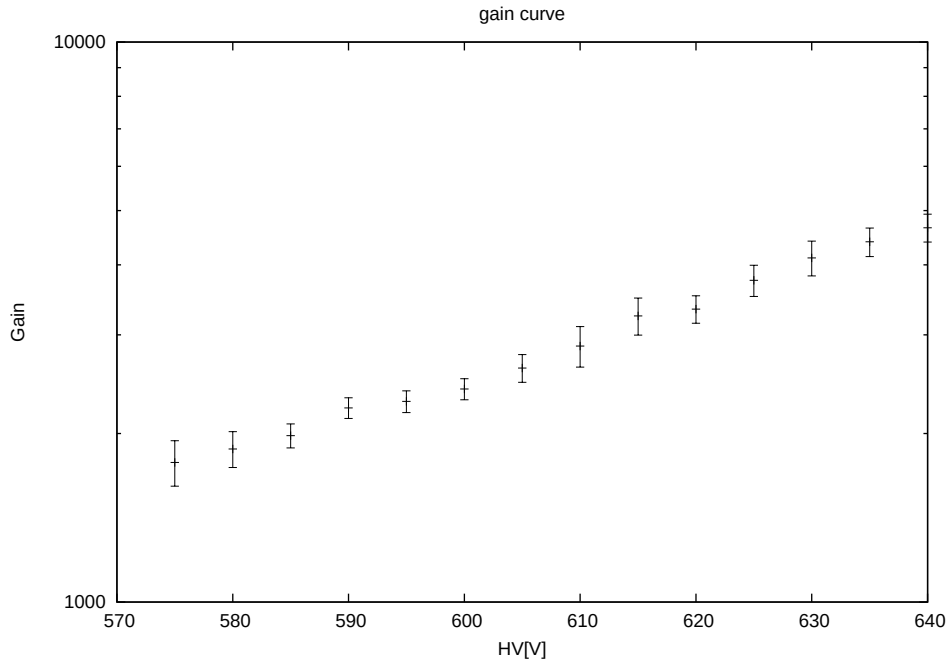

図 13: $\mu-$PIC の HV と増幅率の関係

増幅率は最大で 4600 程度であることが分かった。

3.4 Gain Map

つぎに μ–PIC の検出する領域の違いによる増幅率のバラつきを調べる。実験装置は図 14 である。

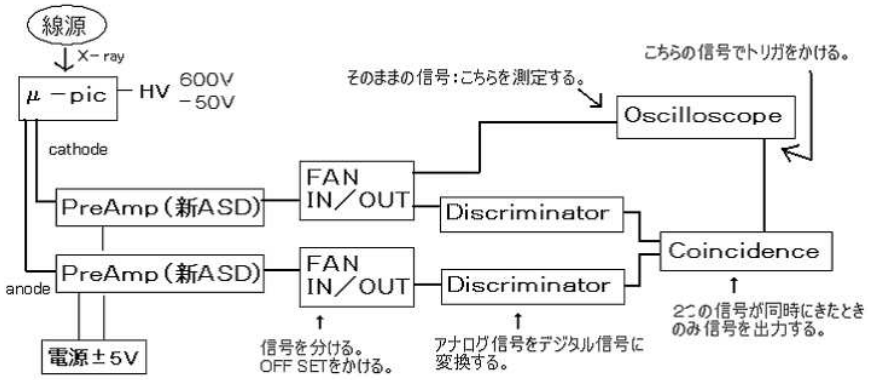

図 14: GainMap 測定用の実験装置

Anode 側、Cathode 側の Amp からの信号のコインシデンスをとることにより、オシロスコープで測定される信号は検出面の 1.25cm 四方の領域に入った信号となる。Amp で見る位置を変えることで 5cm 四方の領域を 16 個の領域に分けてそれぞれの領域での増幅率を調べる。

この測定の結果図 15 のような増幅率の分布（GainMap）が得られた。

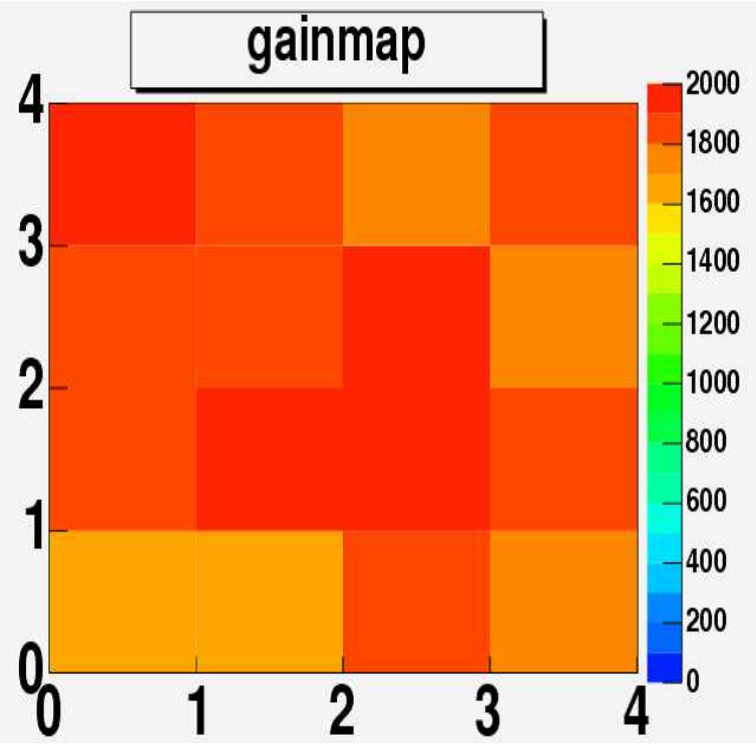

図 15: μ−PIC の Gain Map

これから増幅率は RMS で 7 %程度のばらつきで μ−PIC は検出器として正しく使えることが分かった。

4 Spectroscopy

前節により、μ−PICが検出器として正しく使えていることは分かった。ということで、この節では実際にμ−PICを用いて、目的の情報のうちの一つであるエネルギーを測定することを試みた。

4.1 実験原理

この測定では、増幅率測定の時と同様アナログ信号を取得して解析したが、信号を取得する手段としてFlash ADC（以下FADC）を用いた。このFADCはVMEを介してPCで制御でき、データをPCに取り込むことができる。今回用いたFADCは100MHzのclock数でデータを取得する。つまり、10ns毎に信号の高さを読み取り（あらかじめ、ベースラインは定まっており、それぞれ0〜255channelのいずれかに割り振られる)、デジタル信号へ変換する。このFADCを用いた実験装置配線図は図16のようである。

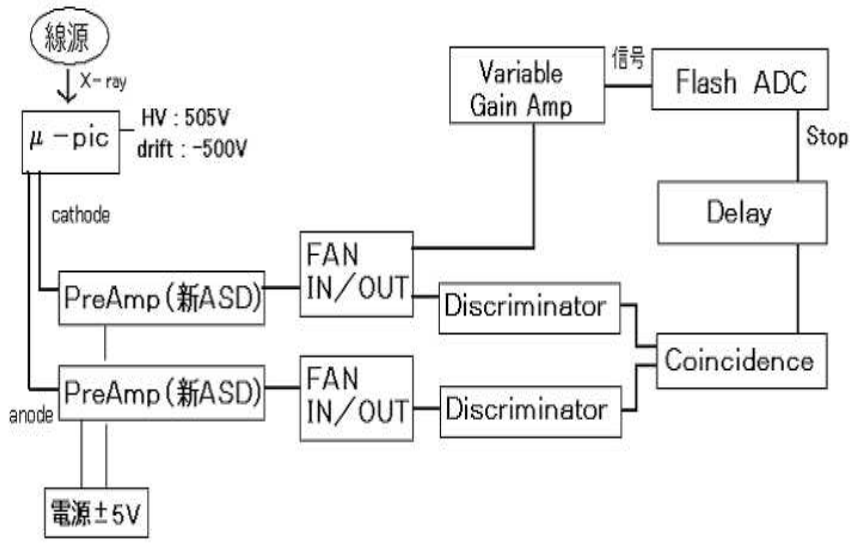

図 16: 実験装置配線図

これを用いた測定原理であるが、次のとおりである。放射線をμ−PIC

が検出し、FADC よりその波形データを取得する。このデータの最大値を何度もとることでスペクトルは得られるはずである。しかし、今回はその方法は用いなかった。なぜなら得られた波形データを見るとベースラインに少しずつではあるが違いがあったのである。よってこのゆらぎの効果をなくすために得られた波形データの平均値を求め、その値を閾値とし、それを超えたもののみを積分するという方法をとった (図 17 参照)。これにより、各イベントごとにそのエネルギーを求めることができる。この積分値を PC に出力し、その結果を 1 次元ヒストグラムとして表すことによりスペクトルが得られる。(プログラムは最後にまとめておく)

