Linuxでのテストベンチ用データ収集システムの開発

渡辺 崇臣

1999/4/2

目 次

第1章　目的

実験の目的はLINUX上で動くデータ収集システムUNIDAQを実際に動かす事である。また実際にデータをとれるようになった後は、イベントが発生してからのUNIDAQの反応までの時間を色々な条件で測定する。

第2章　実験装置

今回の実験で使った物は次の通りである

- 各CAMACモジュール
- ISAcc7000 クレートコントローラー
- GATEWAY DOS/V機 (UNIX,PentiumII 350MHz)
- AQUA DOS/V機 (MS-DOS,Pentium 200MHz)
- NaIシンチレーター2個

第3章　CAMAC

3.1　CAMAC

CAMACは素粒子・原子核実験に使われる回路モジュールの規格である。

CAMACはADC等の各モジュールとそれらのモジュールを複数収納出来るクレートからなる。又モジュールにはコンピュータとの仲介をするクレートコントローラという特殊なモジュールが在る。クレートコントローラと各モジュールは、クレートの背面にあるデータバスと命令等を送るバスでつながっている。

3.2　CAMACドライバ

これはコンピュータからCAMACに命令を送る関数群で、今回はKEKにあるものを使った。これを使ってUNIDAQはCAMACを制御しデータをとっている。

これを使い実験をする時重要となるのは、各モジュールに命令を送る時に必要となるNAFで、Nは命令を出すモジュールがささっているクレートの場所、Aはモジュール内のチャンネル、Fは実行する命令をそれぞれ表している。

第4章　UNIDAQ

4.1　UNIDAQの特徴

UNIDAQの特徴は主にUNIDAQの動いているOS・LINUXの特徴によるものである。そのLINUXの特徴はマルチタスク・ネットワークOSだという事である。

まずマルチタスクによる長所としての特徴は、データ収集等のプロセス毎にプログラムが分かれている事である。これによりプログラムのメンテナンスが楽になる。逆に短所としては、イベントが発生してからデータをとりに行くまでが遅いという事である。この理由はLINUXが割り込み（この場合はイベントがあったという信号）に対する反応が、他のプログラムを動かしているために遅くなる為である。

次にネットワークOSという事による長所は2つある。まず複数のマシンにより測定する事で個々のマシンの負荷を減らす事ができるという事である。これは例えば、3台のマシンを用意してそれぞれデータ収集・記録・解析を行えば、データ収集を行うマシンはそれのみにCPUを使う事が出来るので不感時間が少なくなるという事である。次にtelnetでUNIDAQを動かすコンピュータに入る事で遠隔操作できることである。これは実験を行いデータ収集を行うマシンがある部屋がα線等があり実験中には入れない時等、コンピュータの設定を変えたりする為にわざわざ実験をとめる必要がなくなるという利点が有る。

また実験データを測定中にリアルタイムで表示させる時、KODAQでは新しいヒストグラムを作るのが難しいそうだがが、UNIDAQではanalyzerで新しいヒストグラムを作事と、そこにデータを入れるという2行を加えるだけでPAWで簡単に見る事ができる。

4.2　UNIDAQのプロセス

今回の実験で使ったプロセスは次のものである

- collector

  これはイベントに直接関わるプロセスである。

  ここではイベントが発生するまでひたすら待ちつづけ、イベントが発生した事を認識するとCAMACにアクセスしてデータを読み出し、そのデータをバッファーに記録する。

- recorder

  バッファーからデータを受け取り、それをディスク等に記録する。

- analyzer

  UNIDAQ には標準として 3 種類の analyzer が用意されているが、今回使ったのは Global Section Analyzer というものである。これは PAW を使って解析をするために、共有メモリー上にヒストグラムを作っていく。ヒストグラムを作る時は HBOOK を使っている。

- buufer manager

  collector 等のプロセスはそれぞれ別になっていてデータを直接共有出来ないので、データをバッファーに入れてやり取りをする。そのバッファーを管理・運用しているプロセスかこの buffer manager である。

  バッファーを決められた数・量だけ共有メモリー上に確保して、プロセスから要求があればバッファーの先頭アドレスを教える事によってバッファーの受け渡しをする。ここで buffer manager はプロセスの priority とバッファーの significance というものを見て、その合計が sthreshold を超えていたら必ずそのプロセスにはバッファーを渡す、という判定をしている。これはイベント数が多い時、解析によってデータ収集に支障が出そうであれば analyzer の priority を低くし回ってくるバッファーの数を押さえれば、それだけ負荷が減り不感時間を少なくする事ができるようになる。

