μ-PIC を用いた Cool-X の性能評価

澤田真理　　川手朋子

2006 年 3 月 19 日

概 要

宇宙での高エネルギー現象を捕らえるX線検出器を開発する際、既知のX線源を用いて検出器の較正を行う。地上で得られるX線源としては天然の放射性同位体のほか、X線管などのX線発生装置がある。X線発生装置は、検出器の較正だけでなく、物体や人体の非破壊検査にも役立つ一方、従来は大型で高価なものであったが、近年は持ち運びが可能な小型のX線発生装置が開発され、注目を集めている。このようなポータブルX線発生装置から放射されたX線 beam の形状や強度、そしてエネルギースペクトルを同時に測定し、その性能評価を行うのが本実験の主目的である。

本実験では、AMPTEK 社製の Cool-X(図 0.1) というポータブルX線発生装置を取り上げる。また、先の目的に合致する検出器として、京都大学宇宙線研究室で開発された μ-PIC(図 0.2) を使用した。実験の流れとしては、まず μ-PIC で可能ないくつかの測定・解析を行ってその取り扱いに慣れ、その後、Cool-X の性能評価を行った。



図 0.1: Cool-X

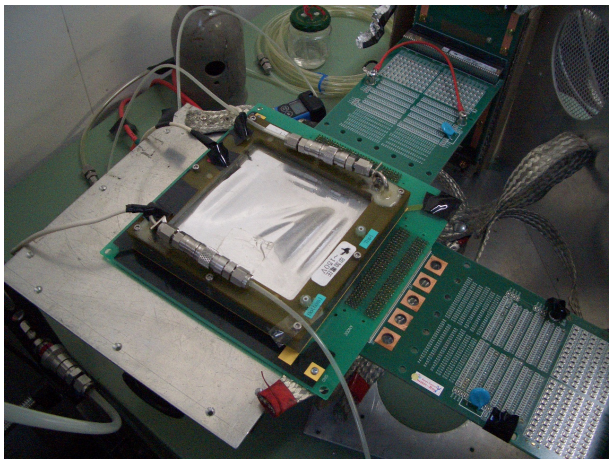


図 0.2: μ-PIC

目 次

1 検出器 **6**
- 1.1 μ-PIC について . . . 6
- 1.2 GEM について . . . 7
- 1.3 ASD について . . . 7
- 1.4 ゲイン測定 . . . 8
  - 1.4.1 μ-PIC のゲイン . . . 8
  - 1.4.2 GEM のゲイン . . . 10
  - 1.4.3 ゲインマップ . . . 12

2 Spectroscopy **13**
- 2.1 実験原理 . . . 13
- 2.2 実験方法 . . . 13
- 2.3 結果・考察 . . . 15

3 2-D imaging **21**
- 3.1 実験原理 . . . 21
- 3.2 実験方法 . . . 21
- 3.3 結果・考察 . . . 23

4 cool-X の性能評価 **26**
- 4.1 cool-X の原理 . . . 26
- 4.2 rate の時間変化 . . . 30
- 4.3 spectrum . . . 33
- 4.4 beam の拡がり . . . 37

5 まとめ **42**

A 実験に関する補足 **43**
- A.1 バックグラウンド測定 . . . 43
- A.2 実験中の μ-PIC の変調 . . . 44

B プログラム集 **45**
- B.1 spectrum の取得 . . . 45
  - B.1.1 RPV160 起動プログラム . . . 45
  - B.1.2 RPV130 起動プログラム . . . 47
  - B.1.3 波形取得プログラム . . . 51
  - B.1.4 積分用プログラム . . . 61
  - B.1.5 Cool-X の spectrum 取得用シェルスクリプト . . . 65
- B.2 2-D imaging . . . 69
  - B.2.1 memory board からのデータ取得 . . . 69
- B.3 rate の時間変化の取得 . . . 72
  - B.3.1 encorder board のクロックおよび 2-D 位置情報を取得するプログラム . . . 72

B.3.2 high rate と low rate を振り分けるプログラム 72
B.3.3 encorder board のクロックから rate を計算するプログラム 73
B.3.4 データ取得を繰り返すシェルスクリプト 74
B.4 解析用プログラム 76
B.4.1 2-D image のデータを x,y および count 数で cut するプログラム 76
B.4.2 2-D image を projection するプログラム 79
B.4.3 rate の時間変化を folding するプログラム 81
B.4.4 2-D image の beam の拡がりを計算するプログラム 83

謝辞 **92**

1 検出器

1.1 μ-PIC について

μ-PIC(micro-Pixel Chamber) はピクセル型ガス検出器で、図 1.1 のような 10cm × 10cm の大きさの領域に図 1.2 のような小型の比例計数管をプリント基板技術を用いて 400μm 間隔で 256 × 256 個並べたような構造をしたものである (図 1.3)。検出器に放射線が入射すると、検出器内のガスを電離する。これによって多数の陽イオン・電子対が生成され、発生した電子は Drift Plane と検出面の間にかけられた電場により検出面へと移動し、Anode-Cathode 間にかけられた電場によって電子なだれを起こし増幅された後、Anode から負のパルスとして出力される。これと同時に Cathode から陽イオンに電子が供給されるので、Cathode からは正のパルスが出力される。また、同時刻に出力のあった Anode-Cathode のピクセルの位置情報を座標として読み出す仕組みになっている。

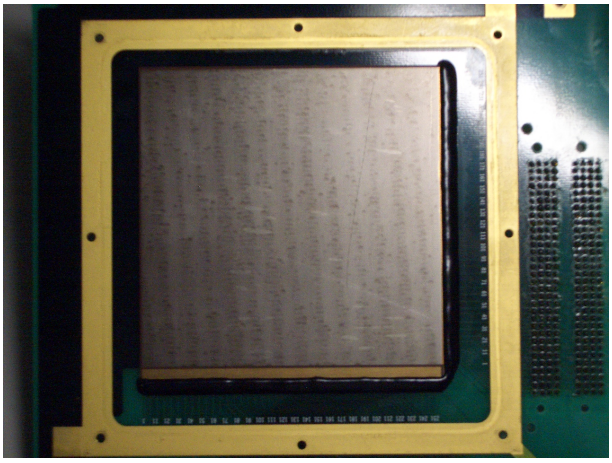


図 1.1: μ-PIC 検出面

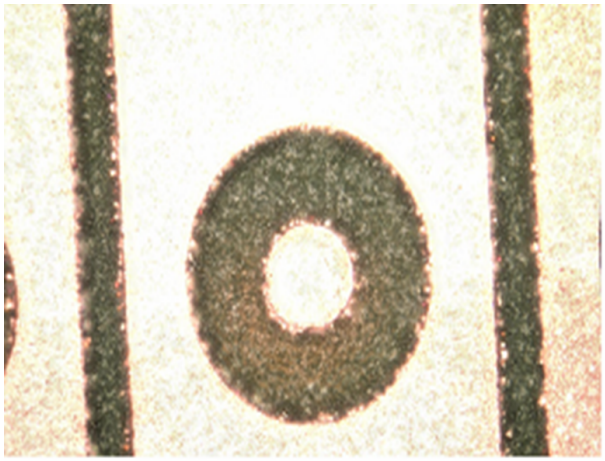


図 1.2: 検出面拡大図

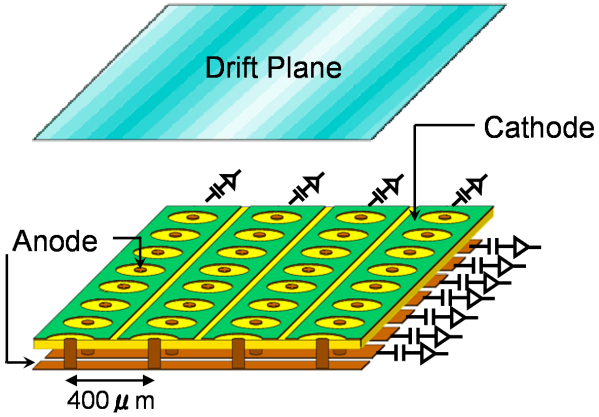


図 1.3: μ-PIC の模式図

1.2 GEM について

GEM(Gas Electron Multiplier) は厚さ 50μm の KAPTON という絶縁体の両面に厚さ 5μ の銅をコーティングしたもので、図 1.4 のように六角形のパターンで直径 70μm の穴が 140μm 間隔で空いた構造になっている。コーティングされた銅に表面 (GEM TOP) と裏面 (GEM BOTTOM) で異なる電位を与えることで、電子が穴を通過すると電子なだれを起こす。図 1.5 のように GEM を μ-PIC の Drift Plane と検出面の間に挟み、補助増幅器とすることで、μ-PIC の S/N を向上させることができる。

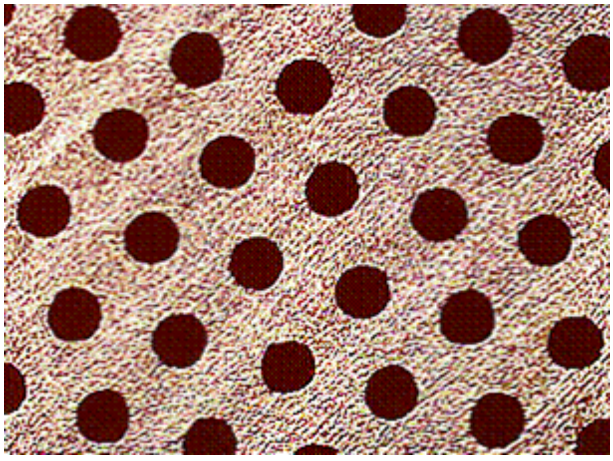


図 1.4: GEM の拡大図

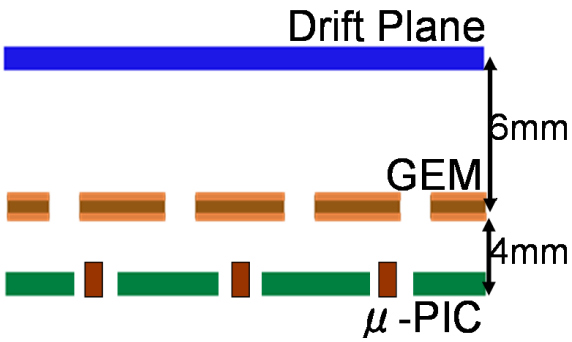


図 1.5: GEM の配置

1.3 ASD について

μ-PIC からの出力は、ASD(Amplifier Shaper Discriminator) と呼ばれる専用のプリアンプを用いて読み出される (図 1.6、1.7)。ASD で増幅された μ-PIC からの 256ch 分の信号は、16ch もしくは 32ch ずつ足し合わせたアナログ信号と、ASD で設定された閾値に応じた 256ch のデジタル信号として出力される。今回の実験では、時定数が 80[nsec] で、μ-PIC からの入力信号 1[pC] あたり 700[pC] の信号を出力するものを用いた。

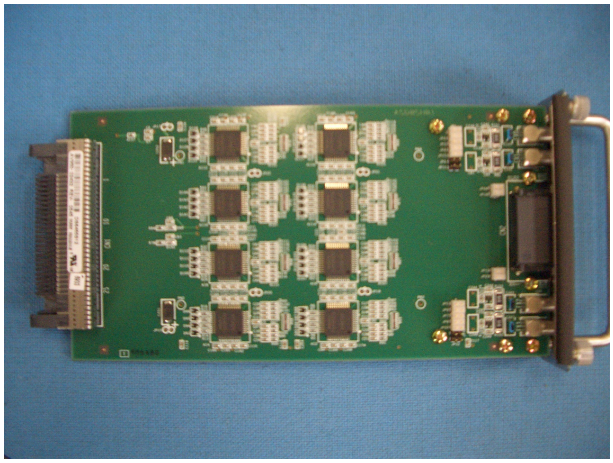


図 1.6: ASD 基盤

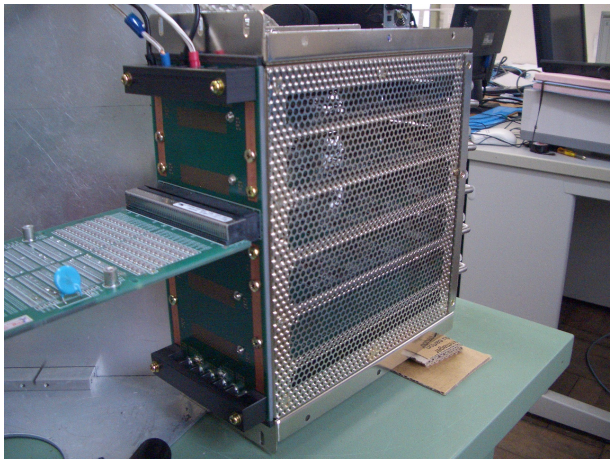


図 1.7: ASD の外見

1.4 ゲイン測定

入射した放射線が、ガスとの相互作用により検出器内でそのエネルギー E を全て落とすならば、放射線によって電離された 1 次電子の総電荷 Q_i は、

$$Q_i = \frac{E}{W} \times e[\mathrm{pC}] \tag{1.1}$$

となる。

ここで W は電子 1 個あたりを電離するのに必要な平均エネルギー (W 値) であり、ガスの種類に固有な量である。この電荷 Q_i は、GEM、μ-PIC 本体および ASD で増幅されて出力される。オシロスコープで見られる最終的な出力 Q_f は、ベースラインからのパルスの高さを V[mV]、パルスの高さが $V/2$ になるところでの時間幅 Δt[nsec] を用いて

$$Q_f = \frac{V \cdot \Delta t}{R}[\mathrm{pC}] \tag{1.2}$$

と近似できる。ここで $V \cdot \Delta t$ はパルスの形が三角形であるとして近似したパルスの面積に相当し、これをオシロスコープの終端抵抗 $R[\Omega]$ で割るとパルスの電荷量となる。

よって、GEM、μ-PIC、ASD のゲインをそれぞれ A_{GEM}、$A_{\mu-\mathrm{PIC}}$、A_{ASD} とすれば、

$$Q_i \times A_{GEM} \times A_{\mu-\mathrm{PIC}} \times A_{ASD} = Q_f \tag{1.3}$$

となり、これよりゲインを算出できる。

1.4.1 μ-PIC のゲイン

μ-PIC の増幅率を測定するため、水平に置かれた μ-PIC に流量 150[ml/min] で Ar:C_2H_6=9:1 の混合ガスを流し、安定化させるため 1 時間ほど待った。線源として ^{55}Fe を用いた。^{55}Fe の主な輝線は Kα の 5.9[keV] である。今回用いたガスでは W 値は 26eV であるから 5.9[keV] の X 線が入射するとおよそ 230 個の一次電子が生成される。よってこの一次電子の総電荷は式 1.1 より $Q_i = 3.63 \times 10^{-5}$[pC] となる。

Drift Plane に-1600[V] をかけ、Cathode を GND とした。以降の実験では Drift Plane と Cathode にかける電圧は常にこの値である。μ-PIC のほぼ中央、Drift Plane から約 5cm の位置に垂直かつ鉛直に ^{55}Fe を設置した。GEM には電圧をかけず、Anode に+500V 前後の電圧をかけた。Anode 電圧を変化させて、それに対する V、Δt を測定した。

式 1.2 から得た各 Anode 電圧に対する Q_f を図 1.8 に示す。

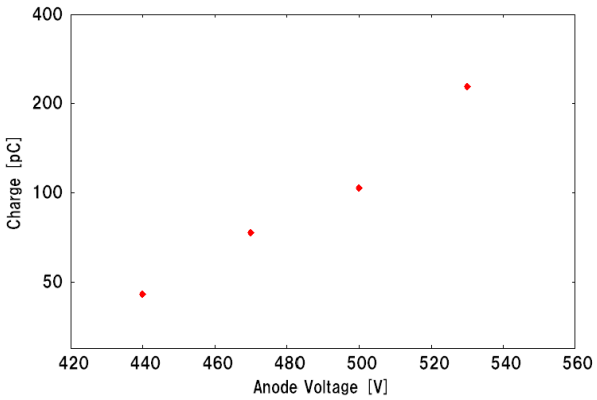


図 1.8: Anode 電圧に対する Q_f

$A_{ASD} = 700$[pC/pC] であることを用い、式 1.3 から $A_{\mu-\mathrm{PIC}}$ を算出し、図 1.9 に示す。この場合 GEM に電圧をかけていないため、式 1.3 における A_{GEM} は便宜的に 1 となる。

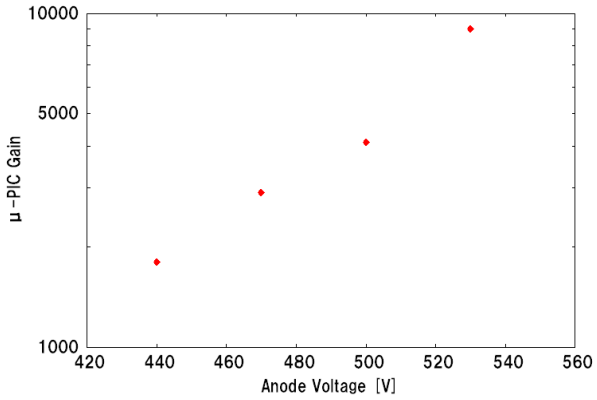


図 1.9: Anode 電圧に対する $A_{\mu-\mathrm{PIC}}$

これより、μ-PIC の電圧に対して $A_{\mu-\mathrm{PIC}}$ は指数関数的に増加している。なお、Anode 電圧が 540V 以上のとき μ-PIC は放電を起こした。このため、十分なゲインが得られ、且つ安定な動作が期待できる Anode 電圧は+500V 程度であると考えられる。

1.4.2 GEM のゲイン

μ-PIC と線源をさきほどと同様にセットアップし、Anode 電圧を+500V とした。GEM TOP に-1250V 前後、GEM BOTTOM は-1000V で一定の電圧をかけ、GEM TOP の電圧を変化させて V、Δt を測定した。ここで GEM の TOP-BOTTOM 間の電位差を ΔGEM とし、ΔGEM に対して、式 1.2 より Q_f を求め図 1.10 に示す。

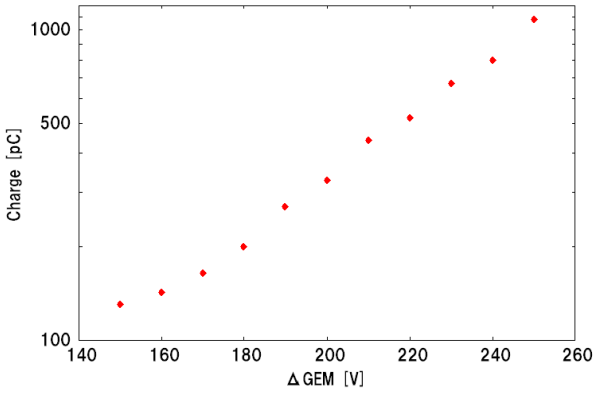


図 1.10: ΔGEM に対する Q_f

$A_{ASD} = 700$[pC/pC] および Anode 電圧が 500V のときの μ-PIC 単体のゲインが 4.1×10^3 であることより、式 1.3 を用いて A_{GEM} を算出し、図 1.11 に示す。

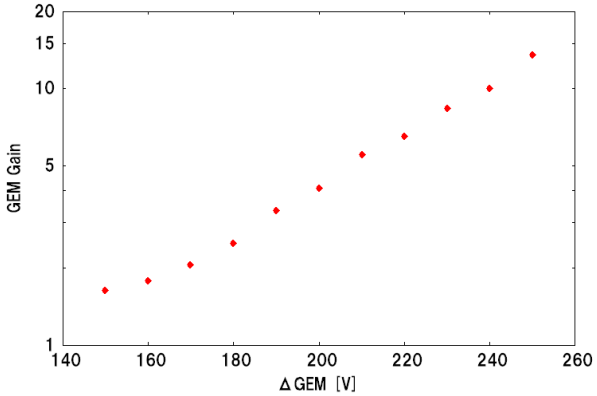


図 1.11: ΔGEM に対する A_{GEM}

図 1.11 より、GEM 間電圧に対して GEM のゲインは指数関数的に増加している。

次に、GEM BOTTOM と μ-PIC の間にかかる電場 (Induction Field) に対する A_{GEM} の変化を調べた。Anode 電圧を 500V とし、GEM BOTTOM にかける電圧を-1000V から-500V まで 50V 刻みで変化させ、同時に GEM TOP にかける電圧も同様に変化させて ΔGEM は 250V で一定とした。このとき、Induction Field は 2.5kV/cm から 1.25kV/cm まで 0.125kV/cm 刻みで変化する。それぞれの Induction Field の大きさに対する V、Δt を測定し、式 1.2 より Q_f を求め、図 1.12 に示す。

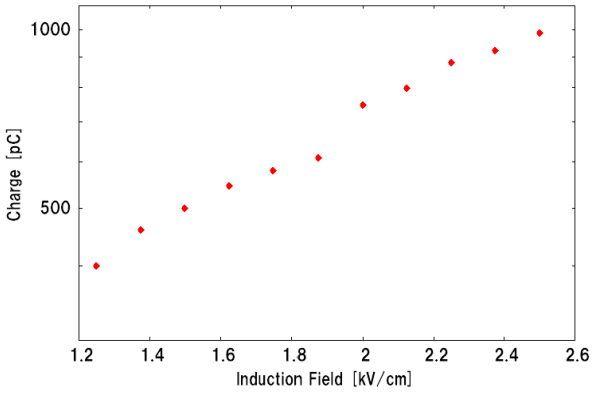


図 1.12: Induction Field に対する Q_f

図 1.12 の縦軸を、式 1.3 を用いて A_{GEM} に変換したものを図 1.13 に示す。

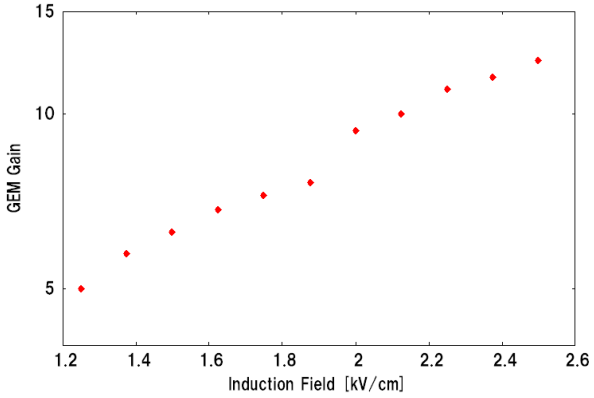


図 1.13: Induction Field に対する A_{GEM}

これより、Induction Field に対しても A_{GEM} は指数関数的に増加するといえる。

以上を踏まえ、以下の実験では特に断りの無い限り Drift Plane:-1600V、Anode:+500V、Cathode:GND、GEM TOP:-1250V、GEM BOTTOM:-1000V (ΔGEM:250V、Induction Field:2.5kV/cm) とする。