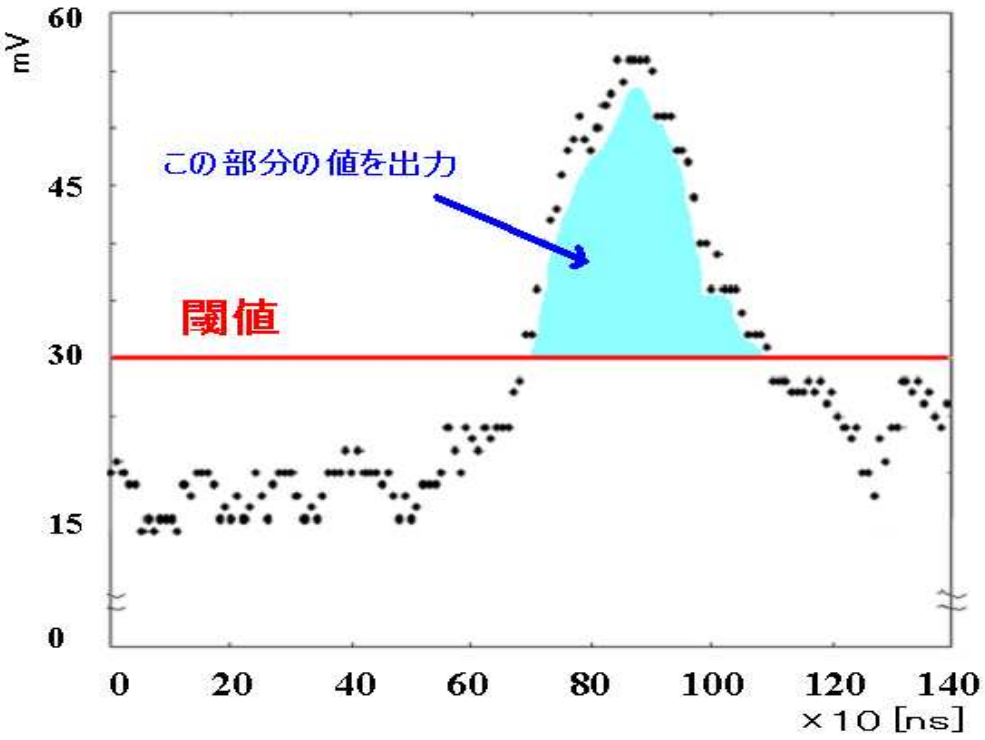

図 17: データ取得

4.2 測定

測定は図 16 のような setup を組んで行った。試料線源としては ^{55}Fe と ^{109}Cd を用いた。Gas Package は 1cm のものを用いた。あまり薄いものを

使用すると Package 内で光電子が全てエネルギーを落とす前に検出器外に出てしまうことがあるからである。ガスはエスケープピークが見えるということを考慮して Ar を用いた。エスケープピークについて少し説明を加えておく。Ar ガスにX線が光電吸収されると、Ar 原子から電子が放出されるが、ある確率でガンマ線（Ar-K 輝線) が放出される。このガンマ線はガス中で再吸収され結局はガスを電離するのであるが、一部の輝線が吸収されることなく外に抜け出してしまうことがある。その結果その輝線の分（Ar では 3.2[keV]）少ないピークがたつのである。このピークのことをエスケープピークとよぶのである。消滅ガスには C_2H_6 を用いた。割合は Ar:C_2H_6=9:1 とした。また、今回は cathode 側、anode 側それぞれに 1 枚ずつ ASD をさし、1.25cm 四方の領域に対しての測定を行った。

4.3 結果

上のような条件で得たスペクトルは図 18 のようになった。ただし、×印が ^{55}Fe、〇印が ^{109}Cd の線源を置いたときの結果である。

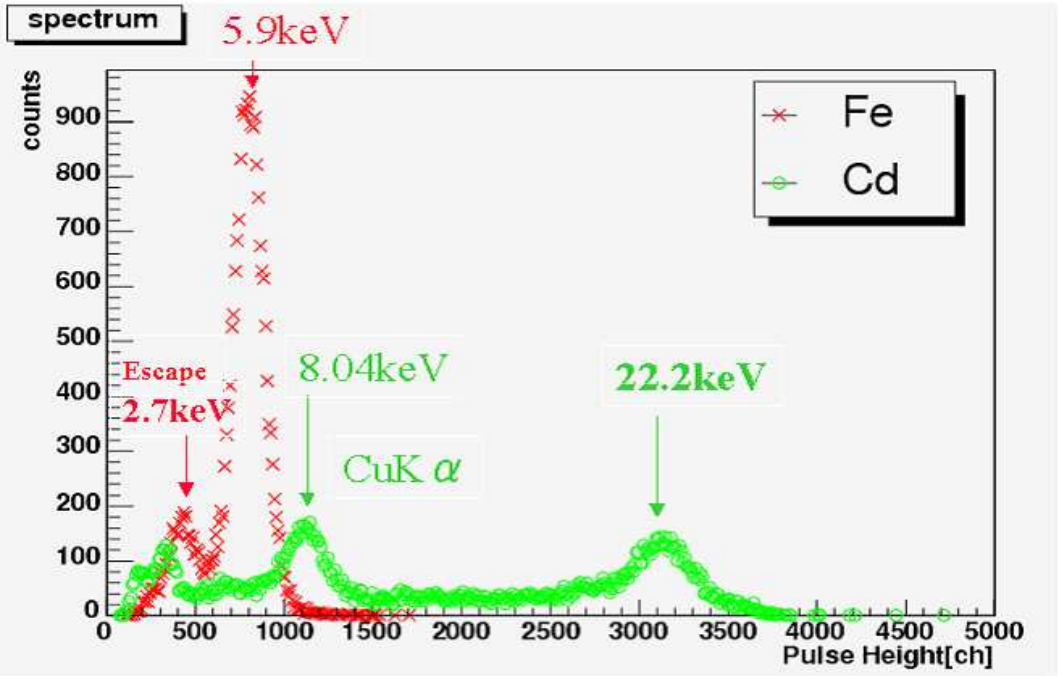

図 18: スペクトル

これより、^{55}Fe の方を見ると、Ar によるエスケープピークがちゃんと見えているようである。また、^{109}Cd の方においては、μ−PIC の基盤に用いられている Cu による特性X線も見えているようである。これらよりｃｈからエネルギーへの校正を行った。

具体的には、ch=a×(Energy)+b　（ただし a、b は定数）に最小二乗法を用いて fitting をした。その結果、a=131.6±4.500 ,b=-9.164±31.48 という結果になった。（図 19 参照）