今回使わなかったものには、ネットワークを介してデータバッファが送られてきた時に使う reciever、イベントのパラメータをディスプレイ上で見る dataview 等が有る。

第5章　^{60}Coのγ線の測定

5.1　目的

この実験では^{60}Coのγ線の測定を行い、UNIDAQでデータを正しくとれているかを確かめる。

5.2　測定方法

図5-1のような回路を使い、ADC1・2の両方に同時にγ線が入った時にデータをとり、ADC1・2それぞれのエネルギースペクトルと、ADC1・2のデータを2次元ヒストグラムにプロットしたものを作る。

5.3　測定結果

ADC1・2それぞれのエネルギースペクトルは図5-2・5-3で、1.17・1.33Mevの2本のγ線のピークとコンプトン散乱の部分がはっきり見えている。また図5-4はADC1・2のデータを2次元ヒストグラムにプロットしたもので、領域1の部分はADC1に1.17Mevのγ線が入った時にADC2には1.33Mevのγ線が入っいる事、又その逆を表しており、これは^{60}Coから出てくる2本のγ線が別々に出てくるのではなく、わずかな時間で連続的に崩壊するカスケード崩壊を起こしていることを示している。又領域2は一方がピークでもう一方がコンプトン散乱、領域3は両方ともコンプトン散乱の時のデータをプロットしている。

これらからこのUNIDAQではデータを正しくとる事が出来たと分かった。

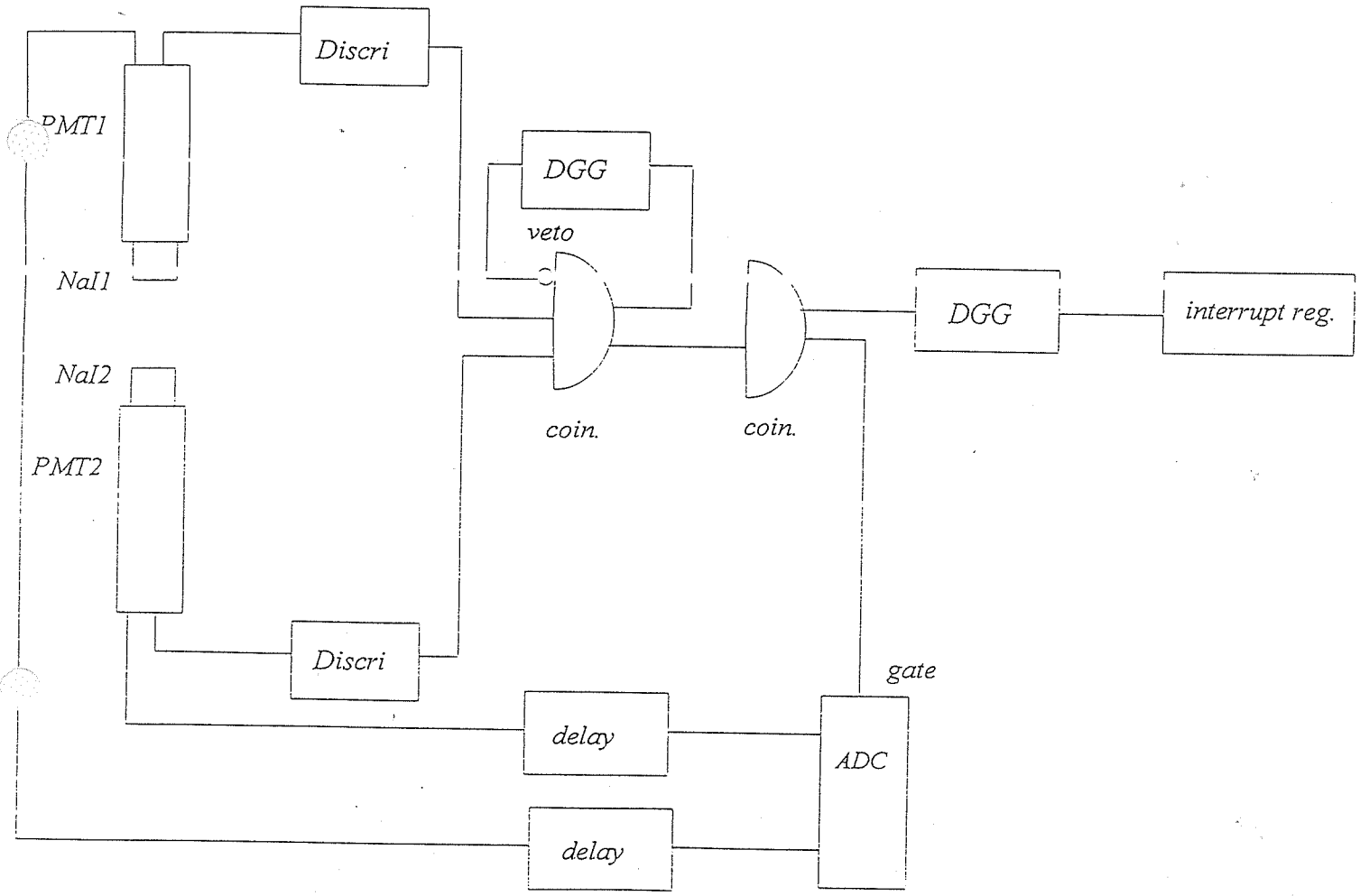
Discri
PMT1
DGG
veto
NaI1
NaI2
PMT2
coin.
coin.
DGG
interrupt reg.
Discri
gate
delay
ADC
delay