1.4.3 ゲインマップ

μ-PIC のアナログ出力は Anode 側、Cathode 側各々32ch ずつ足し合わせた 8×8 箇所から読み出すことができる。理想的にはこれら 64 箇所でのゲインが完全に一様であることが望まれるが、実際にはある程度のばらつきが存在する。この実験では、このばらつきがどれほどになるかを調べた。

実際に μ-PIC の全面でゲイン測定を行った結果を、横軸を Anode 方向、縦軸を Cathode 方向とし、各領域でのゲインの大きさを全体の自乗平均に対する割合として下図に示す。

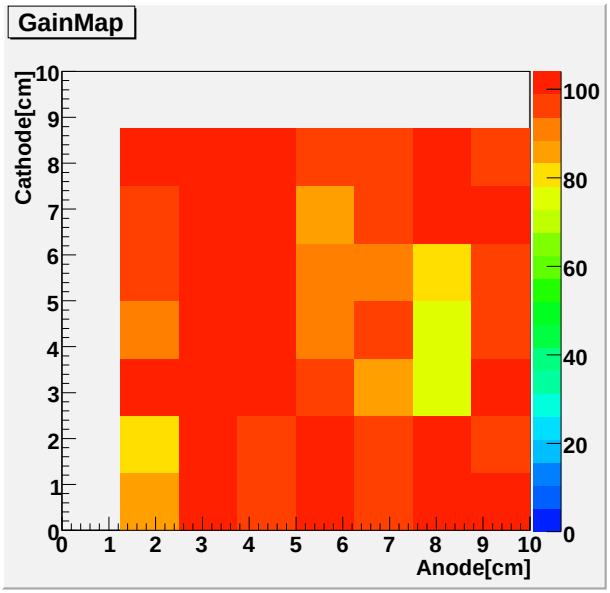


図 1.14: ゲインマップ

Anode 方向、Cathode 方向に 1 列つずつ存在する空白の部分は、検出器内部で GEM がずれて、GEM の外枠が μ-PIC の検出面の一部を遮ってしまったために生じた不感領域である。この部分を除いた 7×7 箇所でのゲインのばらつきを RMS で評価すると、ばらつきはおよそ 7% となった。10cm×10cm サイズの μ-PIC におけるゲインのばらつきは既に 4.5% 程度が達成されているが、比較的ゲインの低い箇所が数箇所存在するためにばらつきが大きくなってしまった。原因として、ゲインの低い領域内に、Anode-Cathode 間で放電が起こっているピクセルが多数存在することなどが考えられるが、実際には他のゲインの高い領域に比べてこのようなピクセルが特別に多く見つかったわけではなく、詳しい原因は特定できなかった。

2 Spectroscopy

2.1 実験原理

μ-PIC 内で X 線によって電離された電子は，Anode，Cathode 間に印加された電場によって Anode に捕獲され，また同時に Cathode には正イオンが捕獲される。これにより，Anode 側からは負の信号，Cathode 側からは正の信号が観測される。2 つの信号は同じ大きさであり，時間-電圧で書かれるパルスの面積が電離された電子の量，つまり X 線のエネルギーに比例する。したがって，十分な量のパルスを読み込むことによって，X 線のエネルギースペクトルを取得することができる。

2.2 実験方法

Anode または Cathode から出力される信号の波形を FADC(Flash Analog-to-Digital Converter) によって取得する。FADC は STOP 信号が入った時，そこからある時間を遡った情報を出力するため，信号を取得した時点からやや遅らせて STOP 信号を FADC に入力しなければならない。そこで，Anode と Cathode から信号が来た時，それぞれ Discriminator に通し，ある Threshold でデジタル信号化する。それらが同時に入力された場合のみデータを取得するため，Coincidence を通す。その後，Coincidence の出力信号を Gate Generator により 2 μ秒遅らせて FADC の STOP に入力する。FADC が STOP した直後，FADC がデータを読む前にさらに STOP 信号を入力させると挙動がおかしくなるため，Coincidence の出力を Discriminator の VETO に Gate Generator の Latch モードで入力し，Coincidence の信号が出力された後は，Discriminator から信号を出力させないようにする。ただし，連続して信号を取得するために，Interrupt Register から Latch を停止させる信号を Gate Generator に送る。

以上の配線図を図 2.1 に示す。今回の実験では，Discriminator の Threshold は Anode 側，Cathode 側共に 40mV とした。

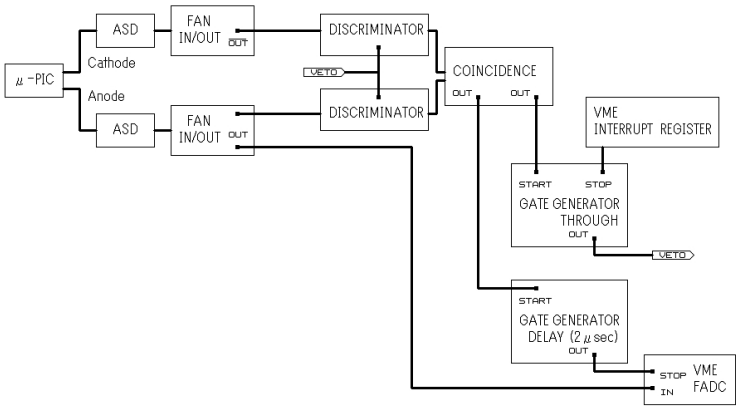


図 2.1: 配線図

さらに，ノイズや放電によって取得してしまったピークを取り除くため，プログラム側から

- ピークがデータの端にある。
- ピークの半値幅が広すぎる。
- ピークが FADC の STOP 信号から 5 μ秒以上遅れている。
- FADC のデータ量が 1000 を切る。
- ピークの高さがノイズと区別が付か無い程小さい。
- 複数のピークがある。

等の条件によりデータを篩にかけ，これらに該当するデータは削除した。
なお，線源は ^{55}Fe と ^{109}Cd を用いた。

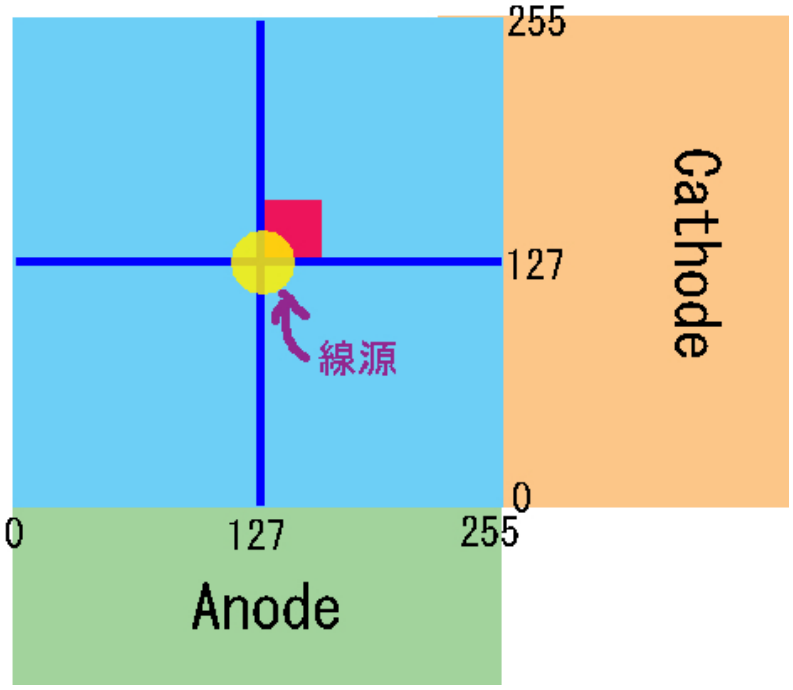


図 2.2: 設置図

2.3 結果・考察

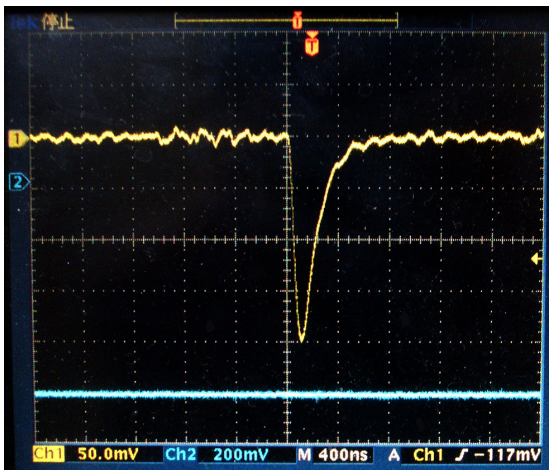


図 2.3: オシロスコープで確認した信号

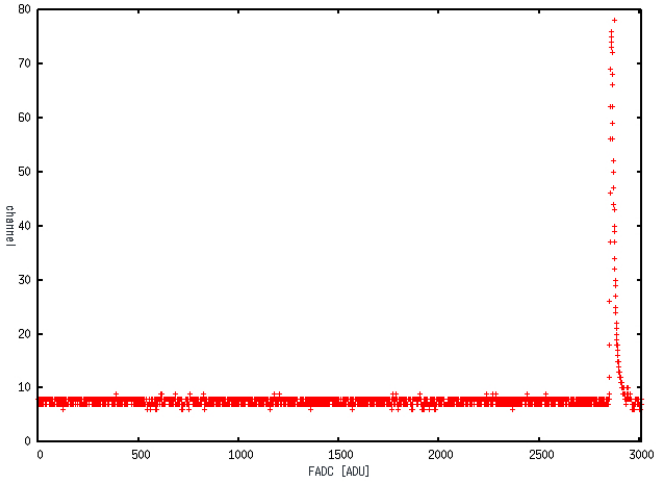


図 2.4: FADC で取得した信号

Drift plane，GEM top，GEM bottom，Anode にそれぞれ-1600V，-1250V，-1000V，+480V をかけ，図 2.2 に示すように，線源を中央に設置しμ-PIC 上で Anode，Cathode ともに 128ch から 160ch，12.5mm 四方で得た信号を足し合わせ，上記の実験を行ったところ，図 2.4 に表されるような信号の波形が FADC で取得された。

パルスの面積が X 線のエネルギーに比例することから，FADC で得られたピークの周りを積分し，その値をヒストグラムにすることで線源のスペクトルが得られる。解析する際にはピークの 100nsec より前までのデータの平均により Base Line を定め，ピークの前 100nsec から後 500nsec の間で Base Line を除いた値について積分を行った。なお，FADC の時間分解能は 10nsec，電位の分解能は 1/256[V] であるから，積分された値に対し

$$ASD \text{ までの信号 } [pc] = FADCcharge[ADU] \times (10[ns]) \times (1000/256[mV]) \div (50[\Omega])$$

が成り立つ。

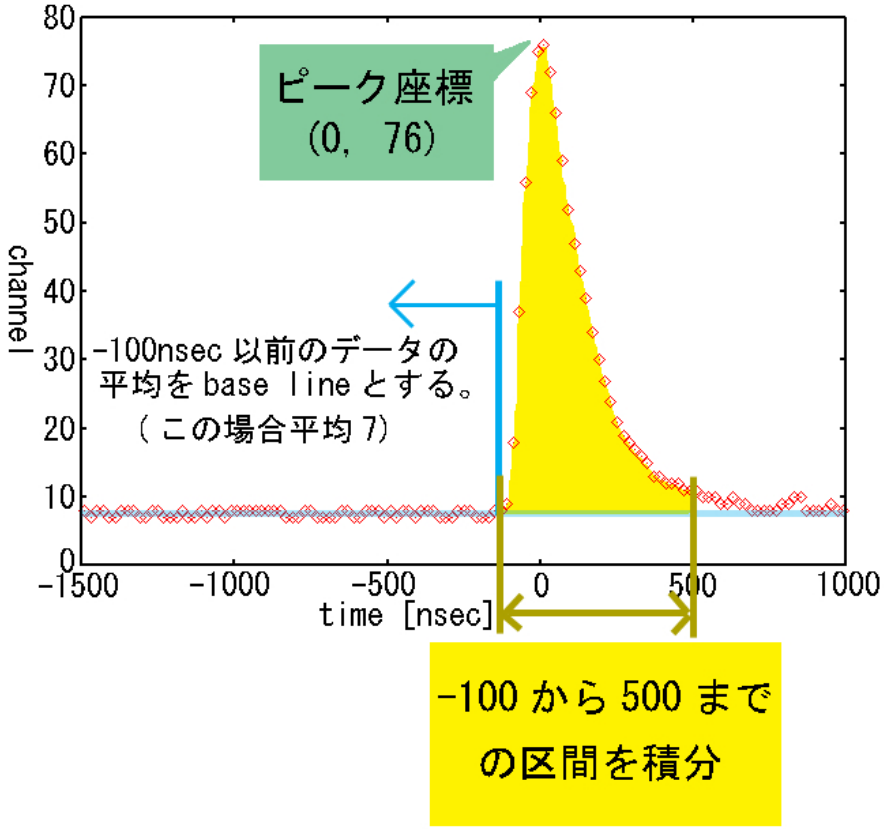


図 2.5: 積分方法

^{55}Fe，^{109}Cd 共に 4000 パルスのデータについて，上記の積分値によるヒストグラムをそれぞれ図 2.3，図 2.3 に示す。

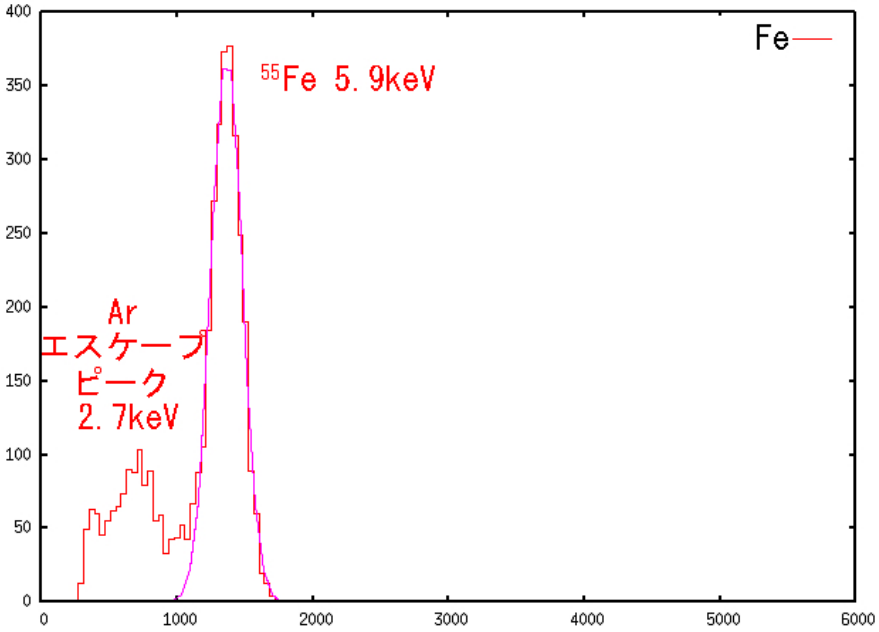


図 2.6: ^{55}Fe のスペクトルと Gaussian による fitting 例

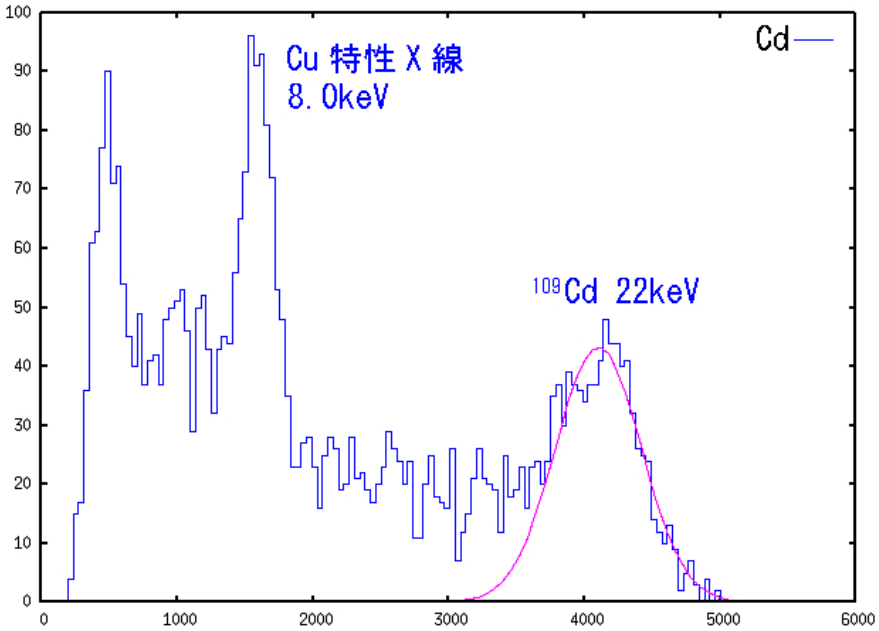


図 2.7: ^{109}Cd のスペクトルと Gaussian による fitting 例

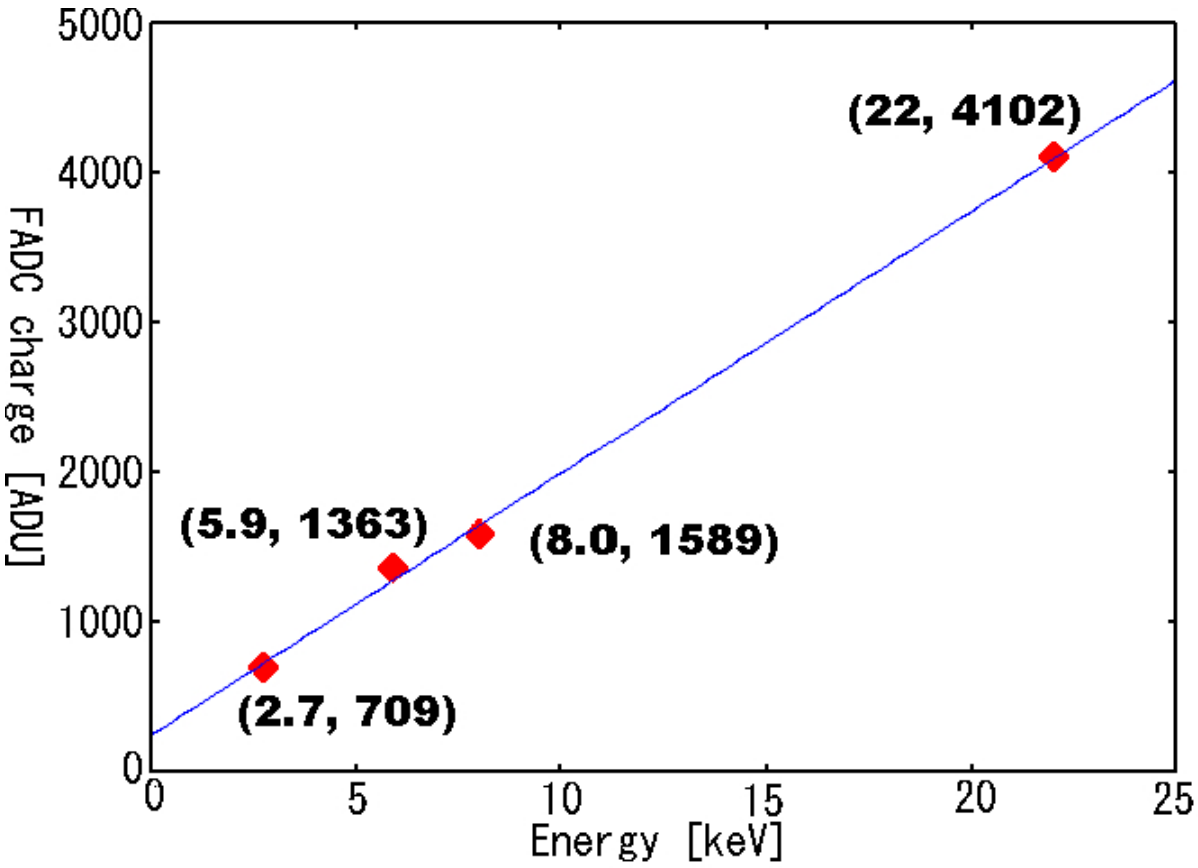


図 2.8: エネルギー較正

^{55}Fe のスペクトル中には，^{55}Fe 自体の放出する 5.9keV のピークと μ-PIC 中に封入している Ar のエスケープピーク (5.9 - 3.2)keV，また ^{109}Cd のスペクトル中には ^{109}Cd の放出する 22keV のピークと μ-PIC の基板に用いられている Cu の特性 X 線である 8.0keV が確認できる。4 つのピークに対して Gaussian による fitting を行い，それらを (integral-energy) のグラフ上にプロットすると図 2.8 に示すように，一直線上に並ぶことがわかる。4 点を最小二乗法により直線で fitting した式は