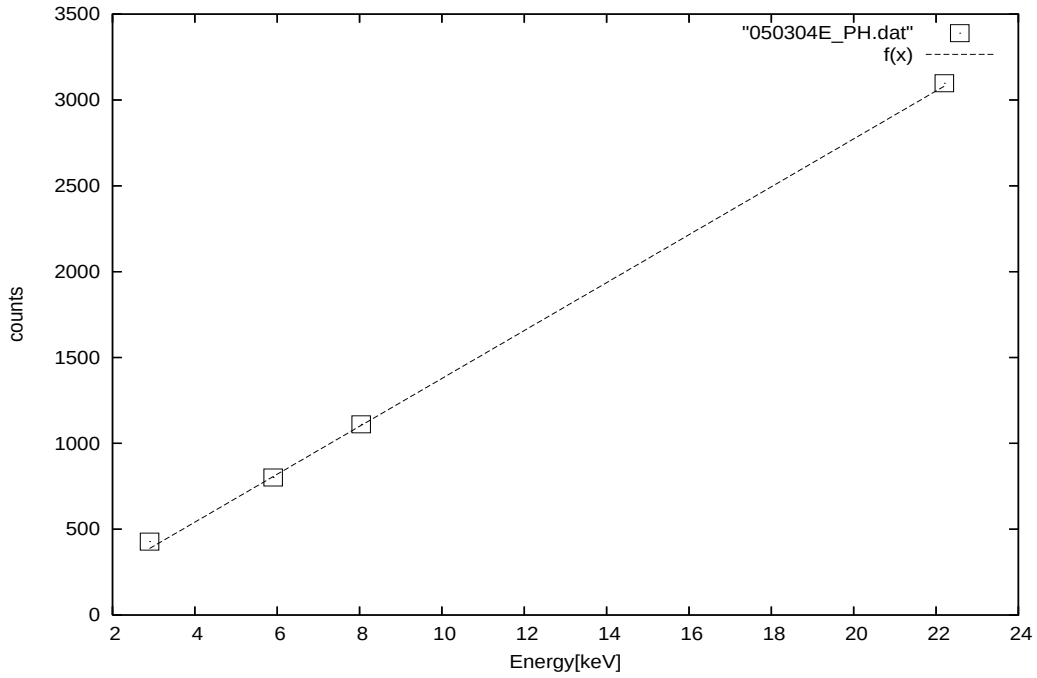

図 19: calibration

これを用いて横軸をエネルギーに変えたものが図 20 である。

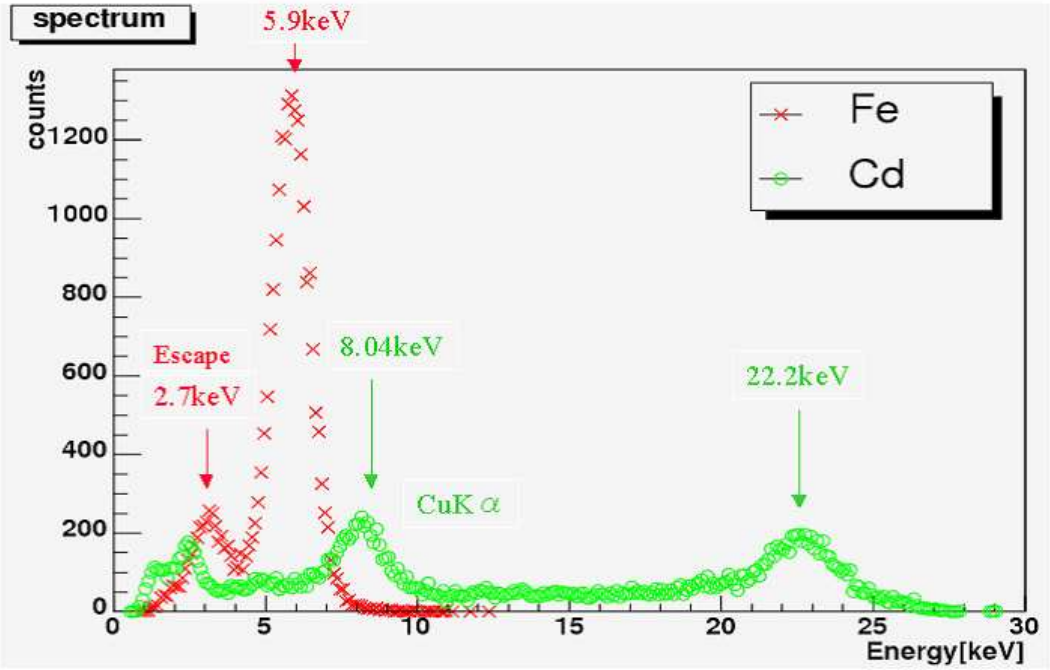

図 20: スペクトル

またエネルギー分解能を計算すると、FWHM で、^{55}Fe の 5.9[keV] に対し 24.7 %、^{109}Cd の 22.2[keV] に対し 16.6 %という結果が得られた。

これらの結果より、線形性が保たれている上に、過去に報告されているエネルギー分解能と今回のものを比べても数%の違いしかなく妥当な結果だと言える。これより、今回用いた解析方法でエネルギーを測定することができるということがわかった。

5　2-D imaging

μ−PIC は高い位置分解能を持った検出器であり、その公称値は 100μm に達する。すると、μ−PIC を用いて２次元のイメージをかなり正確に取得することができるということである。そこで、実際にテストチャートを用いて、２次元のイメージを取得することを試みた。

5.1　実験原理

装置の配線は図 21 のようにした。今までの増幅率測定や Spectroscopy ではアナログ信号を用いて測定を行っていたが、本測定の特徴は ASD から出力されるデジタル信号を利用する点である。

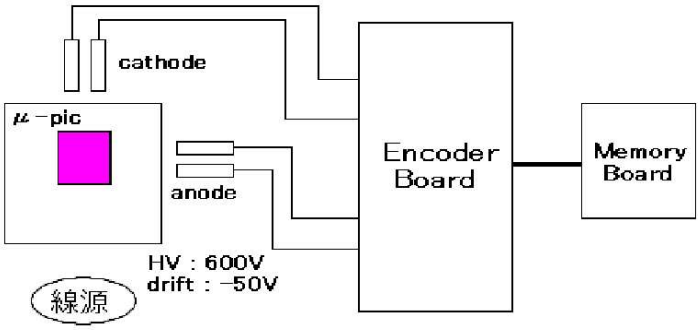

図 21: 2-D image 検出のための配線図

その測定原理は次のとおりである。まず、Gas Package 内に入った放射線により電離された電子は、ドリフトする過程で増幅された後、信号として anode、cathode 双方の ASD に入力される。ここで、ひとつの ASD は 32ch 持っており、チャンネル毎に受け持つ領域に入った信号を別々に処理できる。一方 ASD は、その信号が設定した閾値を超えた場合にデジタル信号、パルスを出力するようになっている。つまり、ひとつの ASD は各領域に閾値を超える信号が入ったかどうかの 32ch のデジタル信号を出力できる。

その anode、cathode 双方より出力された信号は、Encoder Board に入力される。この Encoder Board は、FPGA、つまり信号をどのように処

理するかのプログラムをPCから自由にダウンロードして変更できるものを用いている。本測定で使用したプログラムは、anode、cathodeで同時に信号が検出された場合をとらえて、信号が発生したと思われる領域を(X,Y)の２次元情報として出力するものである。

この信号はVMEに接続したMemory Boardに蓄積され、PCから取り出される。（プログラムは後ろに掲載）

5.2 測定

測定は図22、23のようなsetupを組んで行った。検出面のすぐ前に鉛製のテストチャートを置き、少し離れた位置に固定した線源から放射線を照射した。用いた線源は^{109}Cdである。Gas Packageは1mmのものを用いた。薄いPackageを用いたのは、線源からのX線は放射線状に放出されるので、Packageが厚くなるほどテストチャート透過後Package内で広がり、放射線が検出される領域も広がってしまうので、像がボケるからである。また、Packageを満たすガスはKr:80%、CO_2:20%の割合で混合したものを用いた。電離ガスとしてKrを用いたのは、Arよりも電子が走る際のエネルギー損失が大きいので、入射X線に電離された光電子の飛程が短く、X線が入射した位置により近い部分で信号を検出でき、イメージの精度が上がるからである。

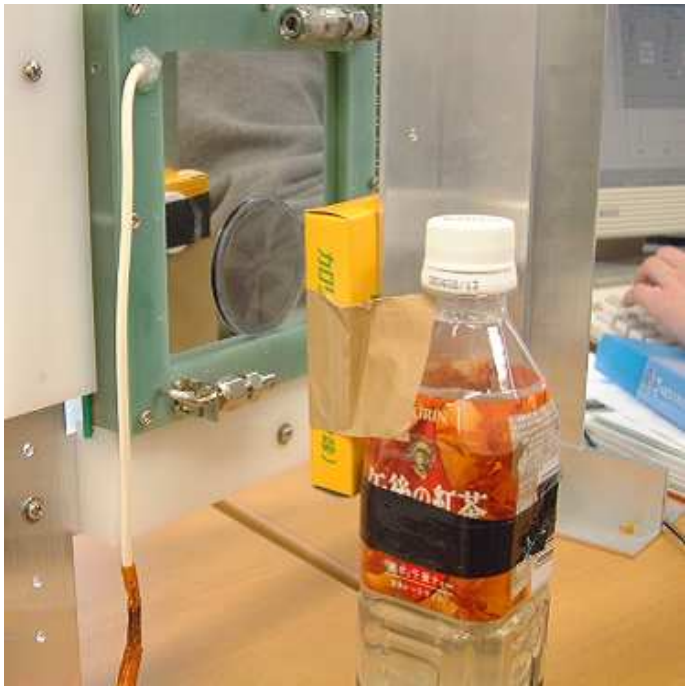

図 22: 2-D image 検出のための setup

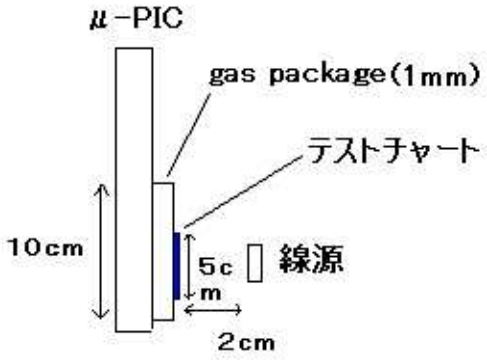

図 23: 2-D image 検出のための setup（上からの概念図）

5.3　結果

まず、図 24 の写真のような円形のテストチャートを置いて測定した。このテストチャートは、円の中心から円外に向かう放射線状の縞と、円の外周のやや内側に円形に開いた部分がある。右上にある縦線が 1cm を表し、四角で囲った部分は ASD Amp を挿した有感領域である。測定時間はおよそ 40 分、event 数は 33110count である。

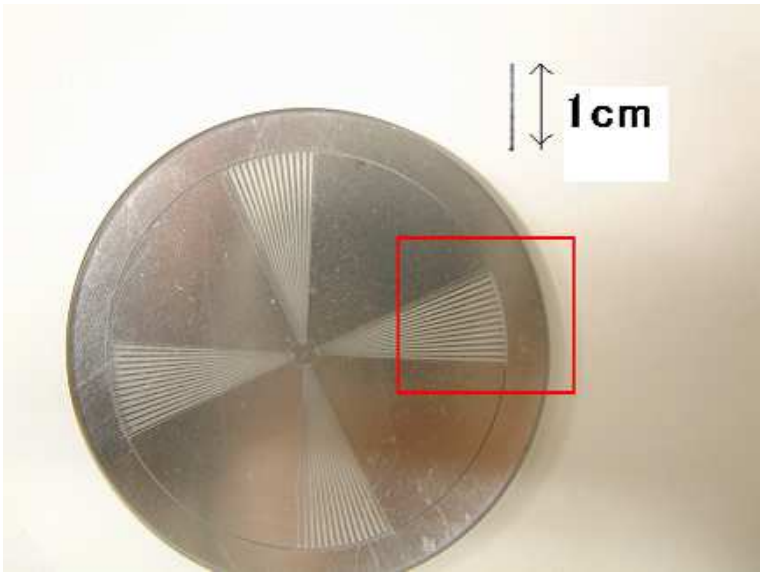

図 24: テストチャート写真（丸）

結果は次頁の図 25 のようになった。

この図は、黒にいくほどカウント数が少なく、白にいくほどカウント数が多いグレースケールで表示されている。結果では、放射線状の縞はぼんやりとしか読み取れないが、外側の円形の隙間の部分ははっきりと確認できる。縦横に入った黒線は μ−PIC 自体の問題で読み取れない領域である。