図5-1

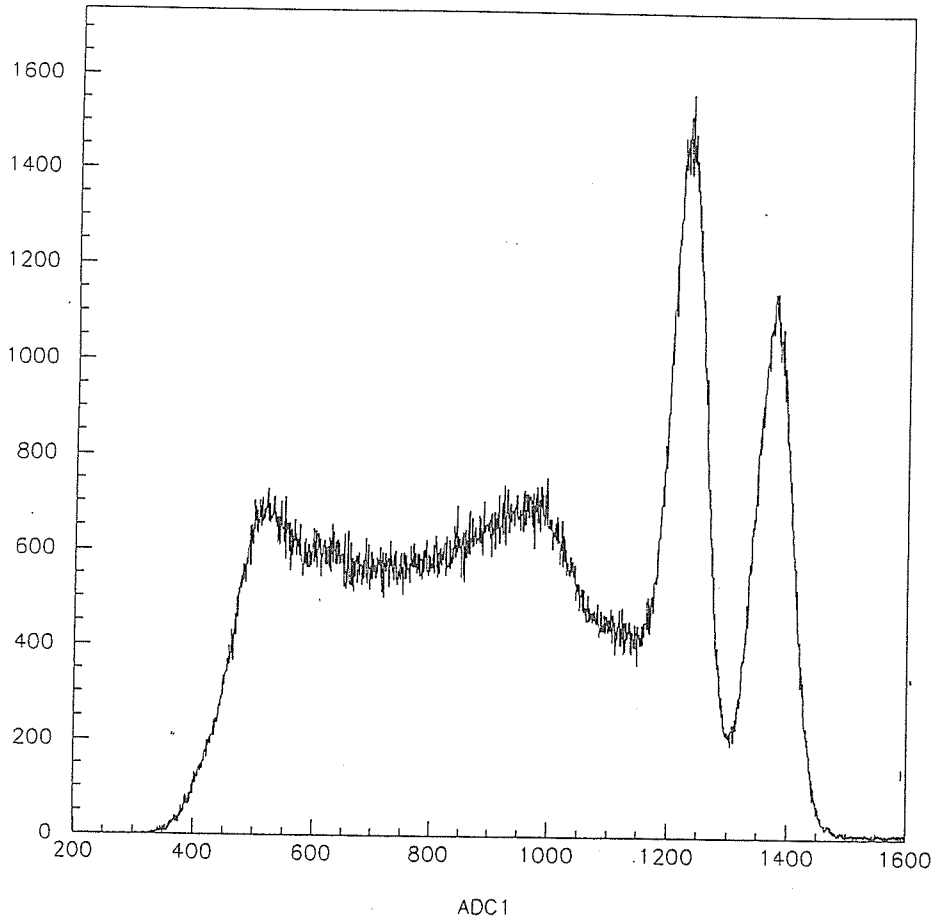
1600
1400
1200
1000
800
600
400
200
0
200
400
600
800
1000
1200
1400
1600
ADC1