$$FADCcharge[ADU] = 175 \times (Energy[keV]) + 256$$

である。

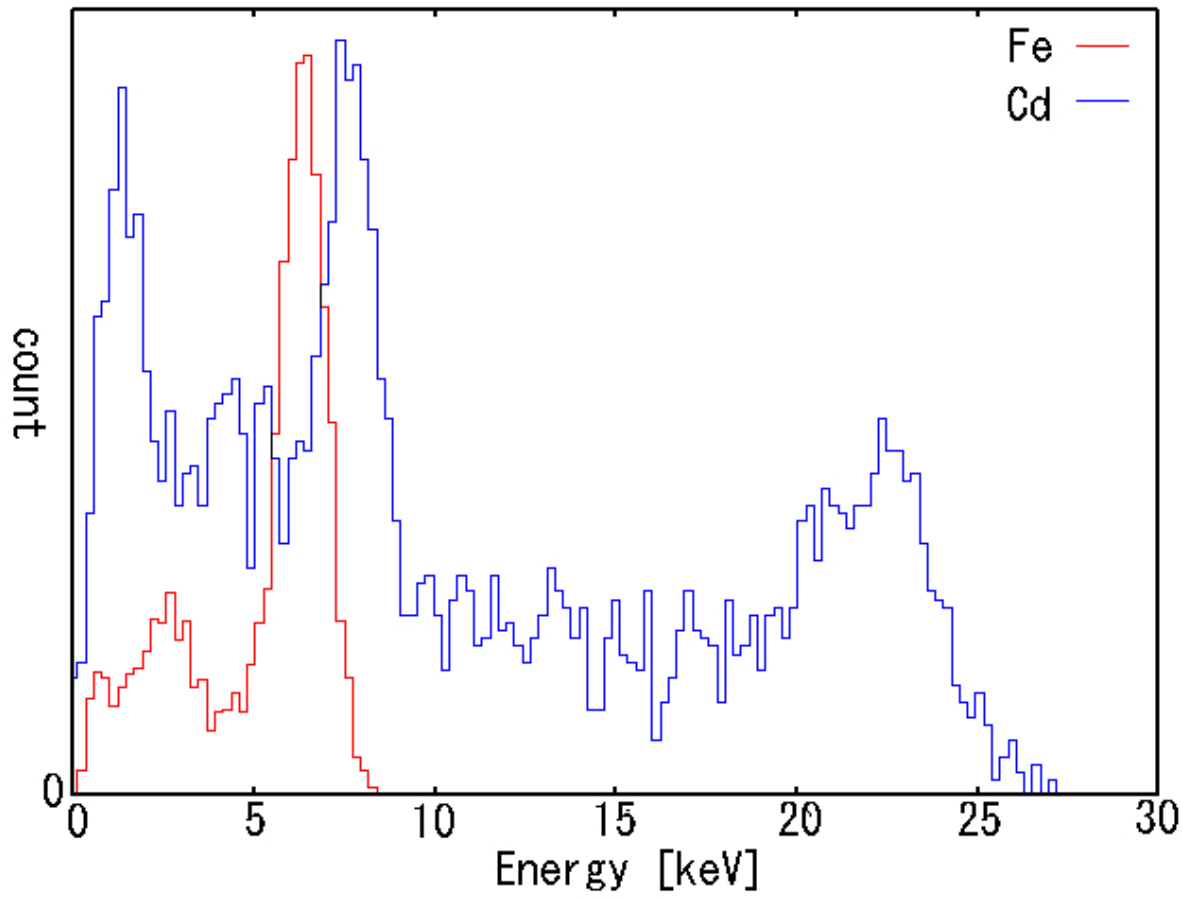


図 2.9: ^{55}Fe, ^{109}Cd のスペクトル

これを元に図 2.3，図 2.3 の横軸を keV に書き直したものが図 2.9 である。

エネルギー分解能は FWHM で，2.7keV が 75.3%，5.9keV が 26.1%，8keV が 26.4%，22keV が 19.4%であった。この 4 点を (Resolution-keV) のグラフに取ったものが図 2.10 である。これにより power law の冪は-0.5 とはならなかったが，エネルギーが高いほど分解能が良くなる傾向がわかる。

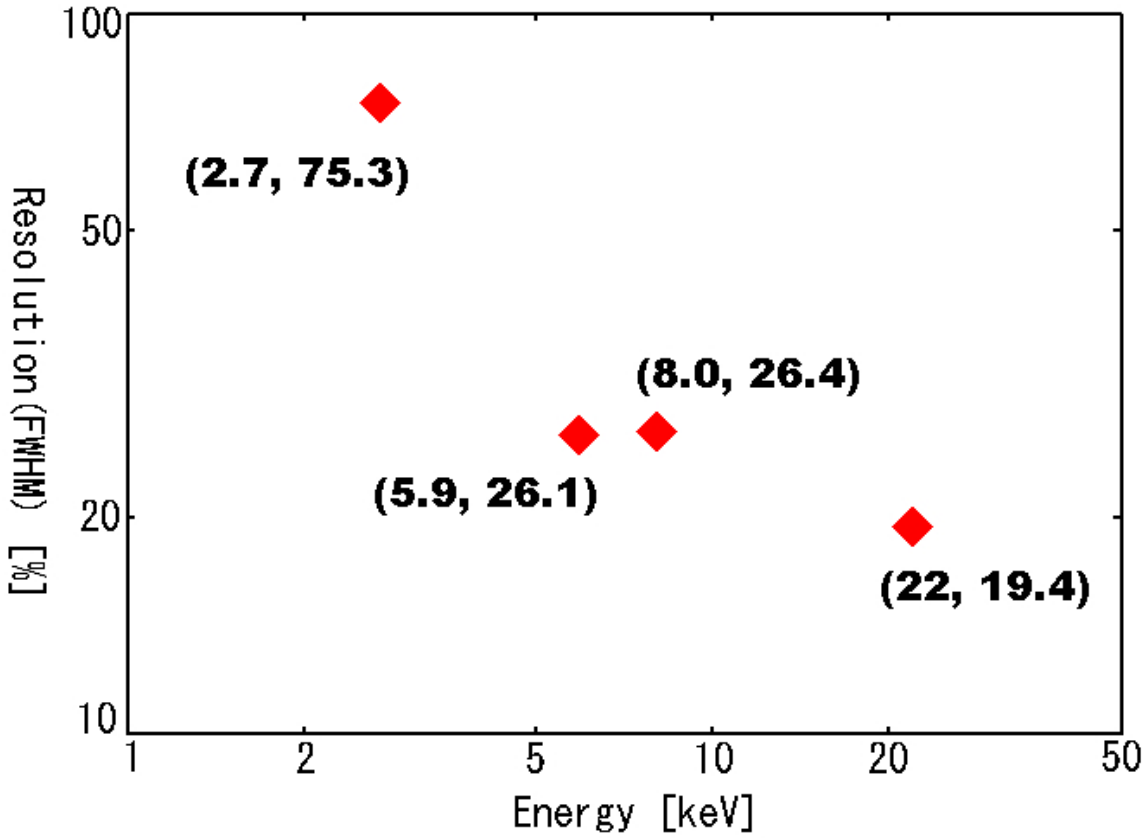


図 2.10: エネルギー分解能

3 2-D imaging

3.1 実験原理

μ-PIC 内では，Anode，Cathode からの信号はそれぞれ 256 本の，互いに垂直に交差するストリップ電極から個別に読み出され、各々が ASD に出力される。256ch のうち，Anode，Cathode に同時に出力された信号は直交座標上の位置として扱うことが出来る。例えば金属製のテストチャート等をμ-PIC と放射線源の間に置いた場合，X 線が金属に吸収された部分はμ-PIC 上で検出されず，2 次元ヒストグラムを書くとその部分はほぼ 0 となる。また金属の無い部分を通過した X 線は信号として検出され，2 次元ヒストグラム上では X 線の量に応じた値となる。このようにしてμ-PIC で 2 次元の像を得る事が出来る。

3.2 実験方法

2 次元画像を取得する際の配線図を図 3.1 に示す。Anode，Cathode それぞれにおいて ASD に入力された信号 256 × 2 の信号は，増幅された後ある閾値でデジタル信号に変換される。そのデジタル信号を Encoder board 内で 100MHz で同期し，Anode 側，Cathode 側の Coincidence を取る。各中心座標を時間情報と共に VME 上の Memory board へ出力し，PC に取り込む。

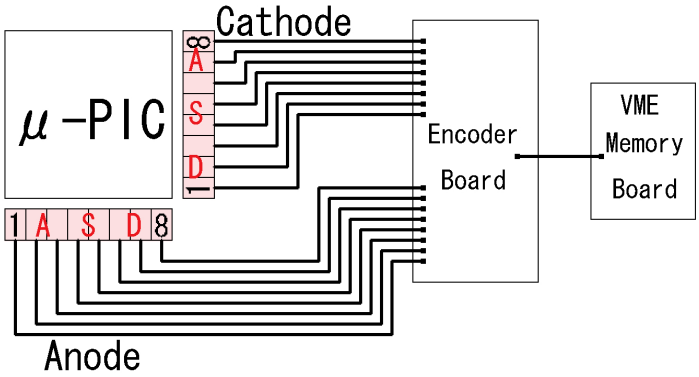


図 3.1: 配線図

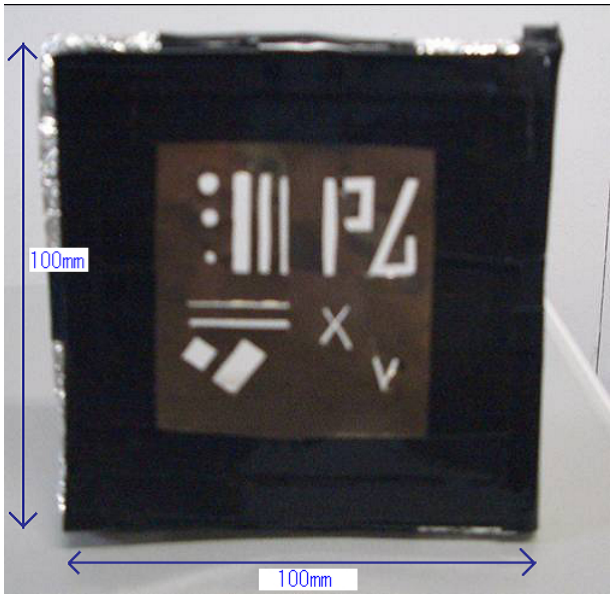


図 3.2: テストチャート

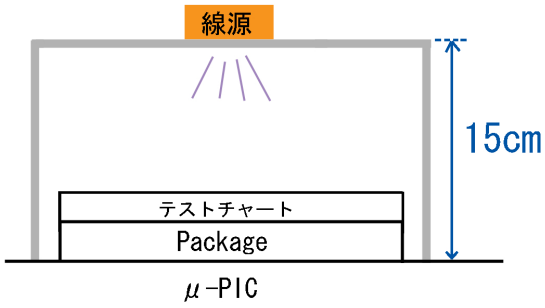


図 3.3: 設置図

今回の実験においては位置分解能を知る為に，図 3.2 の Ti 製 50 μ m 厚，μ-PIC と同じ大きさの 10cm × 10cm の自作テストチャートを用いた。また図 3.3 に示すように線源を μ-PIC から 15cm 離し，μ-PIC の中央真上に位置するように設置し，テストチャートを Drift Plane の上にのせた。Drift plane，Gem Top，Gem Bottom，Anode にはそれぞれ-1600V，-1250V，-1000V，+500V を印加し，Ar:C_2H_6=9:1 のガスでパッケージを満たした。線源は ^{55}Fe を用い，また event 数は 100000count とした。

3.3 結果・考察

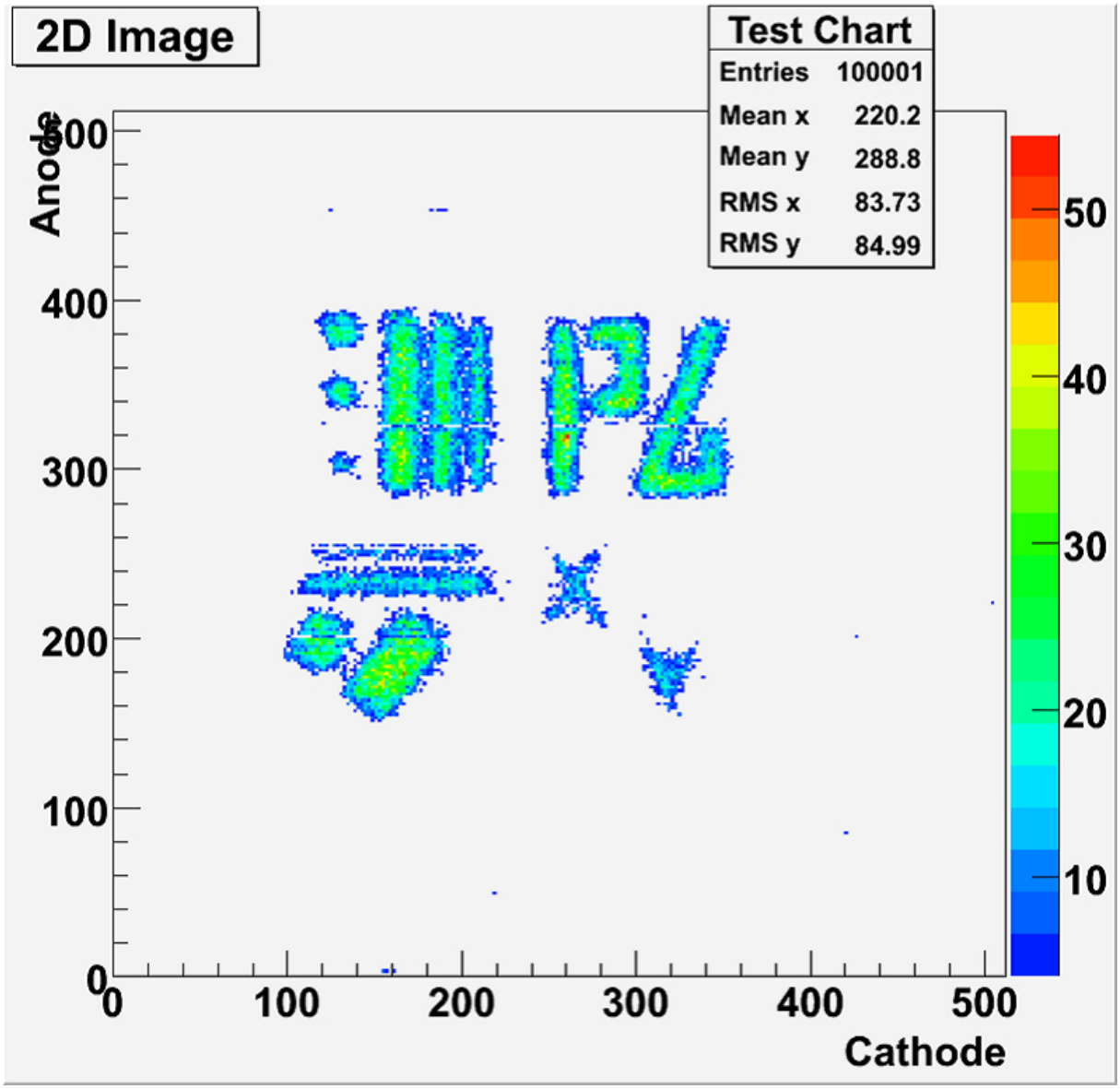


図 3.4: 2D image

上記の条件の結果，図 3.4 の鮮明な画像が得られた。ただし，図 3.4 では Background を無視するため，4count 未満を切り捨ててある。

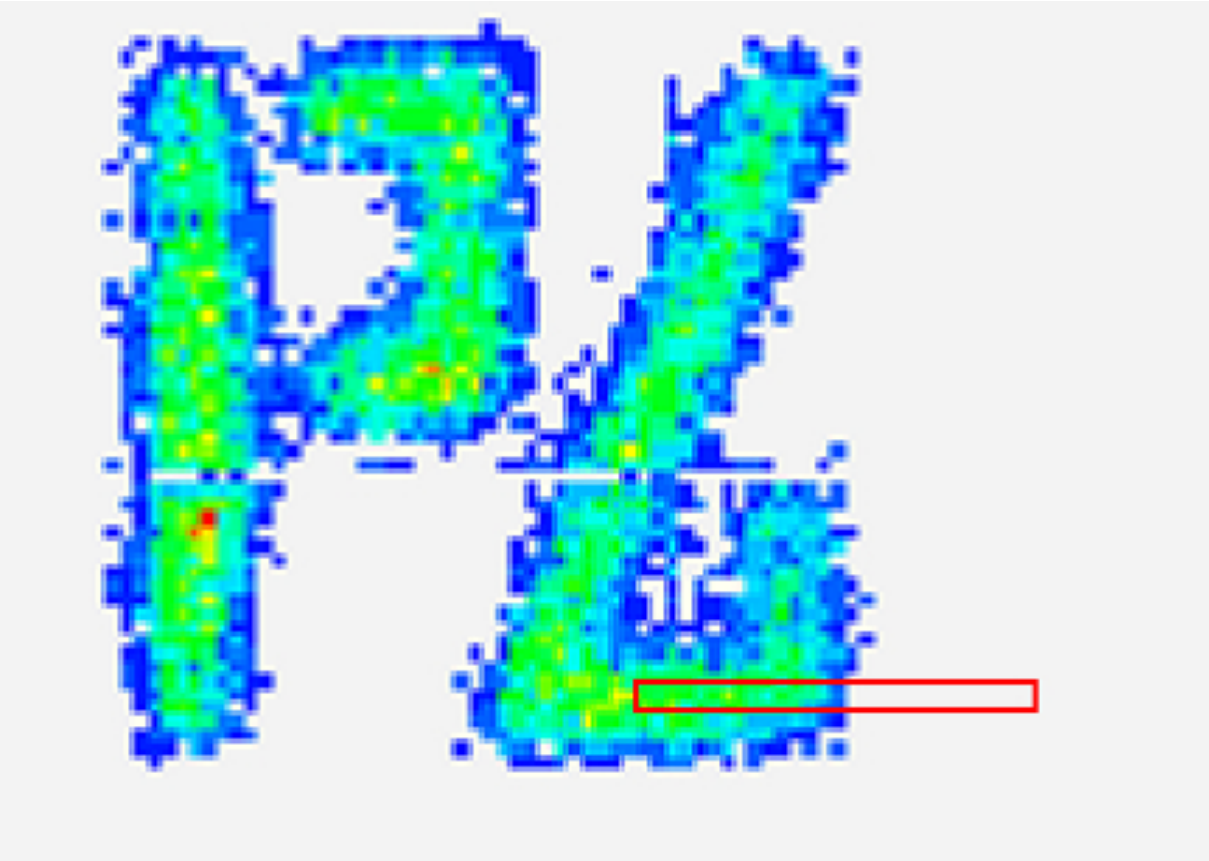


図 3.5: 解析部位

分解能を解析するため，図 3.5 で赤く囲んだ Anode3ch，Cathode30ch 分の部位を縦方向に projection し，得られた Cathode 座標の一次元ヒストグラムに対し誤差関数を用いて fitting を行った。その一次元ヒストグラムを図 3.6 に示す。

得られた誤差関数から分解能は σ ～560 μ m となった。

μ-PIC の位置分解能は公称値では最高 σ ～120 μ m であり，今回の結果は位置分解能としてやや悪い物となった。主な原因として，

- 1cm パッケージ内での電子の拡散
- テストチャートと線源の配置
- 自作テストチャートの正確さ

などがあげられる。

要因 1 については、Drift plane と Gem Top 間での約 580[V/cm] の電場，Ar ガス中という今回の条件ではσ〜500 μ m の拡散が起こる。従って対策としては，1mm などの薄いパッケージを用いる，Kr，Xe などの原子量の大きく拡散の小さいガスを用いる等が考えられる。

要因 2 はテストチャートの解析を行った部分が線源の真下ではなかった事に起因し，X 線の吸収が Drift plane から 1mm で完了したとするとσ〜200 μ m となる。

要因 3 は自作である為多少の誤差はやむを得ないが，以上の誤差評価から、誤差伝播の法則を用いて、Anode 方向 1.2mm の間においてはσ〜200 μ m の精度で作成出来ていることが分かる。

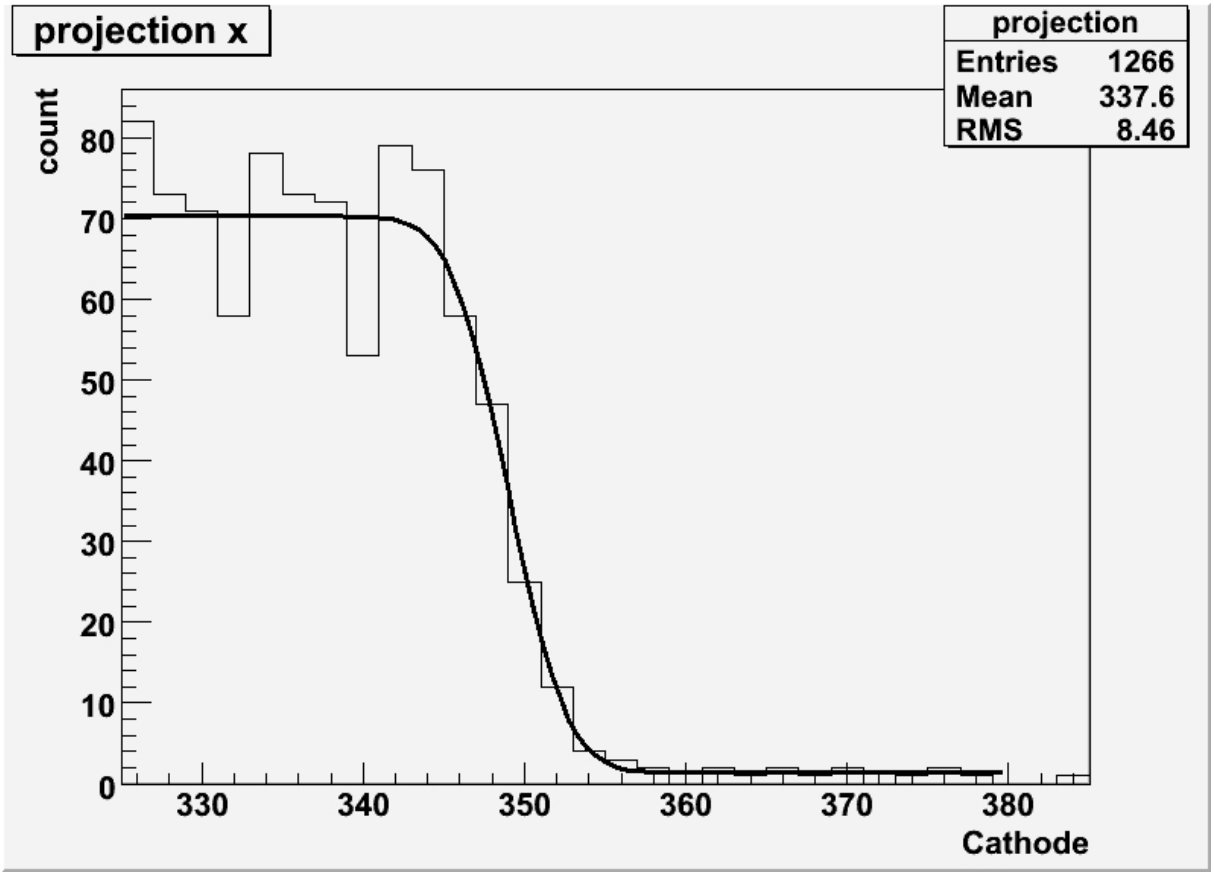


図 3.6: fitting

4　cool-X の性能評価



図 4.1: cool-X

4.1　cool-X の原理

cool-X は大まかに焦電素子と加熱・冷却装置とからなる。焦電素子が加熱されたときは，上面がプラスに帯電し低圧ガスで満たされたパッケージ中を電子が加速される。この電子が焦電素子の上面に衝突する際に制動放射と，焦電素子上面にコートしてある Ta の特性 X 線が発生する。