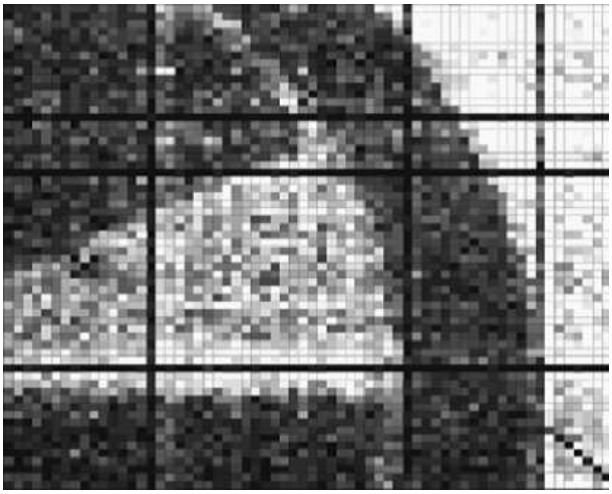

図 25: テストチャートイメージ（丸）

次に、図 26 のような四角のテストチャートを置いて測定した。これには、平行な横線の縞や、数字の形に鉛を配置した部分がある。測定時間はおよそ 120 分、event 数は 109070count である。

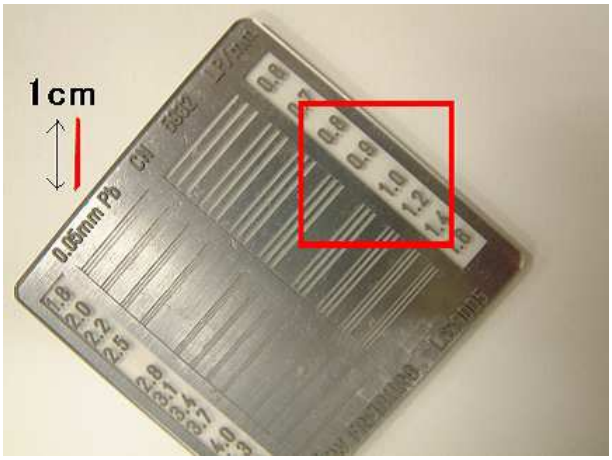

図 26: テストチャート写真（四角）

結果は図 27 のようになった。

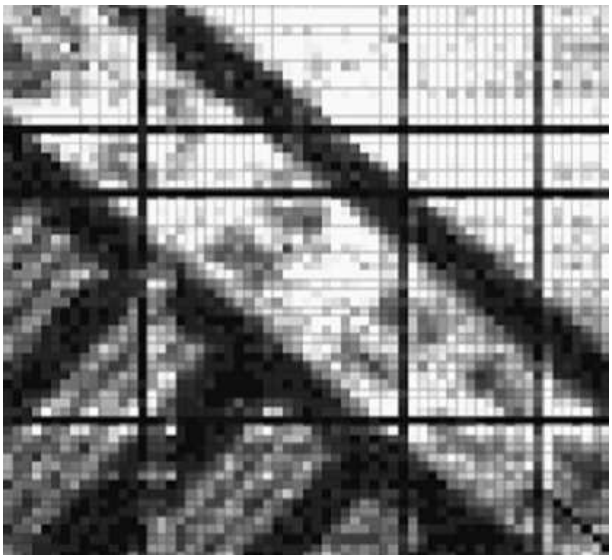

図 27: テストチャートイメージ（四角）

この結果を見ると、数字は、0 と 1 はかろうじて見分けられるが、はっきり読み取れるほどではない。ただ、平行な縞ははっきりと確認できる。

位置分解能としては、この縞の幅およそ 0.5mm 程度は確保できているものと考えられる。

6　3-D Tracking

前節では μ−PIC を用いて２次元の情報を取り出せることを確かめた。この節では、これにさらに深さ方向の情報を加えて、放射線が package 内でどういう飛跡を辿ったかを検出することを試みる。

6.1　実験原理

実験に際し、装置の配線は図 21 と同様である。但し、今回使う Encoder Board のプログラムは、２次元情報とともに、信号を検出した時間情報を同時に Memory Board に出力するものを用いた。時間情報とは、Encoder Board が内部に持っている clock を指す。Encoder Board は 20MHz で、つまり 50ns ごとに ASD からの入力信号を処理しているが、２次元情報を出力するときに、同時にそれを処理した時刻、clock を出力する。（プログラムは後ろに掲載）

放射線が package に入ると２次電子を次々と電離していくが、２次電子が検出面までドリフトし、信号として検出されるまでに要する時間は、電離した位置によって変化する。この時間差が、Encoder Board での clock の差に現れる。一方、ドリフト速度は電離する位置、ドリフトする過程でほぼ一定と考えられるので、この時間差はそのまま電離した位置の深さ情報に置き換えられる。これにより、Encoder Board からのデータを３次元の飛跡のデータとして取り出すことができるのである。

6.2　ドリフト速度の見積もり

時間差の情報を深さ情報に置き換えるには、ドリフト速度を知ることが必要である。そこで、Garfield というシミュレーションプログラムを用いた。このプログラムは、ガス検出器の機械的条件や印加する電圧、封入するガスの種類などを変化させて、その中での荷電粒子のドリフトや飛跡に関する情報を取り出すものである。

今、二枚の無限平面のうち一枚にある負の電圧、もう一枚に 0V がかかっていると仮定し、さらにガスとして実際に測定に用いる $Kr:CO_2$=9:1 の、常温で 1 気圧の混合ガスを考え、その中での電子の速度をシミュレーションで求めた。それをプロットしたものが図 28 である。

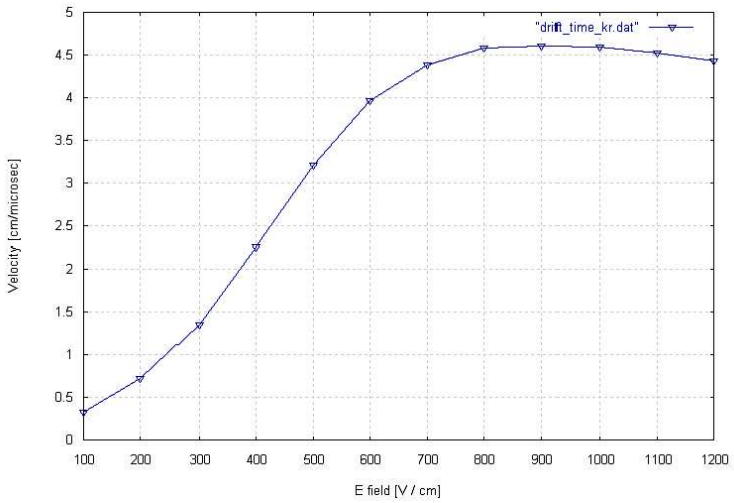

図 28: ドリフト速度の電場依存性

6.3 測定

3 次元の深さ方向の情報を取り出すことを目的としているので、package は深さ方向に厚いことが望ましい。そこで本測定では、図 29 のような 8cm 厚の package を用いた。この package は、電場を出来るだけ平行に保つために、抵抗分割によりドリフト面だけでなくいくつかの電極に連続した電圧を印加して電場を作り出している。

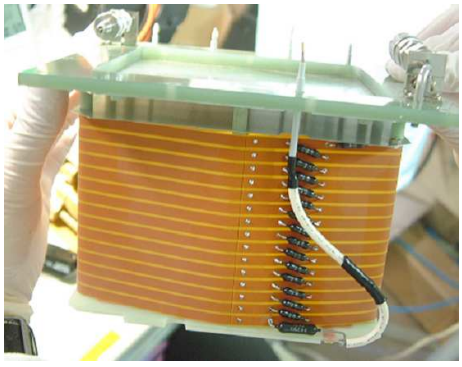

図 29: 8cm gas package

本測定ではこの package に、Kr と CO_2 を 9:1 の割合で混合したガスを流した。ドリフト面に印加した電圧は -4kV である。ドリフト速度の見積もりによれば、この電場でのドリフト速度はおよそ 3.2cm/μsec である。よって、8cm 厚の package では、ひとつの放射線につき最大で 2.5μsec のドリフトの時間差が生じるということであり、つまり Encoder Board からの情報では、ひとつの放射線の最初と最後で、最大 50clock の差が生じるということである。

一方、本測定での count rate はおよそ 10Hz 程度であり、放射線ごとの時間間隔は clock にして十分長い。これより、50clock の間に比較的 clock の連続するデータ列があれば、それはひとつの放射線が連続して電離を起こしたものによると考えてほぼ間違いないと言える。