図5-2

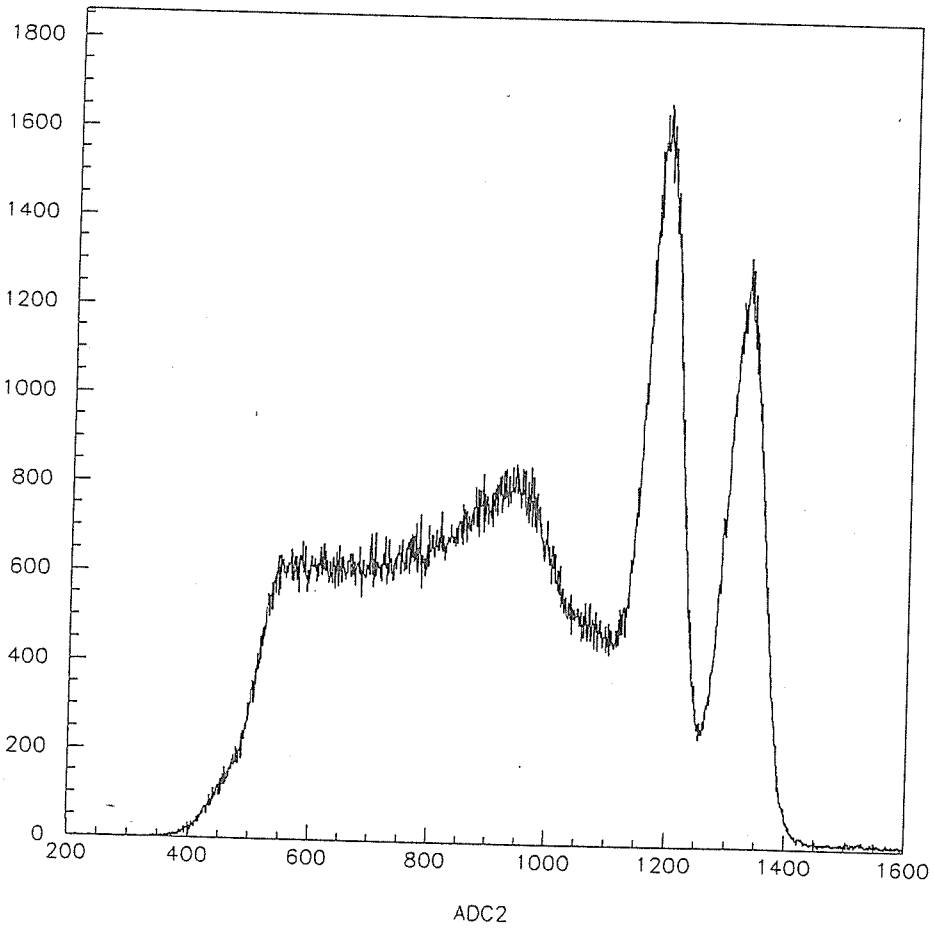
1800
1600
1400
1200
1000
800
600
400
200
0
200
400
600
800
1000
1200
1400
1600
ADC2