焦電素子が冷却された時には，上面はマイナスに帯電する。パッケージ中に飛び出した電子の内，Be 窓方向に飛んだものは Be 窓の内面にコートしてある Cu に衝突し，その際制動放射と Cu の特性 X 線が発生する。この原理図を図 4.2 に示す。

2 つの phase を heating phase と cooling phase とすると，それぞれ約 40 秒，65 秒で交互に繰り返し，これらは cool-X 本体に付いている LED の色により見分けられる。

図 4.3 は制動放射成分のスペクトルに注目したものである。また図 4.4，図 4.5 はそれぞれ heating phase, cooling phase におけるスペクトルである。これら 4 つは製作会社の提供する資料である。

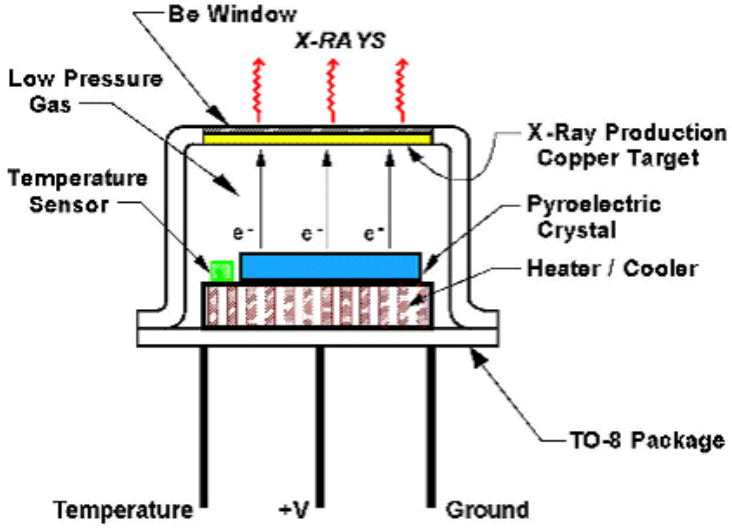


図 4.2: cool-X 原理図

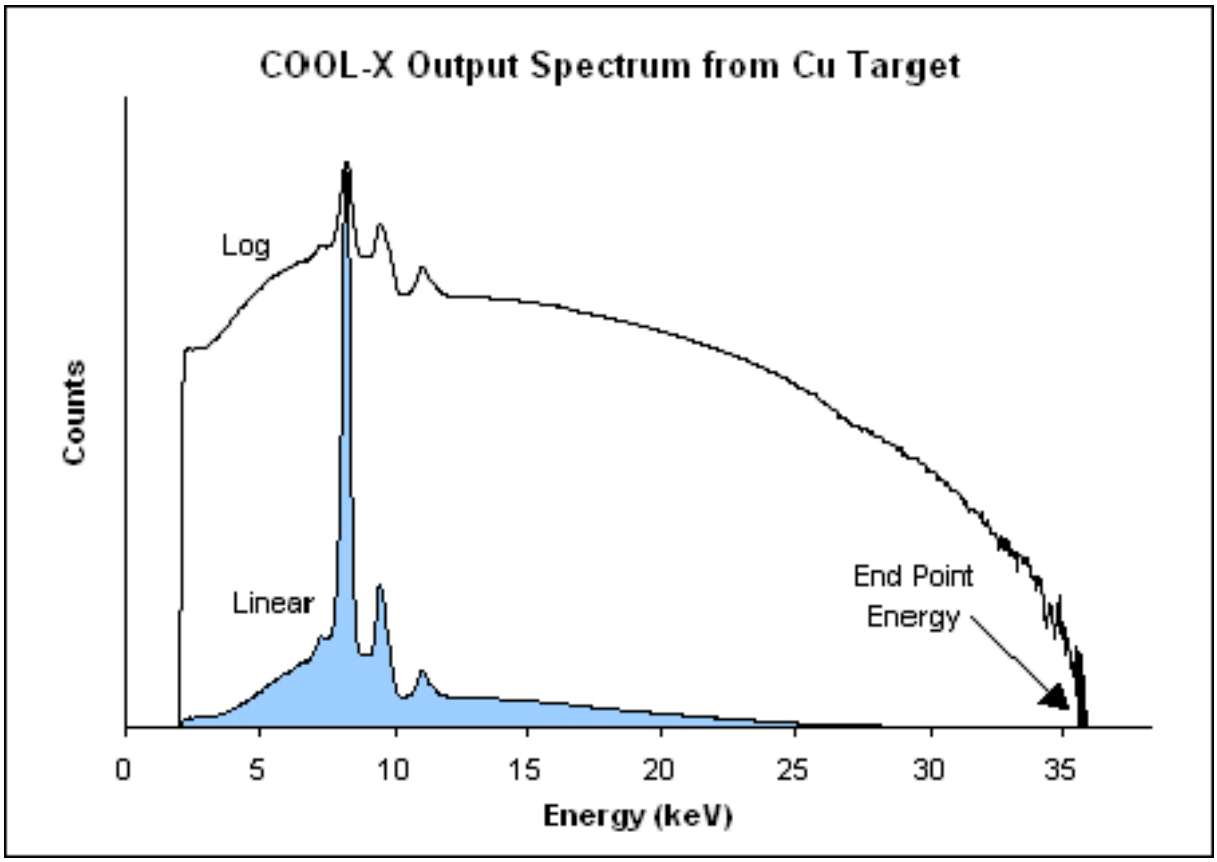


図 4.3: 連続成分のスペクトル

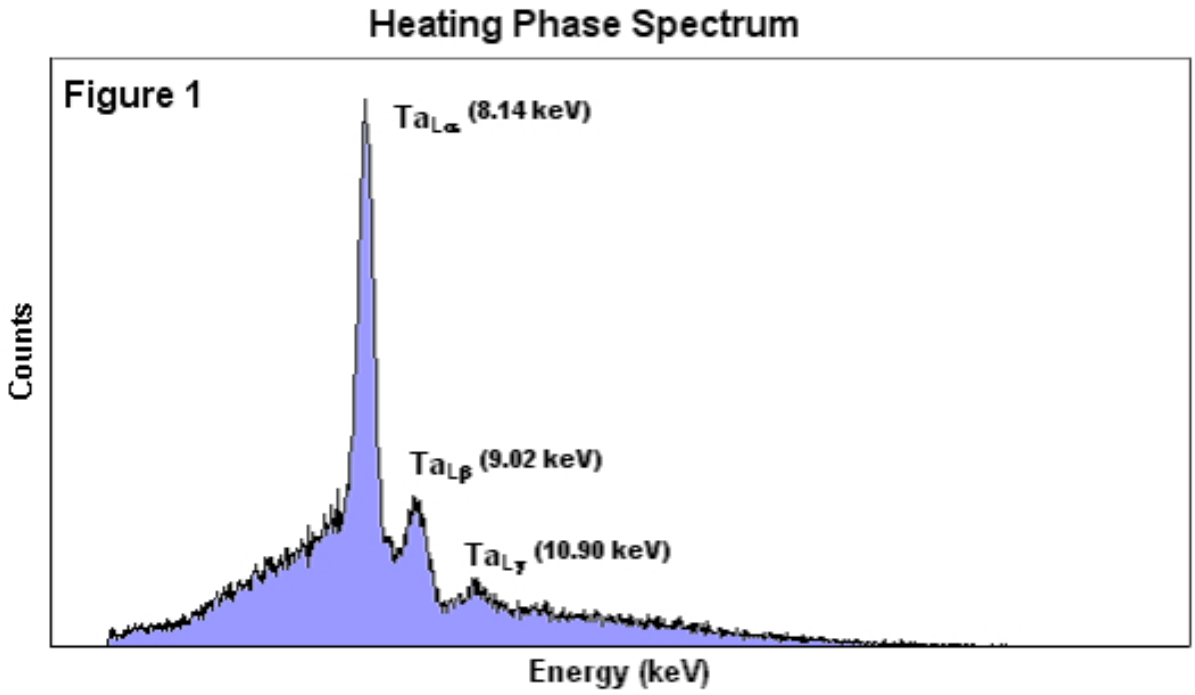


図 4.4: Heating Phase Spectrum

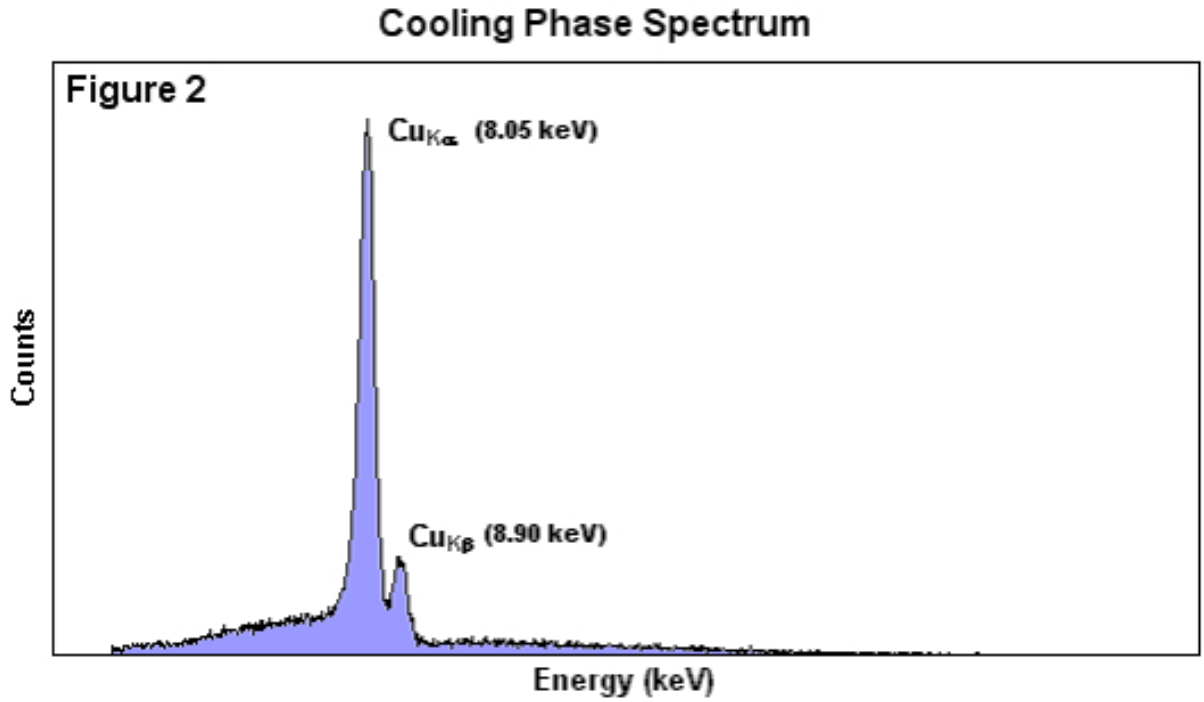


図 4.5: Cooling Phase Spectrum

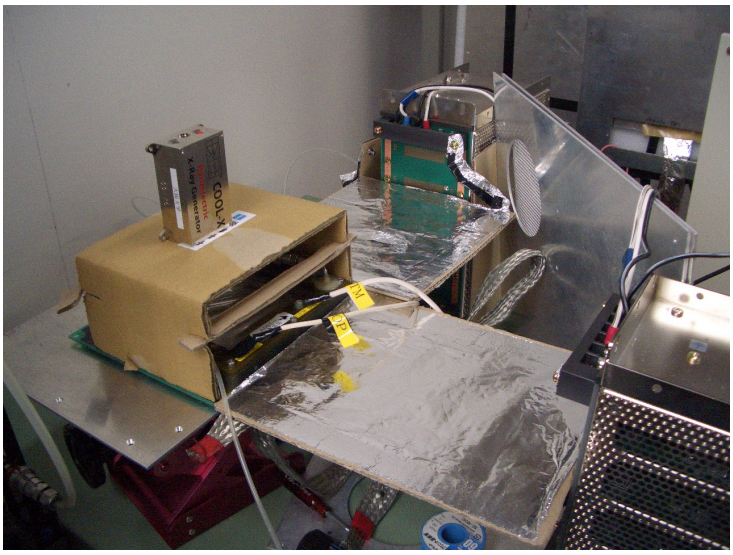


図 4.6: 設置した様子

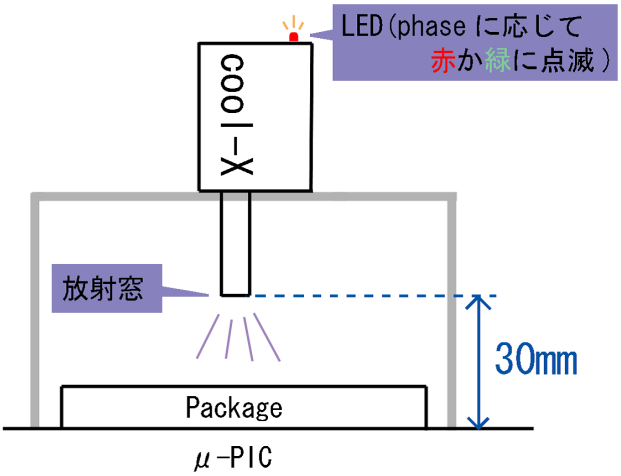


図 4.7: 配置図

Cool-X は放射窓が鉛直かつ μ-PIC に垂直に，μ-PIC の中央に当たるように設置し，rate とスペクトルを取得する際には図 4.7 に示すように μ-PIC と放射窓の距離を 30mm とした。

4.2 rate の時間変化

rate の取得には，2D imaging の際に用いた Encoder Board を使用した。Encoder board は Anode，Cathode から得られる信号を 100MHz で同期し，28bit clock によって時間を測定した後，memory board にその上位 10bit を送る。従って，memory board に特定数のパルスを取得させた時，その最初から最後までに取得したパルスの時間幅（時間分解能 2.62msec）が記録されるため，rate を測定することが出来る。

このようにして取得した rate の変化が 4.8 である。本体付属の LED の色から読み取った phase の変化から，rate と phase に相関があることが分かる。この時，1 回の取得パルス数は 2000 とした。

図 4.8 の rate の変化を 0.5sec 毎に 1 周期で folding したグラフが図 4.9 である。

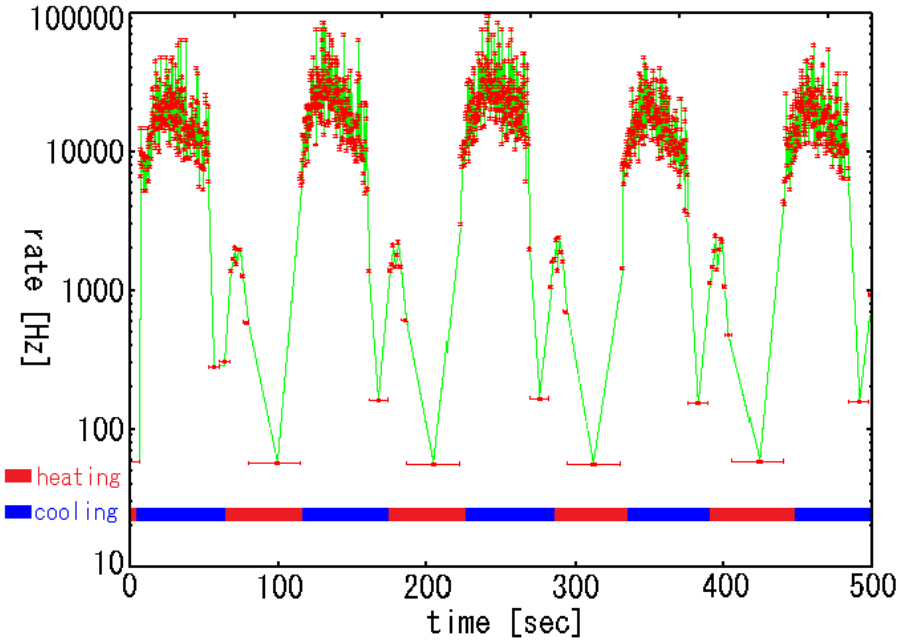


図 4.8: rate の時間変化

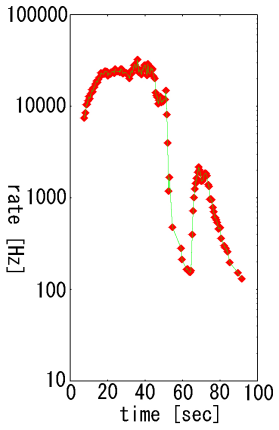


図 4.9: 1 周期の rate 変化

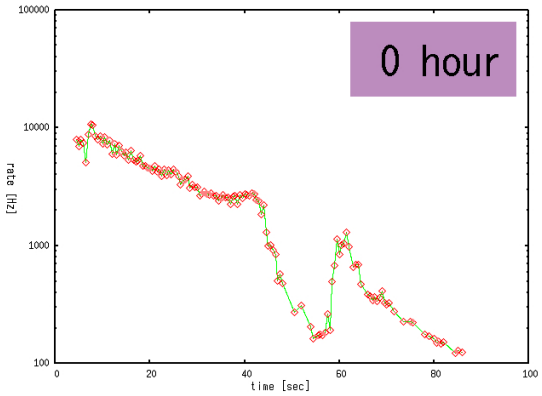


図 4.10: 0 時間

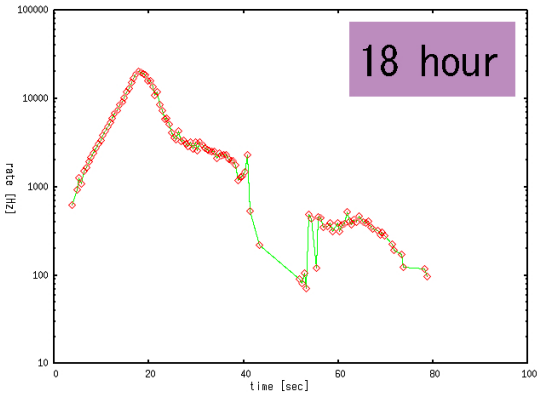


図 4.11: 18 時間

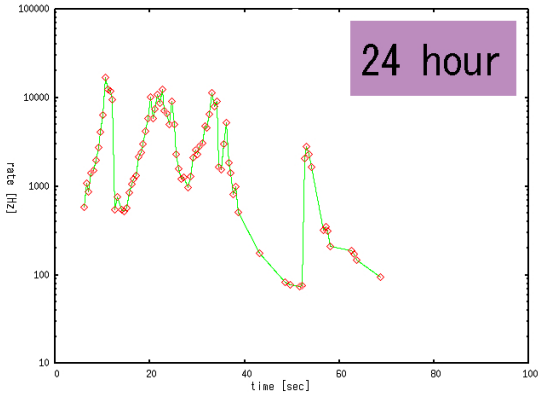


図 4.12: 24 時間

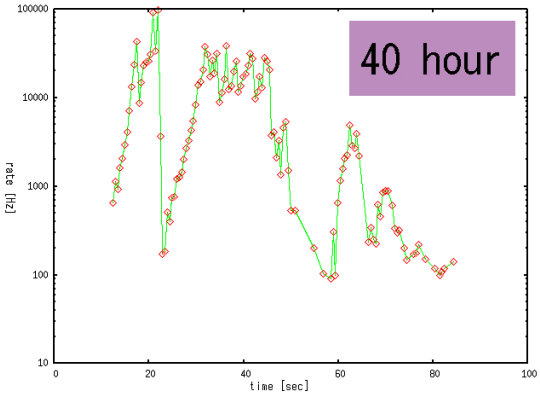


図 4.13: 40 時間

また製造元によると Cool-X は寿命が短く，連続使用では 200 時間程度しか正常に動作しないとのことであり，その様子は μ-PIC による測定中においても確認できた。その様子を図 4.10 から 4.14 に示す。添えてある時間は連続使用時間ではなく，0hour の時刻を基準とした μ-PIC でののべ使用時間数である。使用時間の経過に伴い、形状が崩れる，cooling phase と heating phase が判別できなくなるなどの変化が起こることが分かる。

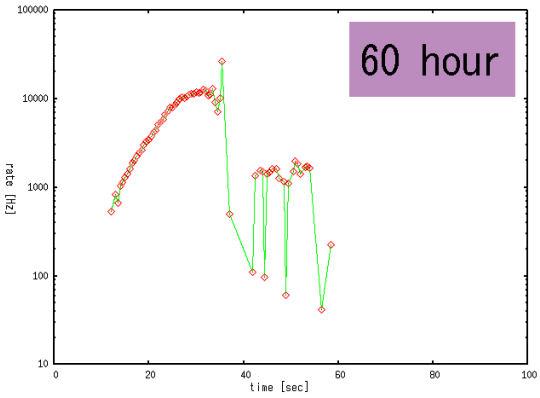

60 hour
100000
10000
1000
100
10
rate [Hz]
0
20
40
60
80
100
time [sec]