さて、実験は図 30、31 のような setup を組んで行った。

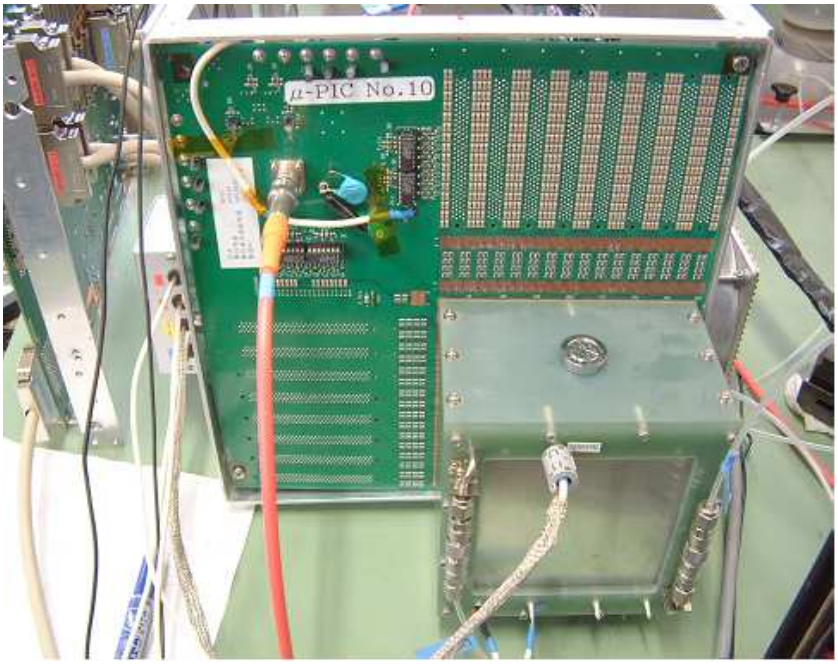

図 30: 3-D 飛跡測定のための setup

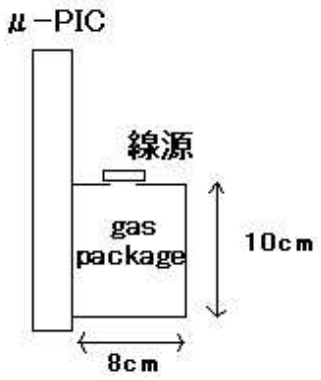

図 31: 3-D 飛跡測定のための setup（横からの概念図）

package の横にある入射窓に線源を置いて、放射線を照射している。用いた線源は β 線源の ^{90}Sr／^{90}Y であり、それぞれの Q 値は 546keV、2.28MeV である。β 崩壊では同時に反ニュートリノが生成されるので、ニュートリノが持ち去るエネルギーに応じて、線源から放射される β 線の Energy は変化する。その Energy 分布の最大値が Q 値である。検出面の HV には 620V 印加した。

6.4 結果

上の条件、つまり 50clock の間で比較的 clock の連続したデータ列をいくつか pickup した。ただし、このデータ列の情報はあくまで電離ごとの時間差であり、正確に電子がどの深さで電離したかを知ることはできない。そこで、一番ドリフト面に近い位置で起こった電離はドリフト面で起こったものと仮定して、そこから時間差の情報を深さ情報に直してプロットしたものが図 32 である。

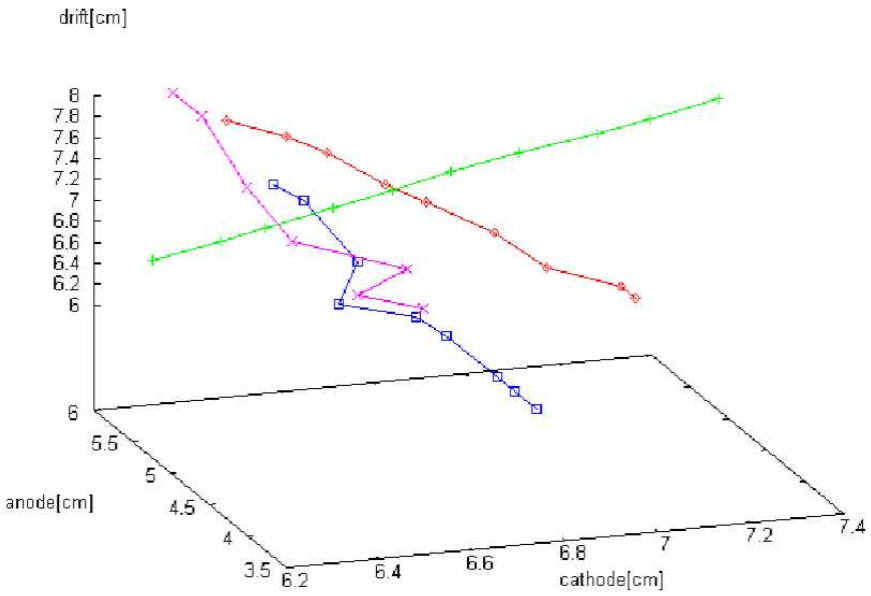

図 32: β 線源を置いたときの飛跡

これらは 10 点近くのデータ列から成っており、およそ直線に近い飛跡をたどっていることから、なんらかの荷電粒子の飛跡であると推測される。ただし、電子である β 線は比較的容易に進路を曲げられるので、一部途中で折り曲がりの見られる飛跡もあった。

一方、線源を置かずに飛跡を取るという実験も行った。すると、図 33 という飛跡群が得られた。

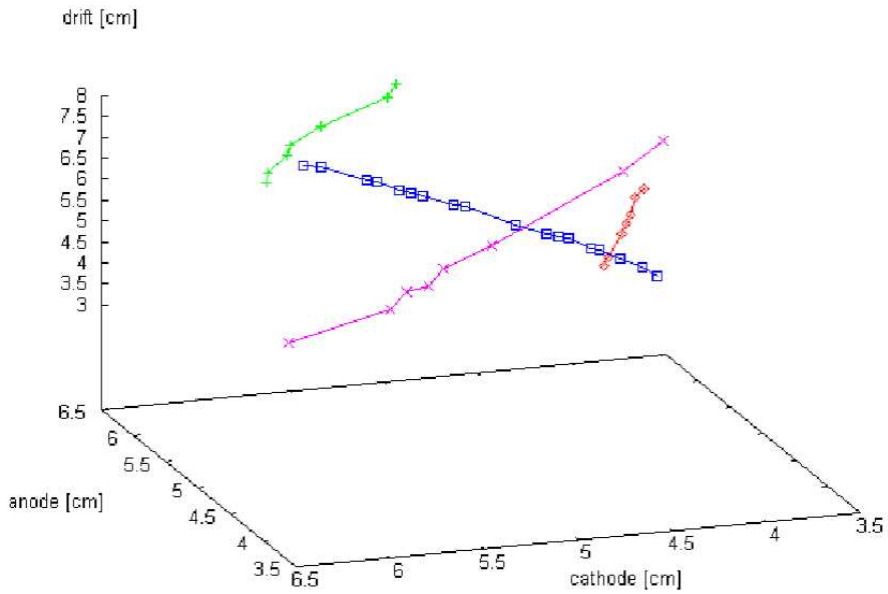

図 33: 線源を置かないときの飛跡

こういった飛跡は、Ar を電離ガスとして用いた場合には得られておらず、Kr ガスを電離ガスとして流した場合に特有のものであると考えられる。具体的には、用いているガス中に、主要に含まれる ^{84}Kr 以外に、同位体の ^{85}Kr が含まれている可能性が考えられる。この ^{85}Kr からは Q 値が 687keV という線源からの放射線に近いエネルギーの β 線が放射されているからである。この線源なしの飛跡を先ほどの β 線源を置いたときの飛跡と区別することは難しく、図 32 の飛跡の全てが線源からの放射線であるとは限らない。

7　本実験

本実験の目的は、入射した放射線のエネルギー損失の分布を求めることで、入射する放射線群の同定を行うことであった。今までの実験より、μ−PIC を用いて、入射した放射線がガス中で落とすエネルギーと、入射した放射線が描く飛跡を取得することが可能であることがわかった。もしこれらふたつの情報を、ひとつの放射線につき同時に測定することができたら、各放射線ごとのガス中でのエネルギー損失を求めることができることになる。すると、その測定を繰り返すことでエネルギー損失の分布が得られる。

本実験では、図 2 に示すようなトリウムが添加されているマントルを実験に用いて、マントルから放射されているはずのトリウム系列の放射線をエネルギー損失を用いて同定することを試みる。

7.1　原理

後述するように、マントルからのほとんどの α 線は、8cm gas package 中で全てのエネルギーを落とす。3-D Tracking と同様の測定により飛跡を取得したとき、同時に全ての ASD Amp からのアナログ信号を Flash ADC で読み出し、Spectroscopy と同様の方法によりそのエネルギー値を算出して、その合計を取るという操作を行う。これにより、放射線の飛跡とその放射線が有感領域で落とした全エネルギーを同時に測定でき、エネルギー損失、及びその分布を求めることができる。

一方、荷電粒子のエネルギー損失は、以下の Bethe-Bloch の式で与えられる。

$$-\frac{dE}{dx} = 2\pi N_a r_e^2 m_e c^2 \frac{Z}{A}\frac{z^2}{\beta^2}\left[ln\left(\frac{2m_e\gamma^2 v^2 W_{max}}{I^2}\right)\right]$$

r_e:古典電子半径　　m_e:電子の質量
N_a:アボガドロ定数　　I:平均励起ポテンシャル
Z:吸収物質の原子番号　A:吸収物質の原子量
ρ:吸収物質の密度　　z:入射粒子の電荷
$\beta = \frac{v}{c}$:入射粒子の速度　$\gamma = 1/\sqrt{1-\beta^2}$
W_{max}:1 回の衝突で入射粒子が与えうる最大の Energy