図5-3

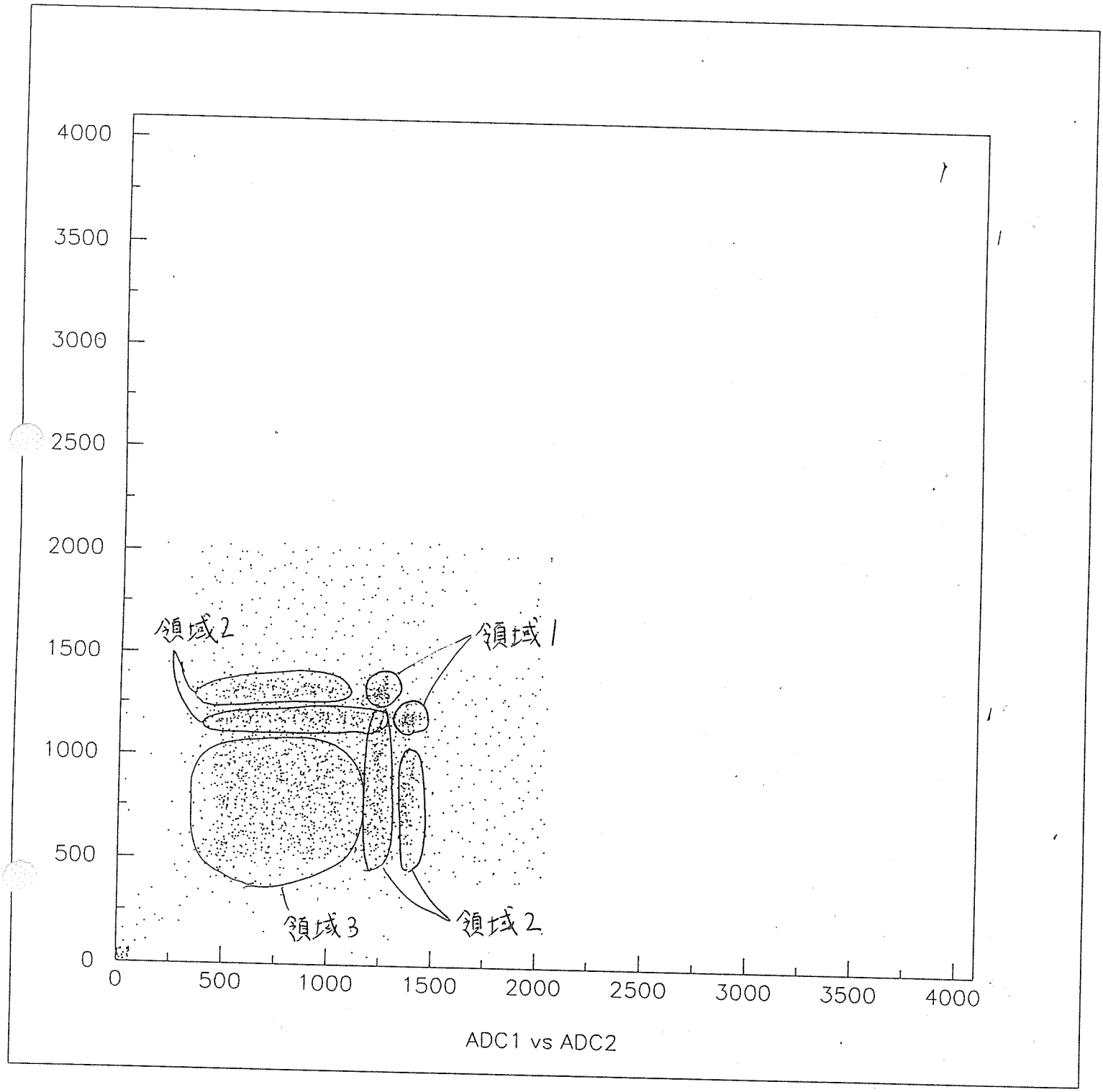
4000
3500
3000
2500
2000
1500
1000
500
0
0
500
1000
1500
2000
2500
3000
3500
4000
領域2
領域1
領域3
領域2
ADC1 vs ADC2

図5-4

第6章　割り込み応答遅れ時間の測定

6.1　目的

ここではイベントが起こりコンピュータに割り込みがかかってから、コンピュータがデータを読みに行くまでにかかる時間を測定する。

応答時間が長ければそれだけ ADC 等はデータを保存しておかなければならず、その間は新しいイベントが起こっても測定出来ない不感時間が長くなるので、この時間は出来るだけ短い方がいい。

測定は UNIDAQ より応答が早い KODAQ との比較と、UNIDAQ を使っている時他の作業をさせた時応答がどのくらい遅れるのかを測定する。

6.2　測定方法

回路を図 6-1 のように組み、interrupt reg. にと同時にオシロスコープの CH 1 に信号が入るようにしそこでトリガーをかける。コンピュータが割り込みに反応したら output reg. にデータを書き込むようにして、その信号をオシロスコープの CH2 に入れる。そして CH1 と CH2 の信号の時間差からコンピュータが反応までにどのくらいかかったかを測定した。

測定した条件は、UNIDAQ では

1. UNIDAQ のみ動かした時
2. UNIDAQ と同時に printf 文を実行し続けた時
3. UNIDAQ と同時に disk access を実行し続けた時

について、KODAQ では

1. KODAQ のみ動かした時

である。

6.3 測定結果

測定結果は下表の通りになった。

また図6-2・6-4はそれぞれ、UNIDAQでの1・2、図6-3はKODAQでの1の条件でのデジタルオシロスコープの画面である。UNIDAQでの3の条件については、表の通りに反応まで時間がかなりの時間がかかるためにあまりトリガーがかからず、そのため分かりずらくなってしまったのでその条件でのオシロスコープの画面はとらなかった。