図 4.14: 60 時間

4.3 spectrum

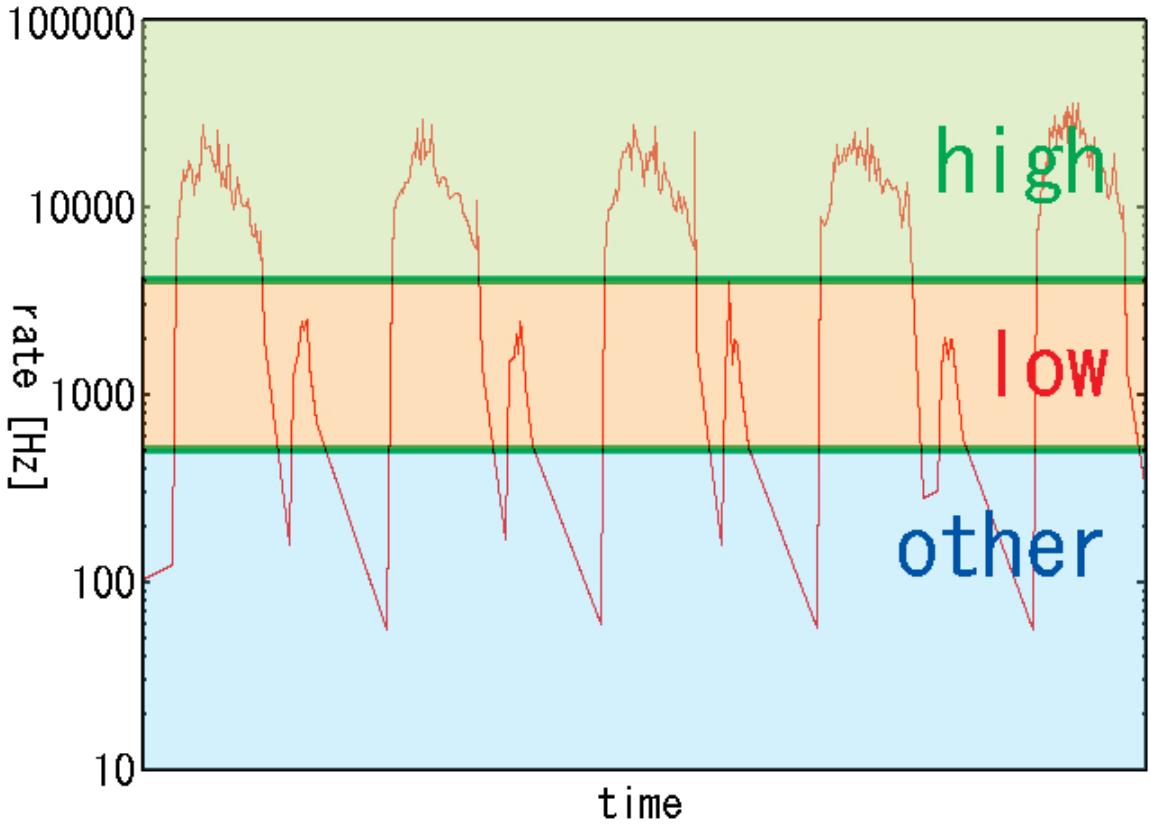


図 4.15: phase 毎の rate の値

図 4.8 から，rate の値によって phase を区別出来ることがわかる。rate の変化が数十秒，スペクトル取得に掛かる時間が数百 msec であることから，rate の測定結果が得られた直後にスペクトルを測定し，rate によってそのスペクトルを分けることを考える。図 4.15 に示すように rate を 3 段階に分け，heating phase と cooling phase での rate のピークでのみスペクトルを取得するようにする。

スペクトルを取得する位置を図 4.16 に示す。後述するように，rate を抑える目的で，中心からやや離れた Anode，Cathode ともに 64ch〜80ch の 6.3mm 四方の信号を足し合わせて取得した。

このようにして取得した phase 毎のパルスから得たスペクトルを，線源 ^{55}Fe を用いてエネルギー較正したものが図 4.17 である。

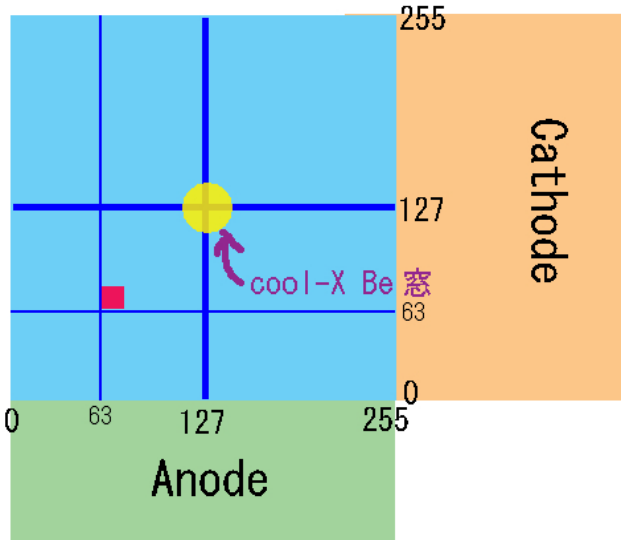


図 4.16: 配置図

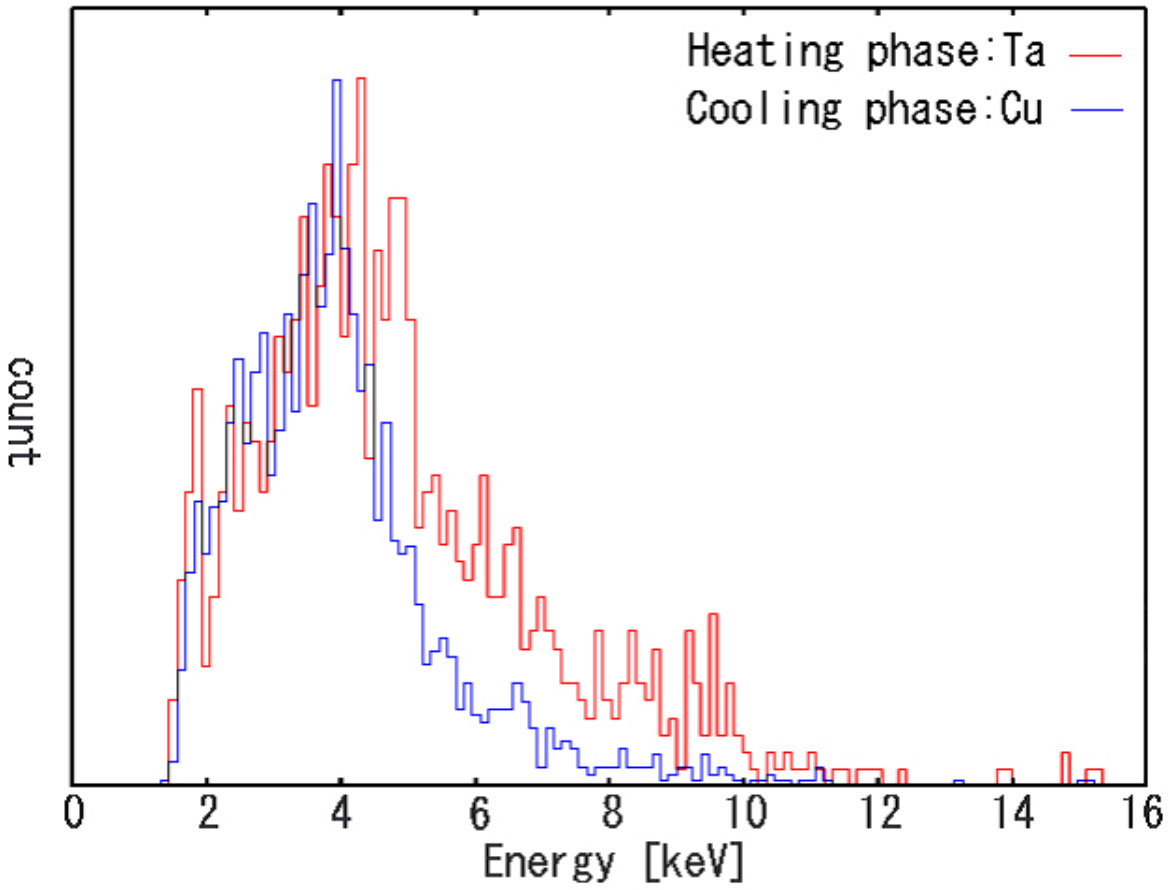


図 4.17: Cool-X からのスペクトル

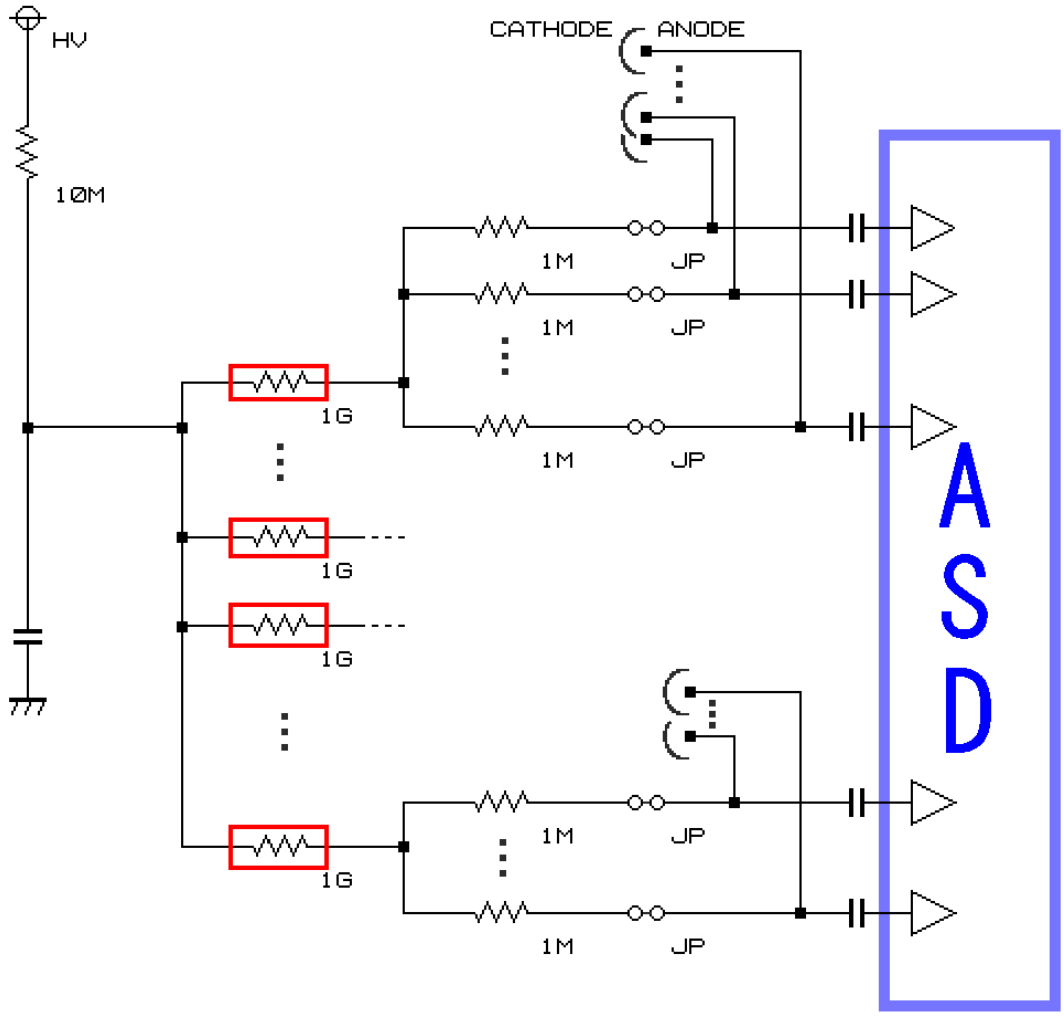


図 4.18: Anode，HV 周辺回路

図 4.4，図 4.5 からも分かるように，Ta と Cu の特性 X 線は共に 8keV 付近にピークを持つ。従って図 4.17 において，μ-PIC のゲインが低くなっていることが分かるが，これは Cool-X の rate が非常に大きくなる事に起因する。

Anode，HV から ASD の周辺回路を図 4.18 に示す。図 4.18 にあるように Anode と HV の間には信号の回り込みを防ぐ抵抗が設置してあるが，この抵抗上では rate が低い場合には無視できる電圧降下が生じる。rate が高い場合には，抵抗に流れる電流が増加し Anode に本来の電圧がかからず，ゲインが低下する。

これを回避するために図 4.18 の赤く囲んだ 1G Ωの抵抗をショートさせ，電圧降下の量を減少させた。

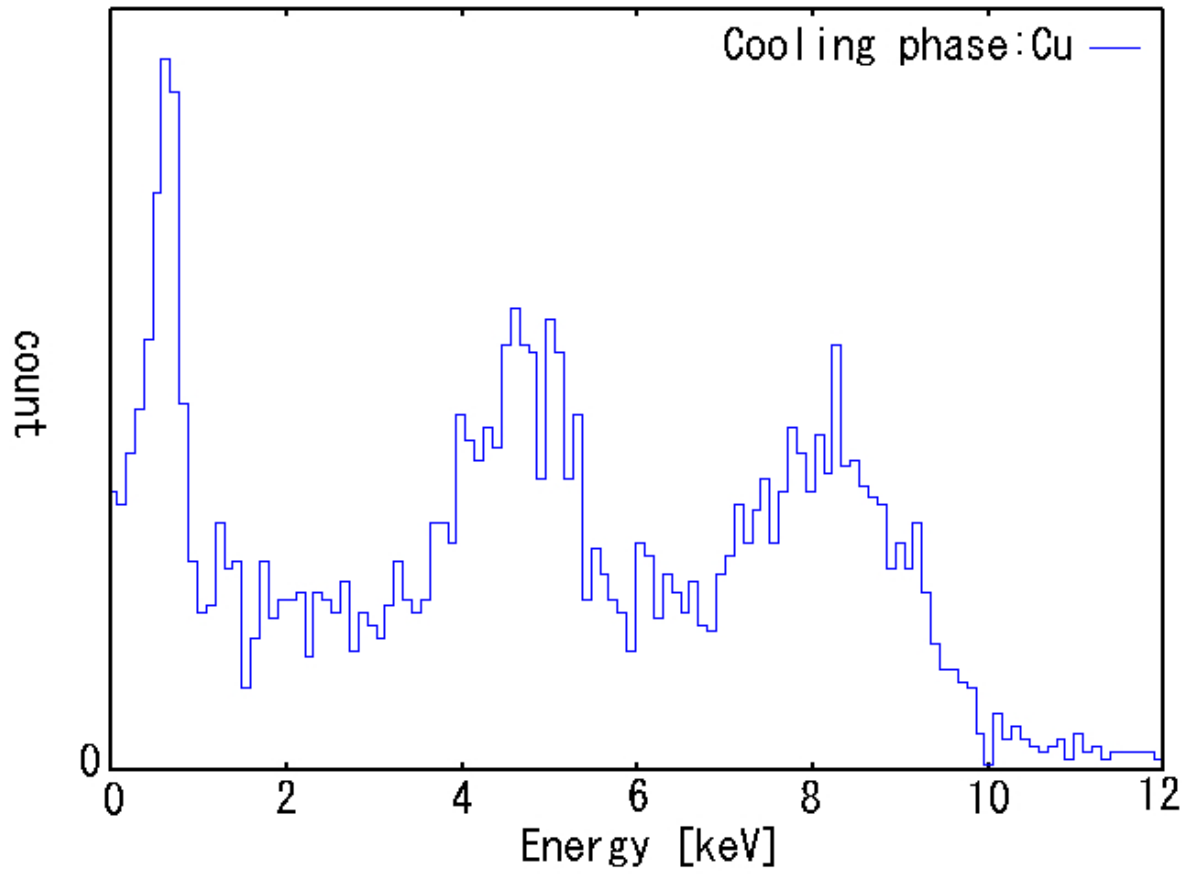


図 4.19: Cool-X からのスペクトル

このようにして同様の位置で 12.5mm の信号を足し合わせて phase に応じて分けずに取得したスペクトルを図 4.19 に示す。ただし，このスペクトルを取得する際においては図 4.14 からもその兆候が出ているように，ほぼ cooling phase でしか信号が確認できなかったので，結果的に Cu の特性 X 線のみを取得したと考えてよい。8.0keV に位置するピークは Cu の特性 X 線であり，4.7keV のものは Ar のエスケープピークである。分解能は FWHM でそれぞれ 30.2%，46.2% であった。

4.4 beam の拡がり

図 4.20 のように Cool-X の放射窓を μ-PIC の検出面の中央部分に向け、DriftPlane に対して垂直かつ鉛直に Cool-X を固定した。このとき Cool-X の放射窓から DriftPlane までの距離を L として、L=3cm の場合と L=8cm の場合について event 数が 500000count となるまで、μ-PIC の検出面全面において 2-D image を測定した。ただし、Cool-X の 2 つの phase によるデータの選別は行わなかった。

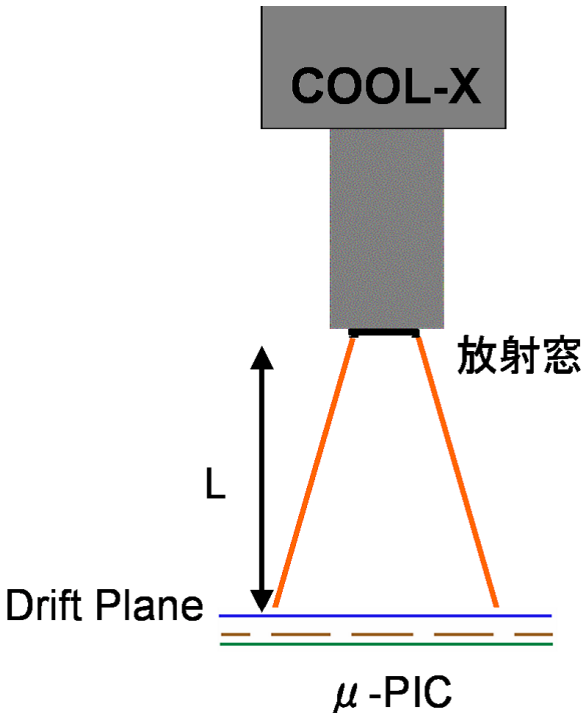


図 4.20: 配置図

結果としてそれぞれ図 4.21、図 4.22 のような画像を得た。

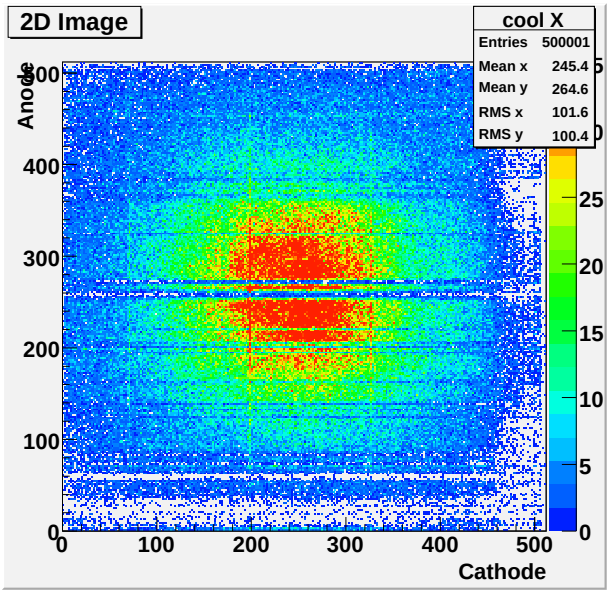


図 4.21: L=3cm の場合

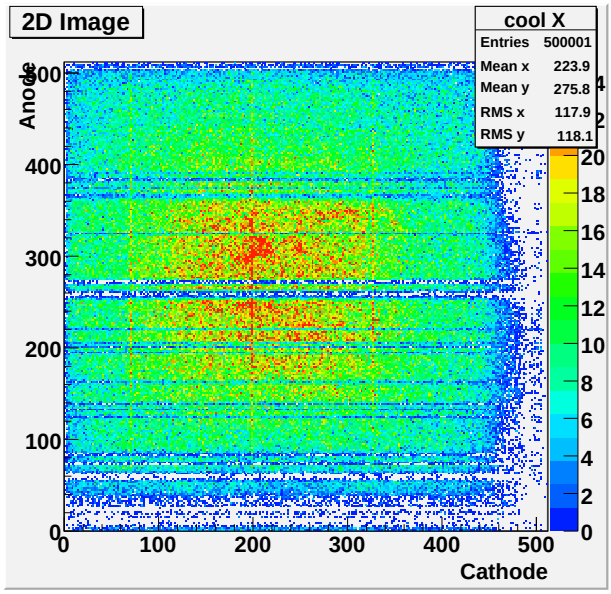


図 4.22: L=8cm の場合

図 4.21、図 4.22 より、それぞれ beam の中心と思われる地点から遠ざかるに従い、count 数が徐々に減少している。これを詳しく解析するため、画像の領域を beam 中心を中心とする幅の等しい円環領域に分割し、各々の円環での単位面積あたりの count 数を計算した。これを半径に対して plot し、片対数で表したものを図 4.23 に示す。ただし、縦軸は最高値に対する割合として規格化してある。

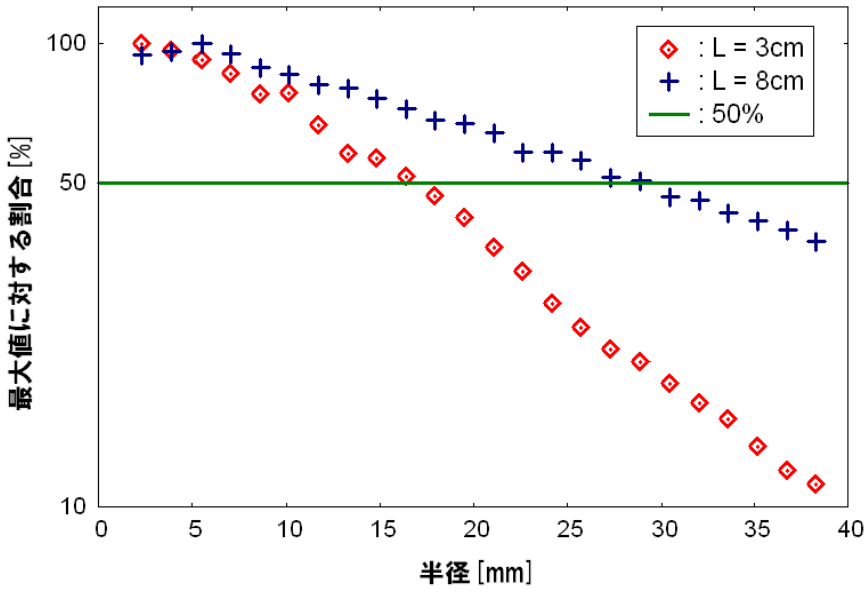


図 4.23: beam の拡がりの L による違い

今回の実験では、beam の拡がりの評価方法として、単位面積あたりの count 数が最高値に対して 50% に落ちる地点を基準とした。この地点から beam 中心までの距離を R とすると、各 L に対して R の値は表 4.1 のように得られた。