この式を利用して、さらに package 内のマントルや ASD Amp の位置などの幾何学的条件を加味することで、理論的に予想されるエネルギー

損失の分布をシミュレーションにより導出することができる。これを実験データと比較することで、マントル中のトリウム系列の同定が可能となる。

7.2 setup

まず、α線はμ−PICのドリフト面のAlなどによって遮蔽されてしまうので、マントルからのα線を見るには、マントルをドリフト面より内側のガス中に入れる必要がある。そこで、図34に示すようにマントルを8cm gas packageの内部に取り付けた。

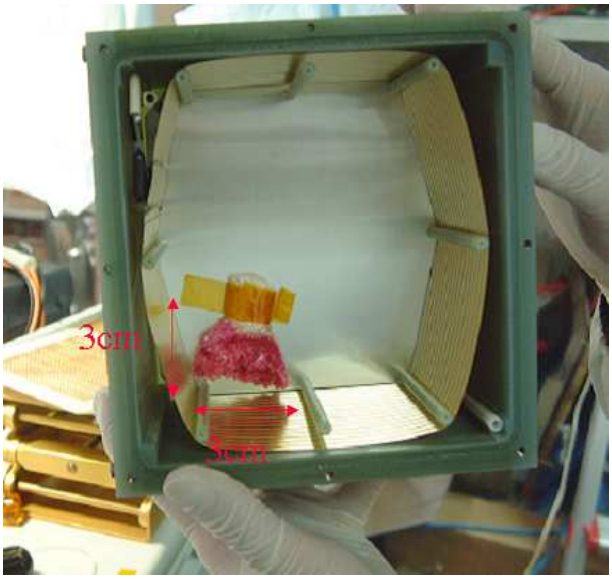

図 34: 本実験のためのsetup。package中にマントルを入れた。

7.3 飛跡の測定

まず、β線の時と同じように飛跡を取得することを試みた。装置の配線は図21と同様である。検出面のHVには620V印加した。用いたガスは、KrとCO_2を9:1の割合で混合したものである。

すると、図35に示すような結果が得られた。

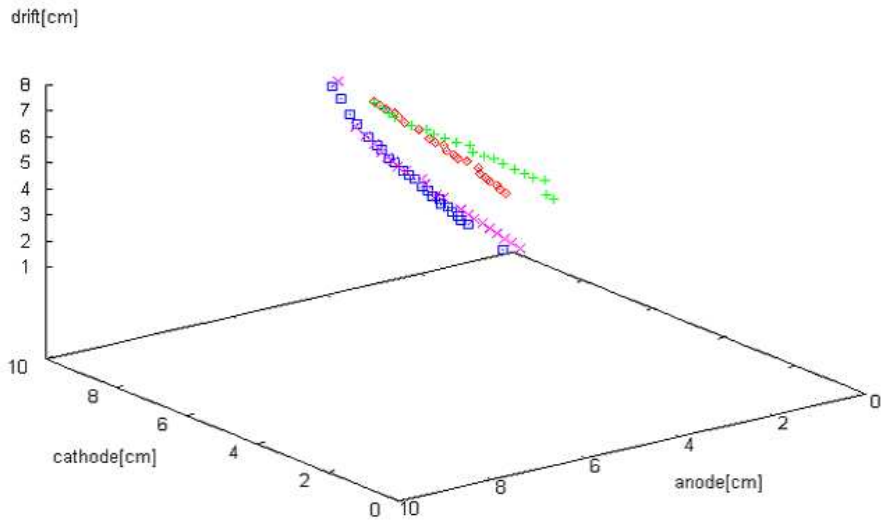

図 35: マントルからの放射線の飛跡

これを anode-cathode 面に投影したものが図 36 である。

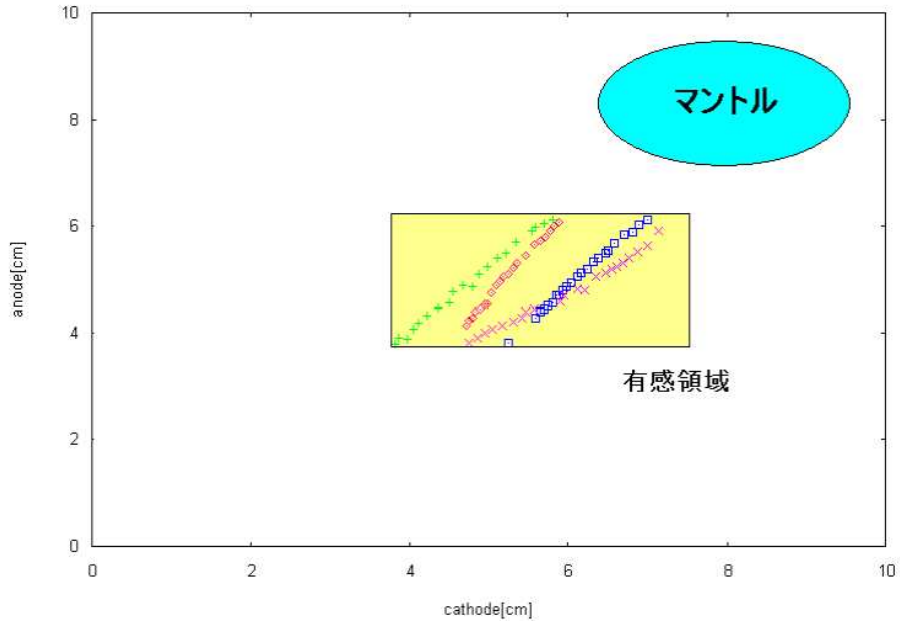

図 36: マントルからの放射線の飛跡（anode-cathode 面）

マントルの位置は、anode、cathode ともに値が大きいところに設置してあるので、確かにマントルからの放射線が飛跡として見えていることがわかる。また、α 線は β 線と比べてエネルギー損失が大きく、ガス中を飛行する時間も長い。これが飛跡の点の密度の濃さに現れている。また、α 線は β 線と比べて電子を電離する際に曲げられにくいが、この飛跡が β 線源を置いたときよりもより直線的であることからも、この飛跡が α 線であることが確認できる。

一方、トリウム系列により放射される α 線の Kr 中の飛程は、表 1 に示す程度である。つまり、8cm gas package 中でほぼ全ての放射線がその全てのエネルギーを落とす。そこで、ここでの測定では有感領域を図 36 の領域にしか取れなかったが、ASD Amp を増やして有感領域を拡大することにより、α 線がガス中で持てるエネルギーを全て失う点、つまり、飛跡の途切れる点を観測することができるはずである。一方、β 線の Kr 中での飛程は、表 2 に示す程度である。^{85}Kr からの β 線の Energy は 687keV であるので、その飛程は Gas Package 内の空間のスケールに比べてずっと長いはずである。これにより観測している放射線が ^{85}Kr からの β 線でなく、マントルからの α 線であることが確実に言えると考えられる。

Energy(MeV)	4.0	5.4	5.7	6.1	6.3	6.8	8.8
飛程 (cm)	2.3	3.2	3.4	3.7	4.0	4.4	6.3

表 1: Kr 中の α 線の飛程

Energy(MeV)	0.04	0.1	0.5	1	2
飛程 (cm)	1.4	6.4	74	180	380

表 2: Kr 中の β 線の飛程

7.4 波形の測定：Bragg 曲線

ガス中での放射線のエネルギー損失の過程は、図 37 に示すような Bragg 曲線で表される。

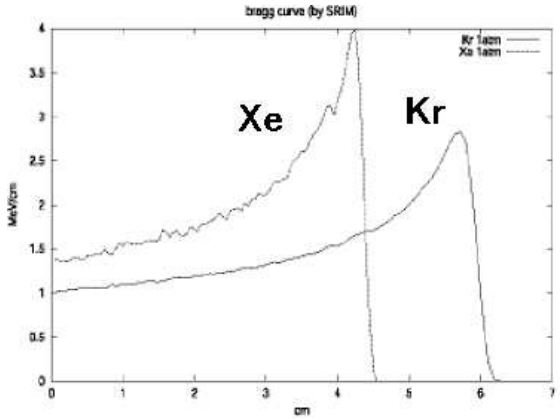

図 37: Bragg 曲線

一方、ASD Amp からのアナログ信号をオシロスコープで観測すると、図 17 のような形の波形以外に、図 38 のような波形が取得できた。

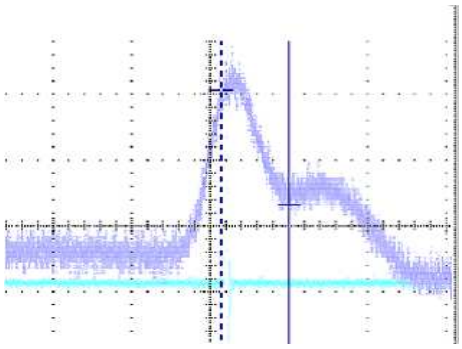

図 38: α 線による信号の波形

α 線はドリフト面から検出面の方向へ進むので、より早く検出された方が飛跡においては終端に近いほうである。つまりこの波形は、Bragg 曲線において飛跡の終端部でより多くのエネルギーを落としていることを実際に観測できているものと考えられる。

こうした波形を全ての Amp で Flash ADC により読み出すことで、有感領域内で落とした全 Energy を測定できる。ただし、本実験では時間の関係もあり、具体的に Energy の測定は行えなかった。