L[cm]	R[mm]
3	16
8	29

表 4.1: 各 L に対する R の値

線源が点源であると仮定すれば、強度は逆二乗則にしたがって減少するので、R の値は L に比例して変化する筈であるが、R の変化はそのような予想に比べて小さい。これは線源が有限の拡がりを持ち、且つ L がその拡がりに対して十分には大きくないためにこの拡がりを無視することができないためだと考えられる。実際、L=3cm の R と L=8cm の R を直線で結ぶと、L=0cm すなわち放射窓の地点ではこの直線の beam 中心からの距離は約 7mm となり、半径が約 9mmBe 窓の大きさと同程度となった。

また、図 4.4 の結果を用いて beam 中心における Flux を計算した。単位面積あたりの count 数をこの測定時間で割ることで、Flux が算出できる。L=3cm、L=8cm の各々の場合の測定時間はそれぞれ 204sec、44sec であった。この測定を行ったときの Cool-X の rate 変化の周期はおよそ 100sec であったから、L=3cm の場合に関しては beam 中心の平均 Flux を評価することができ、約 150[photons/cm^2/sec] という結果を得た。しかし、L=8cm の場合は測定時間が Cool-X の周期に比べ短く、平均 Flux を評価するには妥当ではない。

次に、Cool-X の 2 つの phase に関して beam の拡がりがどのように異なるかを測定した。測定方法としては、先ほどと同様に図 4.20 のように Cool-X および μ-PIC を配置し、Cool-X の放射窓と μ-PIC の Drift Plane の距離を L=3cm で一定とした。phase の選り分けはスペクトル測定の場合とは異なり、rate による選り分けではなく Cool-X 本体の LED の表示によって判定し、heating phase および cooling phase それぞれ 5 周期分ずつ測定した。μ-PIC 全面で測定した 2-D image を図 4.24、4.25 に示す。

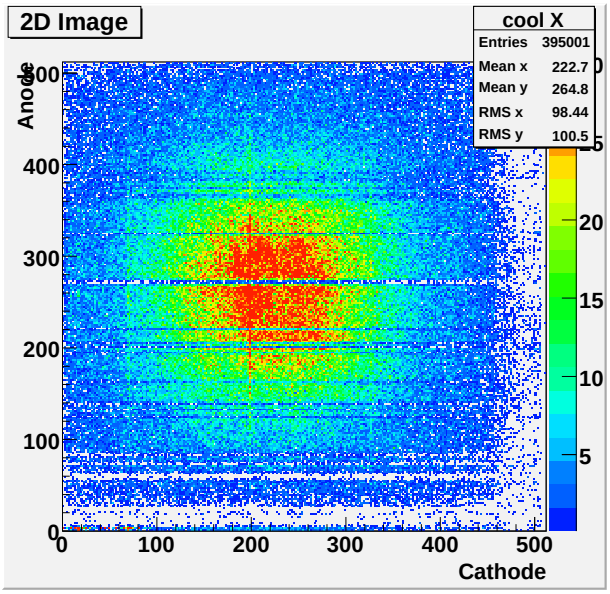


図 4.24: heating phase

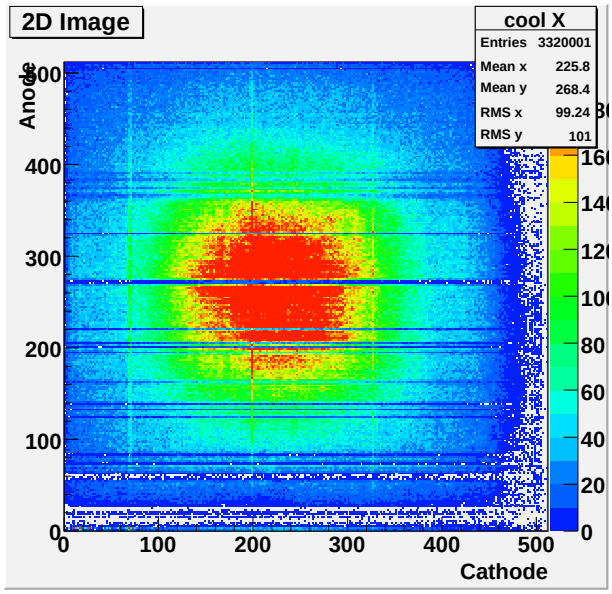


図 4.25: cooling phase

先ほどと同様にして、単位面積あたりの count 数を計算し、beam 中心からの距離に対して片対数で plot したものが図 4.26 である。ここで、縦軸は最高値に対する割合として規格化した。

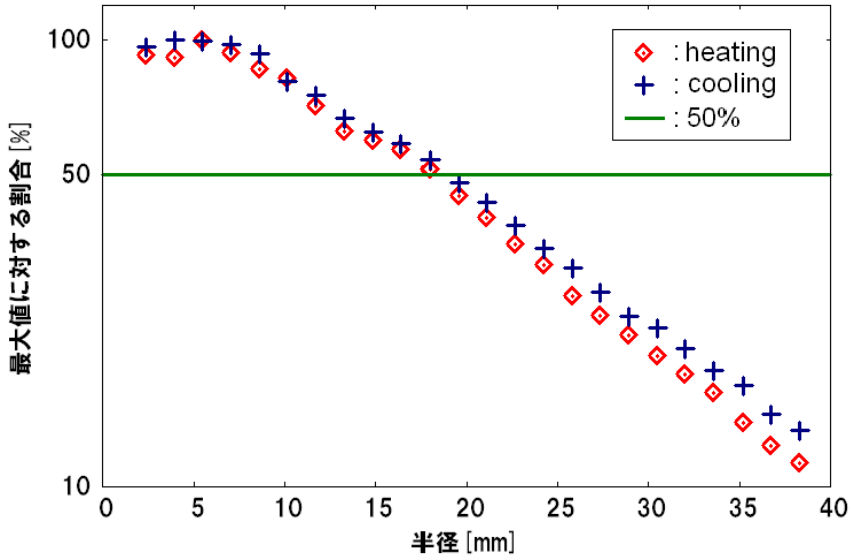


図 4.26: beam の拡がりの phase による違い

図 4.26 によれば、単位面積あたりの count 数が最高値の 50% に落ちる半径 R は、heating phase、cooling phase ともに 20mm 弱で、ほとんど差が無かった。解析の精度が 0.1mm 程度はあると仮定すると、各 phase における L=3cm での R の値は表 4.2 のようになった。

phase	R[mm]
heating	18.1
cooling	18.9

表 4.2: 各 phase に対する R の値

これより両者の R の差は 1mm 弱であり、実際の解析の精度を考えるとただの誤差であるともとれる。しかし、図 4.26 を見ればわかるように、単位面積あたりの count 数が最高値に対して小さくなっていくにつれ、それに相当する両者の半径の差は次第に大きくなっている。十数 % にまで落ちる半径で評価すれば、その差は 3mm になる。Cool-X の構造を考えると、heating phase で特性 X 線を出す Ta は内部の焦電素子表面につけられているので、heating phase のときに放射される X 線は放射筒や放射窓でコリメートされる。一方 cooling phase では、特性 X 線を出す Cu は放射窓の内側に取り付けられており、放射窓の位置で起こった光電吸収によって放射される。このため、放射窓を囲む Cool-X の筐体によってある程度コリメートされるものの、heating phase に比べれば Cu 特性 X 線の photon は比較的広い立体角で放射されることが予想されるが、図 4.26 の結果はこれに矛盾しない。しかしながら、そのような効果が実際に起こっていると考えるには拡がりの差がかなり小さいように思える。phase による拡がりの違いをより正確に測定する方法としては、放射窓と検出器の距離 L を 1cm 程度にして測定するなどが考えられるが、時間的な都合により今回そのような測定を試すことはできなかった。

5　まとめ

Cool-X の性能評価を行うことが今回の実験の目的であったが、実験装置の状況や、時間的な問題により目的を十分に達成することができなかった。

rate の変化については、Cool-X の製造元である AMP TEK 社による公式なデータと同様な、2 つの phase で各々極大となるような時間変化が確認された。ただし、公式データでは heating phase の方が high rate であるのに対し、今回の測定では cooling phase の方が high rate となった。今回 2 台の Cool-X を使用し、掲載したデータは全て 2 台目の物であるが、1 台目においても cooling phase の方が high rate であった。このように Cool-X には個体差があると考えられる。

また、rate の長期的な変化を見てやると、1 周期の中での変化の仕方は使用時間が増すにつれ大きく変容した。中でも平均 rate は明らかに減少していき、Cool-X は長期間に渡る使用には向いていないといえる。

このような Cool-X の性質や時間的な問題のため、スペクトルはほぼ cooling phase についてしか測定できなかったが、正常な動作の Cool-X であれば、rate によってスペクトルを 2 つの phase に選り分けることができると考えられる。ただし、今回用いた logic で 2 つの phase が完全に分離できているわけではなく、この点については今後改善が必要とされる。

beam の拡がりとしては、中心の最高値に対して強度が半分になる半径を用いて大雑把に評価したが、評価方法として本質的且つ精密なものであったとは言い難い。放射源は少なくとも点源でないことがわかり、これは Cool-X の構造と照らして妥当であるが、放射領域がどの程度の拡がりであるかは未知である。これに加え、放射が一様ではないことも拡がりの評価が難しい要因となっている。また、phase による拡がりの違いのより正確な測定も今後の課題である。

以上のように、Cool-X の性能評価としては中途半端な結果に終わってしまったが、基本的な測定はできたので、これを基により正確でより詳細な性能評価をすることができると考えられる。また、そのような測定ができれば、結果を用いてシミュレーションを行い、Cool-X と検出器の相互作用を予測することも可能となる。

A　実験に関する補足

A.1　バックグラウンド測定

2-D imaging など、μ-PIC の全面を用いる測定では、線源を置かずに測定を行っても、十分長い時間測定を行えば、若干数の event があった。この event は放電などのために特定の列に乗るノイズと異なり、μ-PIC 上でほぼ一様な分布をしており、バックグラウンドと考えられる。

実験は建物内で行ったが、窓のすぐ側であったこともあり、このバックグラウンドの主な成分は宇宙からの muon であると考えられる。これを確認するため、水平に置かれた μ-PIC の上面と下面をプラスチックシンチレーターで覆い、上下のシンチレーターおよび μ-PIC で同時刻に観測された event のみを取得し、そのスペクトルを測定した。シンチレーターは鉛直方向から見て μ-PIC の検出面全体を十分覆うサイズのものを使用し、検出面から各々8cm の位置に水平に固定した。μ-PIC 全面で取得したスペクトルを ^{109}Cd を用いてエネルギー較正し、足し合わせたものを以下に示す。

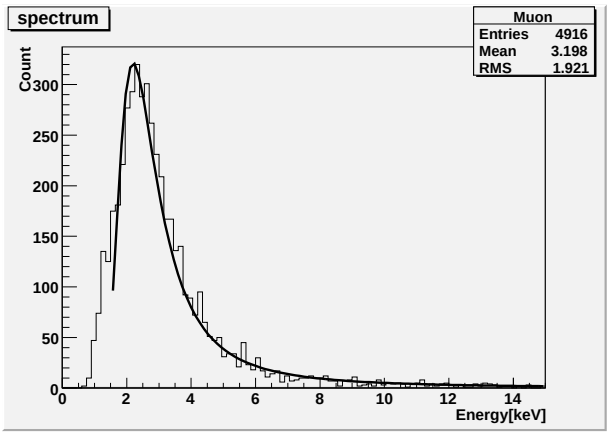


図 A.1: バックグラウンドのスペクトル

muon の到来方向は $\cos^n$ 則に従うので、鉛直方向にやってくるものがもっとも多い。event がこのような成分だと思うと、muon が μ-PIC 中を通過する距離は 1cm であるから、図 A.1 の横軸は大雑把に [keV/cm] 単位で表したエネルギー損失 dE/dx と思ってよい。厚さ 1cm の μ-PIC に対しては muon は MIP(Minimum Ionizing Particle) として扱えるから、dE/dx の分布はいわゆる Landau 分布となる。図 A.1 中の滑らかな実線は、このような予想に基づいてスペクトルを landau 関数で fit した結果である。fitting の結果から、ピークの位置は 2.3keV となった。Ar 中での MIP のエネルギー損失は文献値によれば 2.44keV/cm であるから、この結果は妥当であると考えられる。ただし、やってきた muon は $\cos^n$ 則のために鉛直成分が優位とはいえ、今回のセットアップでは検出器の視野がある程度の拡がりをもっており、鉛直方向以外からやってきた event については検出器中での飛程は 1cm より大きくなるし、また検出器の端の部分を通過した muon に関しては飛程が 1cm より小さくなる。そのためより正確に dE/dx を測定するには μ-PIC を TPC として用いて、各 event に対して検出器中での飛程を測定することが必要となる。

いずれにせよ、このバックグラウンドの到来頻度は検出器全面で数 Hz 程度と、線源や Cool-X の rate に比べ十分無視できるものであった。

A.2　実験中の μ-PIC の変調

実験中には μ-PIC が予期せぬ挙動を示して中断することが度々あったが、そのような事態は往々にして μ-PIC の取り扱いが不慣れであるために起こったことであり、基本的には挙動の様子から推測することで原因を特定できた。

似たような症状が過去に知られていれば、このような事態からすぐに抜け出すことができるが、今回の実験中には、過去に知られていないような変調が現れた。結果として、この変調は μ-PIC や GEM 自体に問題があったわけではなく、電圧が正しく供給されていなかったというとても単純な原因によるものであったが、原因の特定に限られた実験時間をかなり費やしてしまった。今後の実験においてこのようなことがないよう、症状と原因について簡単に述べる。

まず μ-PIC が正常なときには、線源からのパルスの rate は図 A.2 のように当然ながらほとんど一定となる。しかし、まったく同様のセットアップで実験したにもかかわらず図 A.3 のような一見周期的な rate の時間変動が見られるようになり、最終的には図 A.4 に見られるくらいまで変動が大きくなった。この状態では変動はもはや周期的ですらなく、得られるパルス波形も通常とは異なるものとなり、測定が困難となった。図 A.4 の段階での具体的な挙動としては、rate が極大となる phase では通常とほとんど同じ波高、時間幅のパルスが見られるが、rate が低下するに従い波高が小さくなり、時間幅は逆に大きくなる。そして最終的に rate が極小となる phase に至ると、もはや元の滑らかな波形は崩れてしまい、この phase が長く続くためにスペクトルの取得などは基本的に不可能となった。最後に放電をしたような波形が見えた後に極大の phase に戻るというところまでを 1 周期として、図 A.4 のように繰り返し rate が変動した。この過程では、rate のみならずゲイン自体が変動していた。

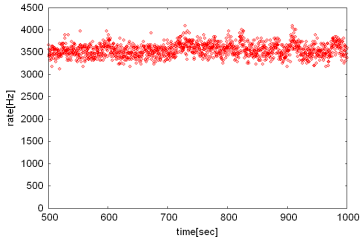


図 A.2: 正常な rate

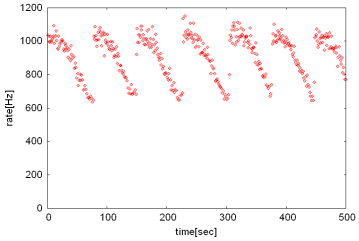


図 A.3: 変動の兆候

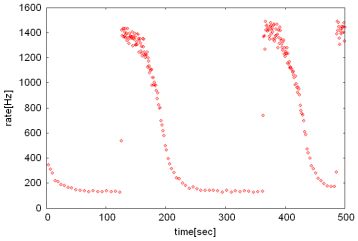


図 A.4: 最大となった変動

検出器や読み出し部など、原因となりうる箇所をそれぞれ調べた結果、Drift Plane 上で電圧供給端子を固定していた接触面において、Drift Plane を構成するアルミの薄膜が剥がれ落ち、接触不良を起こしていたことがわかった。この部分が導通するように直すと、ゲインの変動が収まった。

B プログラム集

B.1 spectrum の取得

B.1.1 RPV160 起動プログラム

```
#include "rpv160.h"

int data_num_fadc(unsigned int addr){
  unsigned short  *ptr;
  int             flag, num;
  while(1){
    //    printf("waiting\n");
    usleep(100);
    ptr = (unsigned short *)(addr + 0x30004);
    flag = ( *ptr & 0x20 ) >> 5;
    //   fprintf(stderr,"flag=%d\n",flag);

    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;
/*
  printf("    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;
/*
  printf("        pagesize         : 0x%x\n", pagesize);
  printf("        mapsize          : 0x%x\n", *mapsize);
  printf("  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);
/*  printf(" memory : unmapped.\n");*/
  return 0;

};
```

B.1.2 RPV130 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;

  fprintf(stderr,"IRQ    = %d\n", pr.irq);
  fprintf(stderr,"VECTOR = 0x%x\n", pr.vector);

  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;
/*
  printf("  RPV-130 base address   : 0x%lx\n", base);
*/
  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;
/*
  printf("        pagesize         : 0x%x\n", pagesize);
  printf("        mapsize          : 0x%x\n", *mapsize);
  printf("  RPV130  addr on linux  : 0x%x\n", (unsigned int) addr);
*/
  return (unsigned int)addr;
};

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

```
/*  printf(" RPV130 : unmapped.\n");*/
  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);
}
```

B.1.3 波形取得プログラム

```
#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 <time.h>
#include <iostream>
#include <cstdio>

#include <vmedrv.h>
#include "rpv160.h"
#include "rpv130.h"


#define BASE 0x80000000
#define BASEi 0x2000

#define PULSE_TH 15
#define PULSE_WIDTH 50
#define FADC_MAX 130624
#define PULSE_MAX 255
#define PULSE_ZONE 500
#define LOW_NUM 1000

void end_vme(int dev,int devi,size_t map,size_t mapi,unsigned int addr,unsigned int addri);

int main(int argc,char *argv[]){

    /*vme 用変数*/
    int         dev, devi;
    size_t      map, mapi;
    char        *addr, *addri;

    /*ファイル作成用変数*/
    FILE *fp1, *fp2;
    char filename1[256];
    char filename2[256];
    char directory[256];
    int PulseNum = 1;
    int iFWHM[8];
    int PulseTime[8];
    int PulseHeight[8];
```

```
  int PulseHalf[8];
  int NoiseHeight[8];
  int Offset[8];
  int GoodChannel = 0;

  int savemode = 0;
  char gomi[80] = "aaaa";

  int flag[8];


  time_t timer;
  struct tm *JTime;

  /*FlashADC のデータ取得 FADC_depth の決定*/
  int FADC_depth=3000;
  if(argc >= 2){
     PulseNum = atoi(argv[1]);
     if(argc >= 3){
        FADC_depth = atoi(argv[2]);
     }
  }

  fprintf(stderr,"FADC depth=%d\n",FADC_depth);
  fprintf(stderr,"PulseNum=%d\n",PulseNum);

  /*時刻取得*/
  time(&timer);
  /*日本時間に直す*/
  JTime = localtime(&timer);

  mode_t mode = S_IRWXU | S_IRGRP | S_IXGRP | S_IROTH | S_IXOTH ;

  /*Save mode の選択*/
  while(1){
  std::cout << "\n<<Choose Save Mode>>\n\n" <<
"press [1] for Normal mode,  or  press [2] for Continue mode.\n";

  std::cin >> savemode;

    if(std::cin.fail() && !std::cin.bad()){
      std::cin.clear();
```

```
      std::cin >> gomi;
    }

    if(savemode == 1){
      std::cout << "Normal mode has started...\n\n\n";//いつもの
      usleep(500000);
      break;
    }

    if(savemode == 2){
      std::cout << "Continue mode has started...\n\n\n";//既に存在する directory へ追加
      usleep(500000);
      break;
    }

    else{
      std::cout << "Choose again!\n";
    }

  }

  /*Normal mode... ディレクトリ名 data_年月日-時分秒"内に保存*/
  if(savemode == 1){

     sprintf(directory, "data_%04d%02d%02d-%02d%02d%02d",JTime->tm_year+1900,JTime->tm_mon+1,
JTime->tm_mday,JTime->tm_hour,JTime->tm_min,JTime->tm_sec);

      if(mkdir(directory,mode) != 0){
         fprintf(stderr, "\n cannot make a directory.\n");
         exit(0);
      }

     sprintf(filename2,"%s//files.dat",directory);
     fp2 = fopen(filename2,"a");
     fprintf(fp2,"CountNum\tnum\tPulseChannel\tPulseTime\tFWHM\tPulseHeight\n");
     fclose(fp2);

  }

  /*Continue mode... ディレクトリ名の読み込み*/

     char outname[256];
```

```
   int oldCountNum = 0;
   int enu = 0;

if(savemode == 2){

   int dr1;
   int dr2;
   FILE *fp3;

   std::cout << "<<Type directory name>>\n\nput former 8keta\n";
   std::cin >> dr1;
   std::cout << "put latter 6keta\n";
   std::cin >> dr2;

   sprintf(directory, "data_%08d-%06d",dr1,dr2);

   for(enu = 0; enu < 100000; enu ++){

      sprintf(outname, "%s//out%05d.dat", directory, enu);
         printf("enu = %5d\n",enu);

         if((fp3 = fopen(outname,"r")) == NULL){
            oldCountNum = enu - 1;
            printf("oldCountNum=%5d\n",oldCountNum);
             break;
         }



   fclose(fp3);

   }

   sprintf(filename2,"%s//files.dat",directory);
   //fp2 = fopen(filename2,"a");
   //fprintf(fp2,"CountNum\tnum\tPulseChannel\tPulseTime\tFWHM\tPulseHeight\n");
   //fclose(fp2);

}//
```

```
int CountNum = 0;
while(1){
    /*ここから１ファイル"./ディレクトリ名/out*****.dat"*/

    for(int i = 0; i < 8; i ++){
        flag[i] = 0;
    }

    /*vme 初期化*/
    addr = (char *)init_fadc( &dev, BASE, &map );
    addri = (char *)init_rpv130( &devi, BASEi, &mapi);
    set_fadc((unsigned int)addr);
    clear_fadc((unsigned int)addr);
    start_fadc((unsigned int)addr);

    usleep(1000);

    /*データ取得やめ*/
    pulse_rpv130((unsigned int) addri, 1);

    /*データ取り込み*/
    int num = data_num_fadc((unsigned int) addr);
    fprintf(stderr,"num = %d\n", num);
    int ch[8][num];
    //fprintf(stderr,"===Channel : %d===\n", Channel);
    if((num < LOW_NUM) || (num == FADC_MAX)){
        end_vme(dev,devi,map,mapi,(unsigned int)addr, (unsigned int)addri);
        continue;
    }

    for( int j = 0; j < 8; j ++ ){
        read_fadc((unsigned int) addr, num, j, &ch[j][0] );
    }

    for(int Channel = 0; Channel < 8; Channel ++){

        /*PulseHeight 探し (単純に最大値)*/
        PulseHeight[Channel] = ch[Channel][0];
        NoiseHeight[Channel] = ch[Channel][0];
        PulseTime[Channel] = 0;
        Offset[Channel] = 0;
```

```
            int OffsetInt = 0;

            for(int j = 0; j < num; j ++){
                if(ch[Channel][j] > PulseHeight[Channel]){
                    PulseHeight[Channel] = ch[Channel][j];
                    PulseTime[Channel] = j;
                }
                if(j < num - PULSE_ZONE){
                    OffsetInt = OffsetInt + ch[Channel][j];
                    if(ch[Channel][j] > NoiseHeight[Channel]){
                        NoiseHeight[Channel] = ch[Channel][j];
                    }
                }
                else{
                    if(ch[Channel][j] == 0){
                        flag[Channel] = 1;
                    }
                }
            }
            Offset[Channel] = OffsetInt / (num - PULSE_ZONE);
            NoiseHeight[Channel] = NoiseHeight[Channel] - Offset[Channel];
            PulseHeight[Channel] = PulseHeight[Channel] - Offset[Channel];
            if(PulseTime[Channel] < num - PULSE_ZONE){
//                fprintf(stderr,"error1\n");
                flag[Channel] = 1;
            }
            if(PulseHeight[Channel] == (PULSE_MAX - Offset[Channel])){
//                fprintf(stderr,"error2\n");
                flag[Channel] = 1;
            }


            /*半値幅 iFWHM を求める*/
            PulseHalf[Channel] = PulseHeight[Channel] / 2;
            iFWHM[Channel] = 0;

            if((PulseTime[Channel] - PULSE_WIDTH < 0) ||
(PulseTime[Channel] + PULSE_WIDTH > num)){
//                fprintf(stderr,"error4\n");
                flag[Channel] = 1;
            }
            for(int j = PulseTime[Channel] - PULSE_WIDTH;j < PulseTime[Channel]
```

```
 + PULSE_WIDTH; j ++){
                if((ch[Channel][j] - Offset[Channel]) > PulseHalf[Channel]){
                    iFWHM[Channel] = iFWHM[Channel] + 1;
                }
            }

            if(iFWHM[Channel] > PULSE_WIDTH){
//                fprintf(stderr,"error5\n");
                flag[Channel] = 1;
            }
            if(PulseHalf[Channel] < NoiseHeight[Channel]){
//                fprintf(stderr,"error6\n");
                flag[Channel] = 1;
            }

            if(PulseHeight[Channel] < PULSE_TH){
//                fprintf(stderr,"error7\n");
                flag[Channel] = 1;
            }
            if(flag[Channel] == 0){
                GoodChannel = Channel;
            }

//            fprintf(stderr,"PulseHeight[%d] = %d\n",Channel,PulseHeight[Channel]);
        }
        if(flag[0]+flag[1]+flag[2]+flag[3]+flag[4]+flag[5]+flag[6]+flag[7] != 7){
            end_vme(dev,devi,map,mapi,(unsigned int)addr, (unsigned int)addri);
            fprintf(stderr,"\n failed[%d%d%d%d%d%d%d%d]\n",flag[0],flag[1],flag[2],
flag[3],flag[4],flag[5],flag[6],flag[7]);
            continue;
        }

        /*Normal mode... データ書き込み*/
        if(savemode == 1){

        fp2 = fopen(filename2,"a");
        fprintf(fp2,"out%05d\t%d\t%d\t\t%d\t\t%d\t%d\n",CountNum, num, GoodChannel,
 PulseTime[GoodChannel], iFWHM[GoodChannel], PulseHeight[GoodChannel]);
        fclose(fp2);


        sprintf(filename1,"%s//out%05d.dat", directory, CountNum);
```

```
fp1 = fopen(filename1,"w");

if(num>FADC_depth){
    for(int i = 0; i < num; i ++ ){
        fprintf(fp1, "%d\t", i);
        for(int j = 0; j < 8; j ++){
            fprintf(fp1, "%d\t", ch[j][i]);
        }
        fprintf(fp1,"\n");
    }
}
else{
    for(int i = 0; i < FADC_depth - num; i ++){
        fprintf(fp1,"%d\t", i);
        for(int j = 0; j < 8; j ++){
            fprintf(fp1, "0\t");
        }
        fprintf(fp1,"\n");
    }
    for(int i = FADC_depth - num; i < FADC_depth; i ++){
        fprintf(fp1,"%d\t", i);
        for(int j = 0; j < 8; j ++){
            fprintf(fp1, "%d\t", ch[j][i - (FADC_depth - num)]);
        }
        fprintf(fp1,"\n");
    }
}

/*おわり*/
end_vme(dev,devi,map,mapi,(unsigned int)addr, (unsigned int)addri);
fclose(fp1);

}//


/*Continue mode... データ書き込み*/
if(savemode == 2){

int allCountNum = CountNum + oldCountNum + 1;

fp2 = fopen(filename2,"a");
fprintf(fp2,"out%05d\t%d\t%d\t\t%d\t\t%d\t%d\n",allCountNum, num, GoodChannel,
```

```
PulseTime[GoodChannel], iFWHM[GoodChannel], PulseHeight[GoodChannel]);
        fclose(fp2);


        sprintf(filename1,"%s//out%05d.dat", directory, allCountNum);
        fp1 = fopen(filename1,"w");

        if(num>FADC_depth){
            for(int i = 0; i < num; i ++ ){
                fprintf(fp1, "%d\t", i);
                for(int j = 0; j < 8; j ++){
                    fprintf(fp1, "%d\t", ch[j][i]);
                }
                fprintf(fp1,"\n");
            }
        }
        else{
            for(int i = 0; i < FADC_depth - num; i ++){
                fprintf(fp1,"%d\t", i);
                for(int j = 0; j < 8; j ++){
                    fprintf(fp1, "0\t");
                }
                fprintf(fp1,"\n");
            }
            for(int i = FADC_depth - num; i < FADC_depth; i ++){
                fprintf(fp1,"%d\t", i);
                for(int j = 0; j < 8; j ++){
                    fprintf(fp1, "%d\t", ch[j][i - (FADC_depth - num)]);
                }
                fprintf(fp1,"\n");
            }
        }

        /*おわり*/
        end_vme(dev,devi,map,mapi,(unsigned int)addr, (unsigned int)addri);
        fclose(fp1);

        }//



        /*ここまで1ファイル*/
```

```
        fprintf(stderr, "\t *** data aquired. [ %d / %d ] *** \n \n", CountNum + 1,
PulseNum);
        CountNum = CountNum + 1;
        if(CountNum == PulseNum){
            break;
        }
    }
    return 0;
}

void end_vme(int dev, int devi, size_t map, size_t mapi, unsigned int addr, unsigned int
addri){
    end_fadc(dev,map,addr);
    end_rpv130(devi,mapi,addri);

    usleep(1000);
}
```

B.1.4 積分用プログラム

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

#define CHANNEL 0
#define PULSE_WIDTH 50
#define PULSE_ZONE 500

int main(){
    FILE *fp;
    char filename[256];
    int ch[8][100000];
    int Count[100000];
    int Channel = 0;
    int PulseHeight[8];
    int PulseTime[8];
    int num;
    int Rnum;

    int CountNum = 0;

    while(CountNum < 100000){
```

```
        /*out**.dat のファイルから中身を全て ch[][] に*/
        sprintf(filename, "out%05d.dat", CountNum);

        if((fp = fopen(filename, "r")) == NULL){
            printf("ファイル'out%05d.dat' まで積分完了。\n", CountNum - 1);
            break;
        }

        int i = 0;
        while(fscanf(fp, "%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\n", &Count[i],
&ch[0][i], &ch[1][i], &ch[2][i], &ch[3][i], &ch[4][i], &ch[5][i], &ch[6][i],
&ch[7][i]) != EOF){
            i = i + 1;
        }
        num = i;
        fclose(fp);


        /*実際の num の値 (Rnum) 探し*/
        Rnum = 0;
        for(Channel = 0; Channel < 8; Channel ++){
            for(i = 0; i < num; i ++){
                if(ch[Channel][i] != 0){
                    break;
                }
            }
            if(Rnum < num - i){
                Rnum = num - i;
            }
        }


        /*PulseHeight(maximum) 及び MaxCH 探し*/
        int MaxCH = 0;

        for(Channel = 0; Channel< 8; Channel ++){
            PulseHeight[Channel] = ch[Channel][0];
            PulseTime[Channel] = 0;

            for(int j = num - Rnum; j < num; j ++){
                if(ch[Channel][j] > PulseHeight[Channel]){
```

```
                PulseHeight[Channel] = ch[Channel][j];
                PulseTime[Channel] = j;
            }
        }
        if(PulseHeight[Channel] > PulseHeight[MaxCH]){
    MaxCH = Channel;
        }
    }

    FILE *maxchfp;
    maxchfp = fopen("MaxCH.dat", "a");
    fprintf(maxchfp, "%d\n", MaxCH+1);
    fclose(maxchfp);

    FILE *phfp;
    phfp = fopen("PulseHeight.dat", "a");
    fprintf(phfp, "%d\n", PulseHeight[MaxCH]);
    fclose(phfp);


    /*BaseLine の決定*/
int BaseInt = 0;
    for(int j = num - Rnum; j < PulseTime[MaxCH] - 10; j ++){
        BaseInt = BaseInt + ch[MaxCH][j];
    }
    int BaseLine = BaseInt / (PulseTime[MaxCH] - 10 - num + Rnum);


    PulseHeight[MaxCH] = PulseHeight[MaxCH] - BaseLine;

    fprintf(stderr, "out%.05d, MaxCH:%d, BaseLine = %d, num:%d\n", CountNum,
 MaxCH, BaseLine, Rnum);


    /*全体の平均を取って全体を積分*/
    int mIntegral = 0;
    for(int j = num - Rnum; j < num; j++){
        mIntegral = mIntegral + ch[MaxCH][j];
    }

    int Mean = mIntegral / Rnum;
    int mIntegral2 = 0;
```

```
        for(int j = num - Rnum; j < num; j ++){
            if(ch[MaxCH][j] > Mean){
                mIntegral2 = mIntegral2 + ch[MaxCH][j] - Mean;
            }
        }
        FILE *meanfp;
        meanfp = fopen("meanInt.dat", "a");
        fprintf(meanfp,"%d\n", mIntegral2);
        fclose(meanfp);


        /*FWHM を求めて, PulseTime を中心に FWHM*2 の区間を積分*/
        int PulseHalf = PulseHeight[MaxCH] / 2;
        int iFWHM = 0;
        int fIntegral = 0;

        for(int j = PulseTime[MaxCH] - PULSE_WIDTH; j < PulseTime[MaxCH] +
 PULSE_WIDTH; j ++){
            if((ch[MaxCH][j] - BaseLine) > PulseHalf){
                iFWHM = iFWHM + 1;
            }
        }
        for(int j = PulseTime[MaxCH] - iFWHM; j < PulseTime[MaxCH] + iFWHM;j ++){
            fIntegral = fIntegral + ch[MaxCH][j] - BaseLine;
        }
        FILE *fwhmfp;
        fwhmfp = fopen("fwhmInt.dat","a");
        fprintf(fwhmfp,"%d\n", fIntegral);
        fclose(fwhmfp);



        /*BaseLine を引いた上で PulseTime を中心に PULSE_WIDTH*2 を積分*/
int bpwIntegral = 0;
        for(int j = PulseTime[MaxCH] - PULSE_WIDTH; j < PulseTime[MaxCH] +
 PULSE_WIDTH;j ++){
            bpwIntegral = bpwIntegral + ch[MaxCH][j] - BaseLine;
        }
        FILE *bpwfp;
        bpwfp = fopen("bpwInt.dat", "a");
        fprintf(bpwfp,"%d\n", bpwIntegral);
```

```
        fclose(bpwfp);


        /*BaseLine を引いた上で PulseTime から前 100nsec, 後 1000nsec を積分*/
        int baseIntegral = 0;
        for(int j = PulseTime[MaxCH] - 10; j < PulseTime[MaxCH] + 100;j ++){
            baseIntegral = baseIntegral + ch[MaxCH][j] - BaseLine;
        }
        FILE *basefp;
        basefp = fopen("baseInt.dat", "a");
        fprintf(basefp,"%d\n", baseIntegral);
        fclose(basefp);




        CountNum ++;
    }

    return 0;
}
```

B.1.5 Cool-X の spectrum 取得用シェルスクリプト

```
#!/bin/bash
#引数 1 つ目がフォルダ名, 2 つ目がデータ取得数
#(データ取得数/Width)< 50 万でかつ割り切れる値に設定


Count=1
HighCount=0
LowCount=0
Width=1000
SleepTime=0
Flag0="other"
num=‘expr $2 / $Width‘
mkdir $1
cd "./$1/"

echo "waiting time : $SleepTime sec"
echo "waiting time : $SleepTime sec" >> TimeElaps.dat
echo " "
```

```
mkdir highrate
mkdir lowrate
echo "filename\tnum\tPulseTime\tFWHM\tPulseHeight" > ./highrate/file.dat
echo "filename\tnum\tPulseTime\tFWHM\tPulseHeight" > ./lowrate/file.dat

while [ $Count -le $num ];
do
    echo "notyet" > flag2.dat
    /home/p6/vme/coolXspctrm/2d/DAQclock.out $Width > ichiji.dat
    cat ichiji.dat >> 2Dclock.dat
    /home/p6/vme/coolXspctrm/read2D.out > flag1.dat 2>> TimeElaps.dat

    #"flag1.dat"の中身はレートに合わせた"high""low"か, どちらとも取れない"other"
    Flag1=‘cat flag1.dat‘
    echo " flag0 : $Flag0"
    echo " flag1 : $Flag1"
    if [ "$Flag1" = "high" ]; then
        if [ "$Flag0" = "high" ]; then
            /home/p6/vme/coolXspctrm/spctrm/DAQspctrm.out "./highrate
/out$HighCount.dat" >> ./highrate/file.dat 2> flag2.dat
            cat ichiji.dat >> 2Dhigh.dat
    if [ "‘cat flag2.dat‘" = "DAQ" ]; then
                echo " *** data stored in ./highrate/out$HighCount.dat ***"
                HighCount=‘expr $HighCount + 1‘
            fi
        fi
        Flag0="high"

    elif [ "$Flag1" = "other" ]; then
        if [ "$Flag0" = "high" ]; then
            Flag0="middle"
        elif [ "$Flag0" = "low" ]; then
    Flag0="other"
        fi

    elif [ "$Flag1" = "low" ]; then
        if [ "$Flag0" = "high" ]; then
    Flag0="high"
        elif [ "$Flag0" = "middle" ]; then
    Flag0="low"
            /home/p6/vme/coolXspctrm/spctrm/DAQspctrm.out "./lowrate/out$LowCount.dat"
```

```
>> ./lowrate/file.dat 2> flag2.dat
            cat ichiji.dat >> 2Dlow.dat
    if [ "`cat flag2.dat`" = "DAQ" ]; then
                echo " *** data stored in ./lowrate/out$LowCount.dat ***"
                LowCount=`expr $LowCount + 1`
            fi
        elif [ "$Flag0" = "low" ]; then
    Flag0="low"
            /home/p6/vme/coolXspctrm/spctrm/DAQspctrm.out "./lowrate/out$LowCount.dat"
>> ./lowrate/file.dat 2> flag2.dat
            cat ichiji.dat >> 2Dlow.dat
    if [ "`cat flag2.dat`" = "DAQ" ]; then
                echo " *** data stored in ./lowrate/out$LowCount.dat ***"
                LowCount=`expr $LowCount + 1`
            fi
        elif [ "$Flag" = "other" ]; then
            Flag0="high"

        fi
    fi

    DaqCount=`expr $Count \* $Width`
    echo " data acqured [ $DaqCount / $2 ]"
    echo " "
    sleep $SleepTime
    Count=`expr $Count + 1`
done
echo "finished DAQ"
echo "$HighCount pulses in ./highrate"
echo "$LowCount pulses in ./lowrate"
echo " "
#/home/p6/vme/coolXspctrm/alltime3.out TimeElaps.dat >> "plotrate.dat"
rm ichiji.dat
rm flag1.dat
rm flag2.dat

#2D を root でみる
cut -f 1 2Dlow.dat > 2D_1.dat
cut -f 2 2Dlow.dat > 2D_2.dat
paste 2D_1.dat 2D_2.dat >> 2Dlow_root.dat
cut -f 1 2Dhigh.dat > 2D_1.dat
cut -f 2 2Dhigh.dat > 2D_2.dat
```

```
paste 2D_1.dat 2D_2.dat >> 2Dhigh_root.dat
rm 2D_1.dat
rm 2D_2.dat
```

B.2 2-D imaging

B.2.1 memory board からのデータ取得

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <fstream.h>
#include <iomanip.h>
#include <string.h>
#include <ctype.h>
#include <errno.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"
#include "main.h"

using namespace std;

int take_data(){

  int flag = 0;

  // HE2687
  //reset_memory( (unsigned int)addr_he2687 );
  read_memory2( (unsigned int)addr_he2687, num_event );

  return flag;

};


void write_header( int depth ){
  time_t    now_t;
  struct tm *t;
  char      message[HEADERSIZE];
```

```
  now_t = time( NULL );
  t     = localtime( &now_t );

  sprintf( message, "%d/%d/%d/%d:%d:%d %d",
           t->tm_year + 1900,
           t->tm_mon  + 1,
           t->tm_mday,
           t->tm_hour,
           t->tm_min,
           t->tm_sec,
           depth
         );

//  cerr << "Writing the header in the out file.\n"
//       << "Header: " << message << endl;

  outf.write( message, HEADERSIZE );

}

long Now( void ){
  time_t timer, now;

  now = time( &timer );
  return (long)now;
}


int main( int argc, char **argv ){

  long   start_time, end_time, elapsed_time;
  double rate;

  // set default values
  num_event     = 10000;
  // Get options
  /**  int option;
  while( ( option = getopt( argc, argv, "n:" ) ) != -1 ){
    switch( option ){
    case 'n':
      // Number of events acquired
```

```
      num_event = atoi( optarg );
      break;
    default:
      cerr << "Incorrect option was given!" << endl;
      exit(1);
    }
    }**/
  if(argc>1)num_event=atoi(argv[1]);
//  cerr << "num of event" << num_event<<endl;
  // HE2687
  addr_he2687 = (char *)init_memory( &dev_he2687, BASE_HE2687, &map_he2687 );
  reset_memory( (unsigned)addr_he2687 );
  set_memory( (unsigned int)addr_he2687, num_event );
  abort_memory( (unsigned int)addr_he2687 );

  // DAQ start

//  start_time = Now();
//  cerr << "DAQ start...";
//  cerr << "data stored:" <<num_memory( (unsigned int)addr_he2687 )<< endl;
  start_memory( (unsigned int)addr_he2687 );
  take_data();

//  cerr << "data stored:" <<num_memory( (unsigned int)addr_he2687 )<< endl;
//  end_time = Now();
//  elapsed_time = end_time - start_time;
//  rate = num_event / elapsed_time;

//  cerr << "...done." << endl;
//  cerr << "time: " << elapsed_time << " sec (for num = " << num_event << ")" << endl;
//  cerr << "rate: " << rate         << " Hz"  << endl;

  return 0;
}
```

B.3 rate の時間変化の取得

B.3.1 encorder board のクロックおよび 2-D 位置情報を取得するプログラム

```
#include<stdio.h>
#define TIME_RESOLUTION 0.00262144

int main(){
int ch[3][500000];
double clock_delta;
FILE *fp;

fp = fopen("ichiji.dat", "r");
int i = 0;
clock_delta = 0;
while(fscanf(fp, "%d\t%d\t%d\n", &ch[0][i], &ch[1][i], &ch[2][i]) != EOF){
if(i > 0){
if(ch[2][i] - ch[2][i - 1] < 0){
ch[2][i] = ch[2][i] + 1024;
}

clock_delta = clock_delta + ch[2][i] - ch[2][i - 1];
if(ch[2][i] > 1024){
ch[2][i] = ch[2][i] - 1024;
}
}
fprintf(stdout,"%d\t%d\n", ch[0][i], ch[1][i]);
i = i + 1;
}

fprintf(stderr,"time: %.10f sec (for num = %d)\n", clock_delta * TIME_RESOLUTION, i);
fclose(fp);
return 0;
}
```

B.3.2 high rate と low rate を振り分けるプログラム

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

int main(int argc, char *argv[]){
```

```
int ch[3][500000];
int clock_delta;
int Width;
FILE *fp;