8　まとめと計画

8.1　まとめ

時間と装置の使用状況により目的のところまではいけなかったが、やったところまでのまとめを書いておくことにする。

これまでのセクションで挙げたように、実際に μ−PIC を用いることによって放射線のエネルギー、2 次元のイメージ、荷電粒子の 3 次元飛跡を測定できることがわかった。このことを利用することによって、荷電粒子（本実験では α 線）の検出器内での飛跡と、そのとき落とすエネルギーを求めることができる。

8.2　計画

今挙げたように、やってきた結果を利用することで、マントルから放射される α 線の検出器内での飛跡とそのとき落とすエネルギーを求めることができる。よって、このデータをたくさんとることによりマントルから出てくる α 線によるエネルギー損失の分布を求めることが可能となる。

また、これと並行してシミュレーションを行い、この結果と実験結果それぞれを照らし合わせることによりマントルからの放射線が確かにトリウム系列の α 線によるものであると言えるはずである。

9 プログラムソース集

基本的には、もともとあったものに手を加えたものがほとんどである。

9.1 spectrum の取得

9.1.1 RPV160 の起動プログラム

```
#include "rpv160.h"

int data_num_fadc(unsigned int addr){
  unsigned short  *ptr;
  int             flag, num;
  flag=0;
  while(1){
    ptr = (unsigned short *)(addr + 0x30004);
    flag = *ptr & 0x20;
    printf("");
    if(flag)
      break;
  }
  ptr = (unsigned short *)(addr + 0x30000);
  num = *ptr;
  num = num * 2;

  return num;
};

void read_fadc(unsigned int addr, int num, int ch, int *data){
  unsigned short  *ptr;

  num = num/2;
  for(int i=0; i<num; i++){
    ptr = (unsigned short *)(addr + i*2 + ch*0x2000);
    *data = *ptr >> 8;
    data++;
```

```
    *data = *ptr &0x00ff;
    data++;
  }
};

void start_fadc(unsigned int addr){
  unsigned short  *ptr;

  ptr = (unsigned short *)(addr + 0x30008);
  *ptr = 0x0001;
};

void stop_fadc(unsigned int addr){
  unsigned short  *ptr;

  ptr = (unsigned short *)(addr + 0x30008);
  *ptr = 0x0002;
};

void clear_fadc(unsigned int addr){
  unsigned short  *ptr;

  ptr = (unsigned short *)(addr + 0x30008);
  *ptr = 0x0004;
};

void set_fadc(unsigned int addr){
  unsigned short  *ptr;

  ptr = (unsigned short *)(addr + 0x20000);
  *ptr = 0x0003;
  ptr = (unsigned short *)(addr + 0x20002);
  *ptr = 0x0012;
  ptr = (unsigned short *)(addr + 0x20004);
  *ptr = 0x5678;
```

```
  ptr = (unsigned short *)(addr + 0x20006);
  *ptr = 0x0003;

  for(int i=0; i<8; i++){
    ptr = (unsigned short *)(addr + 0x21002 + i*0x200);
    *ptr = 0x0080;
  }
};

unsigned int init_fadc(int *dev, long base, size_t *mapsize){
  int         offset, pagesize;
  char        *addr;

  fprintf(stderr,"    fadc base address    : 0x%lx\n", base);

  if((*dev = open("/dev/vmedrv32d16", O_RDWR)) == -1){
    perror(" open : /dev/vmedrv32d16 ");
    exit(1);
  }

  pagesize = getpagesize();
  *mapsize = 0x40000;
  offset = base % pagesize;
  base = base - offset;
  addr = (char *)mmap(0, *mapsize, PROT_READ|PROT_WRITE,
                      MAP_SHARED, *dev, base);
  if(addr == MAP_FAILED){
    perror(" mmap : fadc failed. ");
    exit(1);
  }
  addr += offset;

  fprintf(stderr, "        pagesize         : 0x%x\n", pagesize);
  fprintf(stderr, "        mapsize          : 0x%x\n", *mapsize);
  fprintf(stderr, "  fadc address on linux  : 0x%x\n", (unsigned int) addr);
```

```
  return (unsigned int)addr;
};

int end_fadc(int dev, size_t mapsize, unsigned int addr){
  munmap((char *) addr, mapsize);
  close(dev);
  return 0;

};
```

9.1.2 RPV130 の起動プログラム

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <errno.h>
#include <unistd.h>
#include <fcntl.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include <sys/mman.h>
#include <sys/param.h>
#include "vmedrv.h"
#include "rpv130.h"

static struct vmedrv_interrupt_property_t  pr;

int latch_1_rpv130(unsigned int addr){
  unsigned short  *ptr;
  int             data;
```

```
  ptr = (unsigned short *)(addr + 0x0);
  data = (int) *ptr;
  return data;
};

int latch_2_rpv130(unsigned int addr){
  unsigned short  *ptr;
  int             data;

  ptr = (unsigned short *)(addr + 0x2);
  data = (int) *ptr;
  return data;
};

int latch_ff_rpv130(unsigned int addr){
  unsigned short  *ptr;
  int             data;

  ptr = (unsigned short *)(addr + 0x4);
  data = (int) *ptr;
  return data;
};

int read_rpv130(unsigned int addr){
  unsigned short  *ptr;
  int             data;

  ptr = (unsigned short *)(addr + 0x6);
  data = (int) *ptr;
  return data;
};

void pulse_rpv130(unsigned int addr, int data){
  unsigned short  *ptr;
```

```
  ptr = (unsigned short *)(addr + 0x8);
  *ptr = (unsigned short) data;
};

void level_rpv130(unsigned int addr, int data){
  unsigned short  *ptr;

  ptr = (unsigned short *)(addr + 0xa);
  *ptr = (unsigned short) data;
};

void intr_rpv130(int dev, void (*sigh)(int)){
  pr.irq = LEVEL;
  pr.vector = VECTOR;
  pr.signal_id = SIGNAL_ID;

  signal(SIGNAL_ID, sigh);
  ioctl(dev, VMEDRV_IOC_REGISTER_INTERRUPT, &pr);
};

void ena_intr_rpv130(int dev, unsigned int addr){
  unsigned short  *ptr;

  ptr = (unsigned short *)(addr + 0xc);
  *ptr = 0x5b;
  ptr = (unsigned short *)(addr + 0xe);
  *ptr = 0x5b;
  ioctl(dev, VMEDRV_IOC_ENABLE_INTERRUPT);
};

void dis_intr_rpv130(int dev){
  ioctl(dev, VMEDRV_IOC_DISABLE_INTERRUPT);
};
```

```
void reset_intr_rpv130(int dev){
  ioctl(dev, VMEDRV_IOC_UNREGISTER_INTERRUPT, &pr);
  signal(SIGNAL_ID, SIG_DFL);
};

unsigned int init_rpv130(int *dev, long base, size_t *mapsize){
  int       offset, pagesize;
  char      *addr;

  if((*dev = open("/dev/vmedrv16d16", O_RDWR)) == -1){
    perror(" open : /dev/vmedrv16d16");
    exit(1);
  }

  pagesize = getpagesize();
  *mapsize = 0x1000;
  offset = base % pagesize;
  base = base - offset;
  addr = (char *)mmap(0, *mapsize, PROT_READ|PROT_WRITE,
      MAP_SHARED, *dev, base);
  if(addr == MAP_FAILED){
    perror(" mmap : RPV130 failed.");
    exit(1);
  }
  addr += offset;

  return (unsigned int)addr;
};

int end_rpv130(int dev, size_t mapsize, unsigned int addr){
    munmap((char *) addr, mapsize);
  close(dev);
  return 0;
};
```

```
// Written by Nagayoshi
void status_rpv130(unsigned int addr){
  unsigned short *ptr;

  ptr = (unsigned short *)(addr + 0xc);
  fprintf(stderr,"CSR1: 0x%x\n", *ptr);

  ptr = (unsigned short *)(addr + 0xe);
  fprintf(stderr,"CSR2: 0x%x\n", *ptr);
}

void clear_csr1_rpv130(unsigned int addr){
  unsigned short *ptr;

  ptr = (unsigned short *)(addr + 0xc);
  *ptr = 0x5b;
  fprintf(stderr,"CSR1: 0x%x\n", *ptr);
}

void clear_csr2_rpv130(unsigned int addr){
  unsigned short *ptr;

  ptr = (unsigned short *)(addr + 0xe);
  *ptr = 0x5b;
  fprintf(stderr,"CSR2: 0x%x\n", *ptr);
}
```

9.1.3 積分値を取得するプログラム

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
#include <errno.h>
#include <unistd.h>
#include <fcntl.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include <sys/mman.h>
#include <sys/param.h>
#include "vmedrv.h"
#include "rpv160.h"
#include "rpv130.h"
#define BASE160 0x00400000
#define BASE130 0x2000

int main(){
  int        dev160, dev130;
  size_t     map160, map130;
  char       *addr160, *addr130;

  // Set-up RPV-160
  addr160 = (char *)init_fadc(&dev160, BASE160, &map160);
  set_fadc((unsigned int)addr160);
  clear_fadc((unsigned int)addr160);
  start_fadc((unsigned int)addr160);

  // Set-up RPV-130
  addr130 = (char *)init_rpv130(&dev130, BASE130, &map130);
  pulse_rpv130((unsigned int)addr130, 0x7);

  int num = data_num_fadc((unsigned int) addr160);
  int ch[8][num];
  int max;
  int data = -1;
  int a[140];
```

```
int s = -1;
int integ;
for(int i=0; i<8; i++){
  read_fadc((unsigned int) addr160, num, i, &ch[i][0]);
}
for(int i=num-140; i<num; i++){
   s = s+1;
  a[s] = ch[7][i];
  if(data == -1){
    data = a[s];
    integ = data;
  }
  else if(data != -1){
    data = a[s];
    integ = integ + data;
  }
}

int mean;
mean = integ/140;

int b[140];
int t = -1;
int peak = 0;
int aho = 0;

for(int i=num-140; i<num; i++){
  t =t+1;
  b[t] = ch[7][i];
  if(mean <= b[t]-0){
    peak = peak + b[t];
    aho = aho + 1;
  }
}
```

```
  int rest;
  int spec;
  rest = aho*(mean+0);
  spec = peak - rest;

  printf("%d",spec);
  printf("\n");

  char fname[100];
  FILE *fp;
  sprintf(fname, "./filename_%dhakei.dat", spec/10);
  fp = fopen(fname, "w");
  for(int i=0; i < 140; i++)
  {
  fprintf(fp, "%d %d\n", i, a[i]);
  }
  fclose(fp);

  end_fadc(dev160, map160, (unsigned int) addr160);

  dis_intr_rpv130(dev130);
  end_rpv130(dev130, map130, (unsigned int)addr130);

  return 0;
}
```