Width = atoi(argv[1]);
fp = fopen("ichiji.dat", "r");
int i = 0;
clock_delta = 0;
while(fscanf(fp, "%d\t%d\t%d\n", &ch[0][i], &ch[1][i], &ch[2][i]) != EOF){
if(i > 0){
if(ch[2][i] - ch[2][i - 1] < 0){
ch[2][i] = ch[2][i] + 1024;
}
clock_delta = clock_delta + ch[2][i] - ch[2][i - 1];
if(ch[2][i] > 1024){
ch[2][i] = ch[2][i] - 1024;
}
}
i = i + 1;
}
/*目安として 10000 発で 2.6sec(1000clock) 以下なら高 rate,13sec(5000clock) 以上なら低 rate*/
for(int j = 0; j < i; j ++){
if(clock_delta > Width / 2){
fprintf(stderr,"%d\t%d\n", ch[0][j], ch[1][j]);
}
else if(clock_delta < Width / 10){
fprintf(stdout,"%d\t%d\n", ch[0][j], ch[1][j]);
}
}

fclose(fp);
return 0;
}
```

B.3.3 encorder board のクロックから rate を計算するプログラム

```
#include<stdio.h>

int main( int argc, char *argv[]){
double ETime[500000];
```

```
int STime;
int num;
double RealSTime = 0;
double ATime = 0;
double RTime = 0;
double EachRate = 0;
double Rate;
FILE *fp;

fp = fopen(argv[1], "r");

fscanf(fp, "waiting time : %d sec\n", &STime);

int i = 0;
while(fscanf(fp, "time: %lf sec (for num = %d)\n", &ETime[i], &num) != EOF){
RealSTime = STime + 0.00262144 * (num * 0.00613658 + 13.1384);
ATime = ATime + ETime[i] + RealSTime;
RTime = RTime + ETime[i];
EachRate = num / ETime[i];
fprintf(stdout, "%f\t%f\n",  ATime, EachRate);
i = i + 1;
}
Rate = num * i / RTime;

fprintf(stderr, "\tall time (contains waiting time)\t : %.10f sec\n", ATime);
fprintf(stderr, "\tDAQ time (without waiting time)\t\t : %.10f sec\n", RTime);
fprintf(stderr, "\trate (using DAQ time)\t\t\t : %.10f Hz\n", Rate);

fclose(fp);
return 0;
}
```

B.3.4 データ取得を繰り返すシェルスクリプト

```
#!/bin/bash
#引数 1 つ目がファイル名，2 つ目がデータ取得数
#(データ取得数/Width) < 500000 でかつ割り切れる値に設定

Count=1
Width=1000
```

```
SleepTime=0
num=‘expr $2 / $Width‘
Time="Time_$1"

echo "waiting time : $SleepTime sec"
echo "waiting time : $SleepTime sec" >> $Time
echo " "

while [ $Count -le $num ];
do
/home/p6/vme/2D/main2 $Width > ichiji.dat
cat ichiji.dat >> $1
/home/p6/vme/2D/read3.out >> "2line_$1" 2>> $Time
/home/p6/vme/2D/cread.out $Width >> "highrate_$1" 2>> "lowrate_$1"

DaqCount=‘expr $Count \* $Width‘
echo " *** data acqured [ $DaqCount / $2 ] ***"
sleep $SleepTime

Count=‘expr $Count + 1‘
done
echo "finished DAQ"
echo " "
/home/p6/vme/2D/alltime3.out $Time >> "plrate_$1"
rm ichiji.dat
```

B.4 解析用プログラム

B.4.1 2-D image のデータを x,y および count 数で cut するプログラム

```
#include<cstdio>
#include<iostream>

int main( int argc, char *argv[]){

FILE *fp, *fp2;
int nbinx;
int nbiny;
int zcut;
int xmin;
int xmax;
int ymin;
int ymax;
int x[1000000], y[1000000];
char name[80];
char name2[80];
char filename[80];
char filename2[80];
int n = 0;
int num = 0;


std::cout << "loadfile? [2line_***.dat]" << std::endl;
std::cin >> name;
sprintf(filename, "2line_%s.dat", name);

std::cout << "savefile? [cutcrop_***.dat]" << std::endl;
std::cin >> name2;
sprintf(filename2, "cutcrop_%s.dat", name2);


fp = fopen(filename, "r");
fp2 = fopen(filename2, "w");

std::cout << "loading 2line_" << name << ".dat" << std::endl;


while(fscanf(fp, "%d\t%d\n", &x[n], &y[n]) != EOF){
n = n + 1;
```

```
}

num = n;

std::cout << "num = " << num << std::endl;

std::cout << "(int) nbinx = ";
std::cin >> nbinx;
std::cout << std::endl;
std::cout << "(int) nbiny = ";
std::cin >> nbiny;
std::cout << std::endl;

int habax = 512 / nbinx;
int habay = 512 / nbiny;


int i = 0;
int j = 0;
int count[256][256];
int minx = 0;
int miny = 0;
int maxx = 0;
int maxy = 0;

for(int a = 0; a < 256; a++){
for(int b = 0; b < 256; b ++){
count[a][b] = 0;
}
}


for(j = 0; j < nbiny; j ++){
for(i = 0; i < nbinx; i ++){

minx = i * habax;
miny = j * habay;
maxx = (i + 1) * habax;
maxy = (j + 1) * habay;

for(int k = 0; k < num; k ++){
```

```
if(minx <= x[k] && x[k] < maxx && miny <= y[k] && y[k] < maxy){
count[i][j] = count[i][j] + 1;
}
}

}
}

std::cout << "(int) zcut = ";
std::cin >> zcut;
std::cout << std::endl;

std::cout << "(int) xmin = ";
std::cin >> xmin;
std::cout << std::endl;

std::cout << "(int) xmax = ";
std::cin >> xmax;
std::cout << std::endl;

std::cout << "(int) ymin = ";
std::cin >> ymin;
std::cout << std::endl;

std::cout << "(int) ymax = ";
std::cin >> ymax;
std::cout << std::endl;


std::cout << "saving cutcrop_" << name2 << ".dat" << std::endl;


for(j = 0; j < nbiny; j ++){
for(i = 0; i < nbinx; i ++){

minx = i * habax;
miny = j * habay;
maxx = (i + 1) * habax;
maxy = (j + 1) * habay;

for(int k = 0; k < num; k ++){
```

```
if(count[i][j] > zcut){

if(minx <= x[k] && x[k] < maxx && miny <= y[k] && y[k] < maxy
&& xmin <= x[k] && x[k] < xmax && ymin <= y[k] && y[k] < ymax){
fprintf(fp2, "%d\t%d\n", x[k], y[k]);
}
}
}
}
}


fclose(fp);
fclose(fp2);

return 0;
}
```

B.4.2 2-D image を projection するプログラム

```
#include<cstdio>
#include<iostream>

int main( int argc, char *argv[]){

FILE *fp, *fp2;
int x[1000000], y[1000000];
char name[80];
char name2[80];
char filename[80];
char filename2[80];
int n = 0;
int num = 0;
int flag;


std::cout << "loadfile? [cutcrop_***.dat]" << std::endl;
std::cin >> name;
sprintf(filename, "cutcrop_%s.dat", name);

std::cout << "savefile? [projection_***.dat]" << std::endl;
```

```
std::cin >> name2;
sprintf(filename2, "projection_%s.dat", name2);


fp = fopen(filename, "r");
fp2 = fopen(filename2, "w");

std::cout << "loading cutcrop_" << name << ".dat" << std::endl;


while(fscanf(fp, "%d\t%d\n", &x[n], &y[n]) != EOF){
n = n + 1;
}

num = n;

std::cout << "num = " << num << std::endl;

std::cout << "choose projection directon." << std::endl;
std::cout << "enter 0 for projection x (for x v.s. count histgram)." << std::endl;
std::cout << "enter 1 for projection y (for y v.s. count histgram)." << std::endl;
std::cin >> flag;

switch(flag){

case 0:
std::cout << "start projection x..." << std::endl;

for(int i = 0; i < num; i ++){
fprintf(fp2, "%d\n", x[i]);
}

break;

case 1:
std::cout << "start projection y..." << std::endl;

for(int i = 0; i < num; i ++){
fprintf(fp2, "%d\n", y[i]);
}

break;
```

```
default:
std::cout << "enter 0 or 1. end projection." << std::endl;
break;

}

std::cout << "saving projection_" << name2 << ".dat" << std::endl;



fclose(fp);
fclose(fp2);

return 0;
}
```

B.4.3 rate の時間変化を folding するプログラム

```
/*引数 1 つめファイル名,2 つめ num, 3 つめ cooling phase 直前の bottom rate の閾値*/
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#define BIN_TIME 0.5
#define CYCLE 105

int main( int argc, char *argv[]){
int num;
int WaveCount = 0;
int TimeCount = 0;
int LastCount[10000];
int BinCount[1000][1000];
int WaveBinCount[10000];
double RealSTime;
double ATime[10000];
double EachRate[10000];
double ETime[10000];
double ErrTime[10000];
double ErrRate[10000];
double RateSum[1000][1000];
double WaveRateSum[10000];
```

```
double BotRate;
char errfile[256];
char binfile[256];
FILE *fp, *errfp, *binfp;

/*plrate_*.dat のデータ読み込み*/
fp = fopen(argv[1], "r");
sprintf(errfile, "err%s", argv[1]);
sprintf(binfile, "bin%s", argv[1]);
errfp = fopen(errfile, "w");
binfp = fopen(binfile, "w");
num = atoi(argv[2]);
BotRate = atof(argv[3]);

fscanf(fp, "%lf\t%lf\n", &ATime[0], &EachRate[0]);
int i = 1;
while(fscanf(fp, "%lf\t%lf\n", &ATime[i], &EachRate[i]) != EOF){
RealSTime = 0.00262144 * (num * 0.00613658 + 13.1384);
ETime[i] = num / EachRate[i];
ErrTime[i] = ETime[i];
ErrRate[i] = EachRate[i] / sqrt(num);
fprintf(errfp, "%f\t%f\t%f\t%f\t%f\t%f\t%f\n",  ATime[i], EachRate[i], ATime[i]-ErrTime[i],
  ATime[i], EachRate[i]-ErrRate[i], EachRate[i]+ErrRate[i]);

if(EachRate[i] < BotRate){
LastCount[WaveCount] = i;
WaveCount = WaveCount + 1;
}

i = i + 1;
}
TimeCount = i - 1;
fprintf(stdout,"%d\t%f\n", WaveCount, ATime[TimeCount]);
if((WaveCount > ((ATime[TimeCount] - ATime[0]) / 100) + 3)||
(WaveCount < ((ATime[TimeCount] - ATime[0]) / 100) - 3)){
fprintf(stdout, "wrong threshold.\n");
}

for(int j = 0; j < CYCLE / BIN_TIME; j ++){
WaveRateSum[j] = 0;
WaveBinCount[j] = 0;
for(i = 0; i < WaveCount; i ++){
```

```
BinCount[i][j] = 0;
RateSum[i][j] = 0;
for(int k = LastCount[i]; k < LastCount[i+1]; k ++){
/* if((ATime[k] >= (ATime[LastCount[i]] + (ATime[LastCount[i+1]]
- ATime[LastCount[i]]) * j / BIN_TIME + 2)) && (ATime[k] < (ATime[LastCount[i]]
+ (ATime[LastCount[i+1]] - ATime[LastCount[i]]) * (j+1) / BIN_TIME + 2))){
RateSum[i][j] = RateSum[i][j] + EachRate[k];
BinCount[i][j] = BinCount[i][j] + 1;
}*/
if((ATime[k] >= (ATime[LastCount[i] + 2] + BIN_TIME*j))
&&(ATime[k] < (ATime[LastCount[i] + 2] + BIN_TIME * (j+1)))){
RateSum[i][j] = RateSum[i][j] + EachRate[k];
BinCount[i][j] = BinCount[i][j] + 1;

}
}
WaveBinCount[j] = WaveBinCount[j] + BinCount[i][j];
WaveRateSum[j] = WaveRateSum[j] + RateSum[i][j];
}
if(WaveBinCount[j] != 0){
fprintf(binfp,"%f\t%f\n", ATime[LastCount[0] + 2] + BIN_TIME * j,
WaveRateSum[j] / WaveBinCount[j]);
}
}
fclose(fp);
fclose(errfp);
fclose(binfp);
return 0;
}
```

B.4.4 2-D image の beam の拡がりを計算するプログラム

```
#include<stdio.h>
#include<stdlib.h>
#include<iostream>
#include<cstdio>
#include<cmath>

int main(int argc, char *argv[]){
int x[1000000], y[1000000];
double centerx, centery;
```

```
double radmin;
double dr;
char name[80];
char filename[80];
char filename2[80];
char filename3[80];
int level;
FILE *fp;
FILE *fp2;
FILE *fp3;

std::cout << "filename? 2line_***.dat" << std::endl;
std::cin >> name;
std::cout << name << std::endl;

std::cout << "centerx?" << std::endl;
std::cin >> centerx;

std::cout << "centery?" << std::endl;
std::cin >> centery;


sprintf(filename, "2line_%s.dat", name);
sprintf(filename2, "flux2_%s", filename);
sprintf(filename3, "share2_%s", filename);

fp = fopen(filename, "r");
fp2 = fopen(filename2, "w");
fp3 = fopen(filename3, "w");

int i = 0;
int imax = 0;
int entry[256];
double mid[256];
double flux[256];
double area[256];
double deadarea[256];
double r[256];
double maxflux = 0;
double allflux = 0;
double share[256];
```

```
const double pi = 3.141592653589793;

radmin = 0;

std::cout << "radius = ?" << std::endl;
std::cin >> radmin;
std::cout << "radius = "<< radmin << std::endl;

std::cout << "level = ?" << std::endl;
std::cin >> level;
std::cout << "level = " << level << std::endl;


dr = radmin / ( double ) level;

std::cout << "dr = " << dr << std::endl;

/*dead area 準備*/
std::cout << "dead area 補整を行ないます" << std::endl;
/*dead area 用変数*/
int deadnum = 0;
double y1[10];
double y2[10];
double y01[10];
double y02[10];
double S[256][10];
double dS[256][10];
double theta1[256][10];
double theta2[256][10];
/*変数の初期化*/
int m = 0;
int n = 0;
for(m = 0; m < 10; m ++){
y1[m] = 0;
y2[m] = 0;
y01[m] = 0;
y02[m] = 0;
for(n = 0; n < 256; n ++){
S[n][m] = 0;
dS[n][m] = 0;
theta1[n][m] = 0;
theta2[n][m] = 0;
```

```
}
}
/*deadnum の指定*/
std::cout << "dead num を指定して下さい (0 ~ 10)" << std::endl;
std::cin >> deadnum;
/*y1,y2 の決定*/
for(int m = 0; m < deadnum; m ++){
int m1 = m + 1;
std::cout << m1 << "番目の deadarea の下限を入力して下さい" << std::endl;
std::cin >> y01[m];
std::cout << m1 << "番目の deadarea の上限を入力して下さい" << std::endl;
std::cin >> y02[m];

y1[m] = y01[m] - centery;
y2[m] = y02[m] - centery;
}
/*deadarea 準備終*/

while(fscanf(fp, "%d\t%d\n", &x[i], &y[i]) != EOF){
i = i + 1;
}

imax = i;


for(int a = 0; a < 256; a ++){

entry[a] = 0;
mid[a] = 0;
flux[a] = 0;
area[a] = 0;
deadarea[a] = 0;
r[a] = 0;
share[a] = 0;


}



/*r[n] の定義*/
for(n = 0; n < level+1; n ++){
```

```
r[n] = n * dr;
mid[n] = r[n] + dr / 2;

}

/*theta1,theta2,S[n][m] の定義*/

for(n = 0; n < level+1; n ++){

for(m = 0; m < deadnum; m ++){

if(((y1[m] >= r[n]) || (-y1[m] >= r[n])) &&
 ((y2[m] >= r[n]) || (-y2[m] >= r[n]))){
if(y1[m] * y2[m] > 0){
theta1[n][m] = pi/2;
theta2[n][m] = pi/2;
S[n][m] = 0;
std::cout << "type 5 : S[" << n << "][" << m << "] = " << S[n][m] << std::endl;
}

else if(y1[m] * y2[m] < 0){
theta1[n][m] = -pi/2;
theta2[n][m] = pi/2;
S[n][m] = pi*r[n]*r[n];
std::cout << "type 1 : S[" << n << "][" << m << "] = " << S[n][m] << std::endl;
}


}

else if(((y1[m] >= r[n]) || (-y1[m] >= r[n])) &&
((y2[m] < r[n]) && (-y2[m] < r[n]))){
theta1[n][m] = -pi/2;
theta2[n][m] = asin(y2[m]/r[n]);
S[n][m] = r[n] * r[n] * (theta2[n][m]
+ cos(theta2[n][m]) * sin(theta2[n][m]) - theta1[n][m]);
std::cout << "type 2 : S[" << n << "][" << m << "] = " << S[n][m] << std::endl;
std::cout << "cos(theta1) = " << cos(theta1[n][m]) << " , sin(theta1) = "
<< sin(theta1[n][m]) << " , cos(theta2) = " << cos(theta2[n][m]) << " , sin(theta2) = "
<< sin(theta2[n][m]) << std::endl;
```

```
}

else if(((y2[m] >= r[n]) || (-y2[m] >= r[n])) &&
((y1[m] < r[n]) && (-y1[m] < r[n]))){
theta1[n][m] = asin(y1[m]/r[n]);
theta2[n][m] = pi/2;
S[n][m] = r[n] * r[n] * (theta2[n][m] - theta1[n][m]
- cos(theta1[n][m]) * sin(theta1[n][m]));
std::cout << "type 3 : S[" << n << "][" << m << "] = " << S[n][m] << std::endl;


}

else if(((y2[m] < r[n]) && (-y2[m] < r[n])) &&
((y1[m] < r[n]) && (-y1[m] < r[n]))){
theta1[n][m] = asin(y1[m]/r[n]);
theta2[n][m] = asin(y2[m]/r[n]);
S[n][m] = r[n] * r[n] * (theta2[n][m] + cos(theta2[n][m]) * sin(theta2[n][m])
 - theta1[n][m] - cos(theta1[n][m]) * sin(theta1[n][m]));
std::cout << "type 4 : S[" << n << "][" << m << "] = " << S[n][m] << std::endl;



}

}

}


/*dS[n][m] および deadarea[n] の定義*/
for(n = 0; n < level; n ++){

for(m = 0; m < deadnum; m ++){

dS[n][m] = S[n + 1][m] - S[n][m];
std::cout << "dS[" << n << "][" << m << "] = " << dS[n][m] << std::endl;
deadarea[n] = deadarea[n] + dS[n][m];

}
```

```
}

for(n = 0; n < level; n ++){

area[n] = pi*(r[n + 1]*r[n + 1] - r[n]*r[n]) - deadarea[n];

std::cout << "**************************" << std::endl
<< "rawarea[" << n << "] = " << area[n]+deadarea[n] << std::endl;
std::cout << "area[" << n << "] = " << area[n] << std::endl;
std::cout << "deadarea[" << n << "] = " << deadarea[n] << std::endl;

for(int k = 0; k < imax; k ++){

int flag = 0;

if((r[n]*r[n] <= (x[k] - centerx)*(x[k] - centerx)
+ (y[k] - centery)*(y[k] - centery)) && ((x[k] - centerx)*(x[k] - centerx)
+ (y[k] - centery)*(y[k] - centery) < r[n + 1]*r[n + 1])){



for(m = 0; m < deadnum; m ++){

flag = 0;


if((y01[m] <= y[k]) && (y[k] < y02[m])){

flag = 1;


}

else if((y[k] < y01[m]) || (y02[m] <= y[k])){

flag = 0;


}


}
```

```
if(flag == 0){

entry[n] = entry[n] + 1;
}

}

}

flux[n] = entry[n] / area[n];
allflux = allflux + flux[n];

if(flux[n] > maxflux){
maxflux = flux[n];
}

std::cout << "entry[" << n + 1 << "]=" << entry[n] << std::endl;
std::cout << "flux = " << flux[n] << std::endl;

}


for(n = 0; n < level; n ++){
fprintf(fp2, "%10.4f\t%10.4f\n", mid[n], flux[n]);
share[n] = flux[n] / maxflux * 100;
fprintf(fp3, "%10.4f\t%10.4f\n", mid[n], share[n]);
}


double allshare = 0;
for(n = 0; n < level; n ++){
allshare = allshare + share[n];
}
std::cout << "allshare=" << allshare << std::endl;

fclose(fp);
fclose(fp2);
fclose(fp3);

return 0;
```

```
}
```

謝辞

今回の実験を進めるにあたり、宇宙線研究室の身内さんには、実験の進め方、実験機材の扱い方、データの解析方法など、実験全般に渡ってご指導していただきました。お忙しい中本当にありがとうございました。TA の上野さんには、μ-PIC にトラブルが起こるたびに深夜まで、時には明け方まで一緒になって原因を考えていただき、大変お世話になりました。

μ-PIC のことをいろいろと教えてくださった西村さん、服部さん、瀧川さん、線源を借りる際などにお世話になった内山さん、黒澤さんをはじめ、宇宙線研究室のみなさまにもお世話になりました。ありがとうございました。

最後に、オシロスコープなど実験機材を僕らと奪い合ったシンチ班の 2 人、ともに Cool-X に振り回された副島君、殺伐とした P6 部屋の雰囲気を和ませてくれた解析班の 2 人、みんなとともに一年間、実験やゼミができて楽しかったです。ありがとう。

参考文献

[1] 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).

[2] F.Sauli:”Properties of operation of multiwire proportional and drift chambers”, $CERN Report$(1977)77-09.

[3] M.S.Longair:”High Energy Astrophysics”,$CambridgeUniversityPress$(1992).

[4] 高田淳史:修士論文, 京都大学 (2004).

[5] K.Hattori et al.:$MicroPixelChamberOperationwithGasElectronMultiplier$,Kyoto Univ.(2005).

[6] XCOM:Photon Cross Section Database (http://physics.nist.gov/PhysRefData/Xcom/Text/XCOM.html).

[7] X-Ray Generator:Pyroelectric Crystal (http://www.amptek.com/coolx.html).