9.1.4 積分値をたくさんとるためのシェルスクリプト

```
#!/bin/sh

i=-1

while let "++i <10000";
```

```
do
    echo "$i"
    ./main_brandnew >> filename.dat
done

while let "++j <100";
do

    ../memory/a.out

done
```

9.1.5 1次元ヒストグラム表示のマクロ for ROOT

二つのヒストグラムを同時に表示させるものをのせておく。

```
{
  gROOT -> Reset();
  gStyle->SetOptStat(0000000);

  TCanvas c1("data","spectrum");

  TH1S *h1 = new TH1S("data","spectrum",100,0,800);
  ifstream data("/home/kazu/macro/filename.dat");

  double x;
  while(data >> x){
    h1 -> Fill(x);
  }
  data -> close();

  TH1S *h2 = new TH1S("data","spectrum2",100,0,800);
  ifstream data("/home/kazu/macro/filename.dat");
```

```
  double x;
  while(data >> x){
    h2 -> Fill(x);
  }
  data -> close();

  leg = new TLegend(0.7,0.7,0.9,0.9);
  leg -> AddEntry(h1,"   name","lp");
  leg -> AddEntry(h2,"   name","lp");

  h1 -> SetMarkerColor(2);
  h2 -> SetMarkerColor(3);
  h1 -> SetMarkerStyle(5);
  h2 -> SetMarkerStyle(4);
  h1 -> Draw("P");
  h2 -> Draw("sameP");
  leg -> Draw();
}
```

9.2 2D-Imaging,3D-tracking

9.2.1 memory board の起動

```
#include "memory.h"

int read_memory(unsigned int addr, int *data, int num){
  int         *ptr;
  int         flag;

  while(1){
    ptr = (int *)(addr + 0x4);
    flag = *ptr;
     if(flag == 0x0){
       break;
```

```
    }
    ptr = (int *)(addr + 0x0);
    *ptr = 0x2;
  }

  if(flag == 0x0){
    ptr = (int *)(addr + 0x0);
    for(int i=0; i<num; i++){
      *data = *ptr;
      data++;
    }
  }

  return flag;
};

int start_memory(unsigned int addr){
  int         *ptr;
  int         flag;

  ptr = (int *)(addr + 0x0);
  *ptr = 0x01;
  do{
    usleep(1);
    ptr = (int *)(addr + 0x4);
    flag = *ptr;
     fprintf(stderr, "%d", flag);
  }while(flag != 0x0);
  return flag;
};

void set_memory(unsigned int addr, int num){
  int         *ptr;

  ptr = (int *)(addr + 0x8);
```

```
  *ptr = num;
};

void abort_memory(unsigned int addr){
  int         *ptr;

  ptr = (int *)(addr + 0x0);
  *ptr = 0x0;
};

void reset_memory(unsigned int addr){
  int         *ptr;

  ptr = (int *)(addr + 0x0);
  *ptr = 0x2;
};

unsigned int init_memory(int *dev, long base, size_t *mapsize){
  int         offset, pagesize;
  char        *addr;

  if((*dev = open("/dev/vmedrv32d32", O_RDWR)) == -1){
    perror(" open : /dev/vmedrv32d32 ");
    exit(1);
  }

  pagesize = getpagesize();
  *mapsize = 0x1000;
  offset = base % pagesize;
  base = base - offset;
  addr = (char *)mmap(0, *mapsize, PROT_READ|PROT_WRITE,
                      MAP_SHARED, *dev, base);
  if(addr == MAP_FAILED){
    perror(" mmap : memory board failed. ");
    exit(1);
```

```
  }
  addr += offset;

  return (unsigned int)addr;
};

int end_memory(int dev, size_t mapsize, unsigned int addr){
  munmap((char *) addr, mapsize);
  close(dev);
  return 0;
};
```

9.2.2 2D,3D のデータ取得

2D の場合は以下のプログラムの//printf の部分の x,y を printf で出力すればよい。

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <errno.h>
#include <unistd.h>
#include <fcntl.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include <sys/mman.h>
#include <sys/param.h>
#include <vmedrv.h>
#include "memory.h"

#define BASE 0x300000
```

```
#define ENUM 100

main(){
  int       dev,data[ENUM];
  size_t    map;
  char      *addr;
  int       x[ENUM], y[ENUM], clk[ENUM];

  addr = (char *)init_memory(&dev, BASE, &map);
  reset_memory((unsigned int) addr);
  set_memory((unsigned int) addr, ENUM);
  start_memory((unsigned int) addr);
  fprintf(stderr, "\n\n");
  read_memory((unsigned int) addr, &data[0], ENUM);

  for(int i=0; i<ENUM; i++) {

    x[i] = data[i] & 0x000003ff;
    y[i] = (data[i] >> 10) & 0x000003ff;
    clk[i] = (data[i] >> 20) & 0x00000fff;

    //printf("%d %d %d\n", x[i], y[i] ,clk[i]);
  }

  for(int j=0; j<ENUM-2; j++){

    if((clk[j]-clk[j+1]) < 10 &&
       (clk[j]-clk[j+1]) > 0){
  printf("%d %d %d\n", x[j],y[j],clk[j]);
    }
  }


  end_memory(dev, map, (unsigned int)addr);
}
```

9.2.3 2D,3D の情報をたくさんとるためのシェルスクリプト

```
#!/bin/sh

while let "++i < 10000";

do

echo $i
./main_clk_alpha2 >> filename.dat

done
```

9.2.4 2 次元ヒストグラム表示のマクロ for ROOT

```
{
  double x,y;

  TCanvas *c1 = new TCanvas("mu-pic","histgram"); //you can decide positions.
  TH2F *h1 = new TH2F("mu-pic","histgram",250,1,1000,250,1,1000);

  ifstream data("/home/kazu/macro/filename.dat");
  while(data >> x >> y){
    h1 -> Fill(x,1024-y);
  }

  data -> close();

  h1 -> Draw("COLZ");
}
```

9.2.5 Garfield に用いたスクリプト

```
&CELL
plane y 0.0 v = 0.0
plane y -8.0 v = -4000
cell-id "mpic"

&GAS
Global gas_file ‘krypton_90_co2_10.dat‘
Call inquire_file(gas_file,exist)
If exist Then
  GET {gas_file}
Else
  MAGBOLTZ KRYPTON 90 CO2 10
  Write {gas_file}
Endif

&drift
Call electron_velocity(0,-3,0,vx,vy,vz)
Say "Velocity: ({vx,vy})"
time

&stop
```

10　参考文献

- G.F.Knoll: ”放射線計測ハンドブック　第 3 版” 日刊工業新聞社 (2001).
- ニコラスツルファニディス: ”放射線計測の理論と演習＜上巻＞” 現代工学社 (1986).
- F.Sauli: ”Properties of multiwire proportional and drift chambers” CERN Report(1977).
- T.Nagayoshi: Ph.D. Thesis,Kyoto Univ(2004).
- 植野優:修士論文　京都大学 (2001).
- 高田淳史:修士論文　京都大学 (2004).
- 2003 年度Ｐ 6 課題研究レポート　京都大学 (2004)
- 日本アイソトープ協会編:”アイソトープ手帳 10 版”丸善株式会社 (2002)

11　謝辞

今回の実験を進めるにあたって、宇宙線研究室助手の身内さんには数多くの助言を頂き、かなりの助けとなりました。実験装置の扱い方、実験の進め方、トラブルにあったときの対処法、レポートのまとめ方などたくさんのことをご指導して頂き、本当にありがとうございました。とても勉強になりました。さらに、TA の岡田さんには、実験をする上でかなりのサポートをして頂きました。ありがとうございました。また、Ｐ６全体を通して、宇宙線研究室のスタッフ・院生の方々にも非常にお世話になりました。感謝致します。最後になりましたが、Ｐ６で共に実験やゼミを進めてきたシンチ班、塩班の人々とは、時には励ましあったり相談しあったりすることでお互いに刺激しあい、実験を楽しく進めることができました。ありがとう。

この経験を今後にも生かしつつ、これからも努力していこうと思う次第であります。皆様、本当にありがとうございました。