	UNIDAQ	KODAQ
DAQ以外実行せず	28-36 μ sec	18-26 μ sec
printf 実行	多くは 34-46 μ sec その他の大部分は 100 μ sec 以内	データ無し
disk access 実行	早いのもでも 60 μ sec 遅いものは 1msec 以上かかる	データ無し

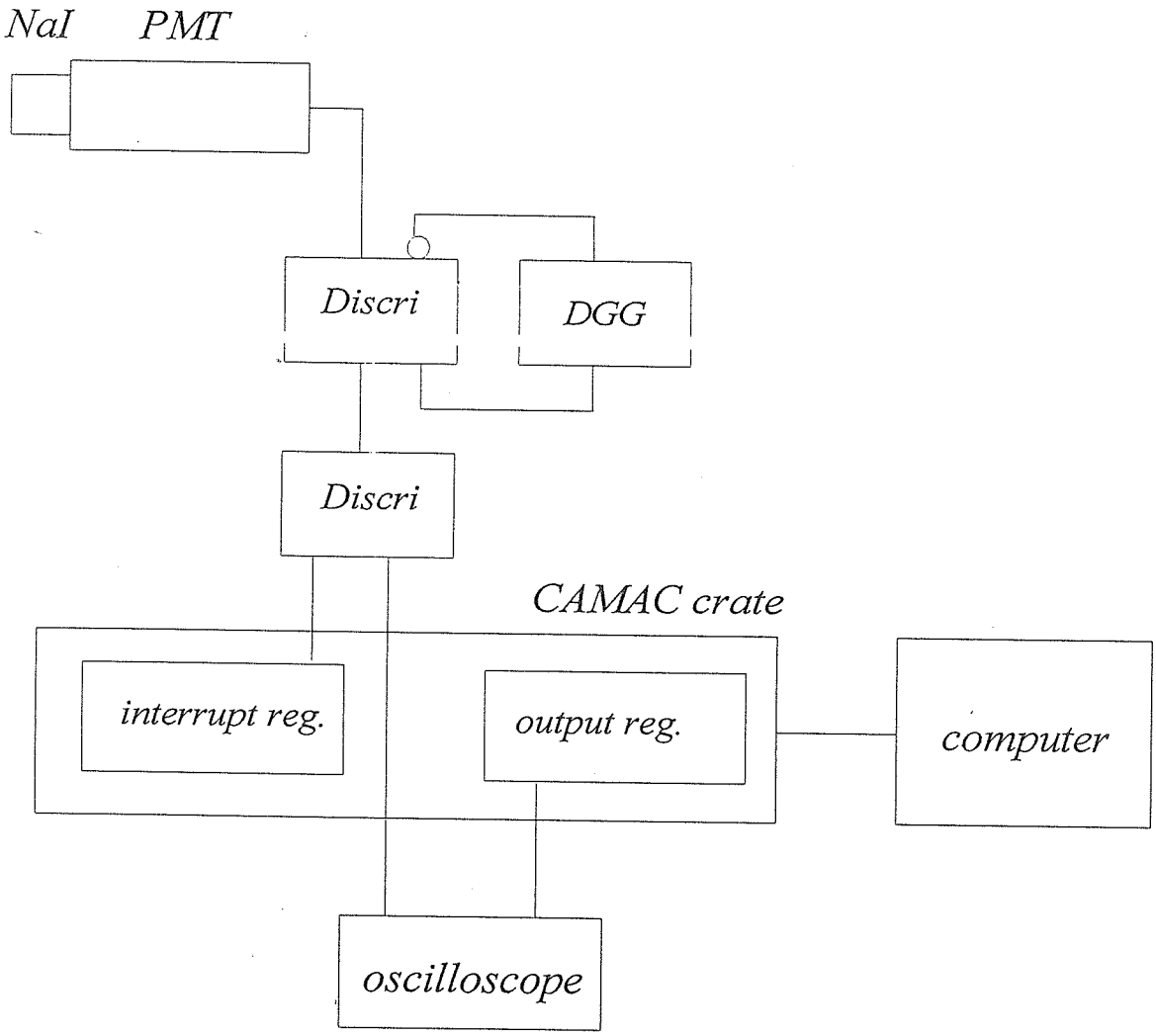
NaI
PMT
Discri
DGG
Discri
CAMAC crate
interrupt reg.
output reg.
computer
oscilloscope

図6-1

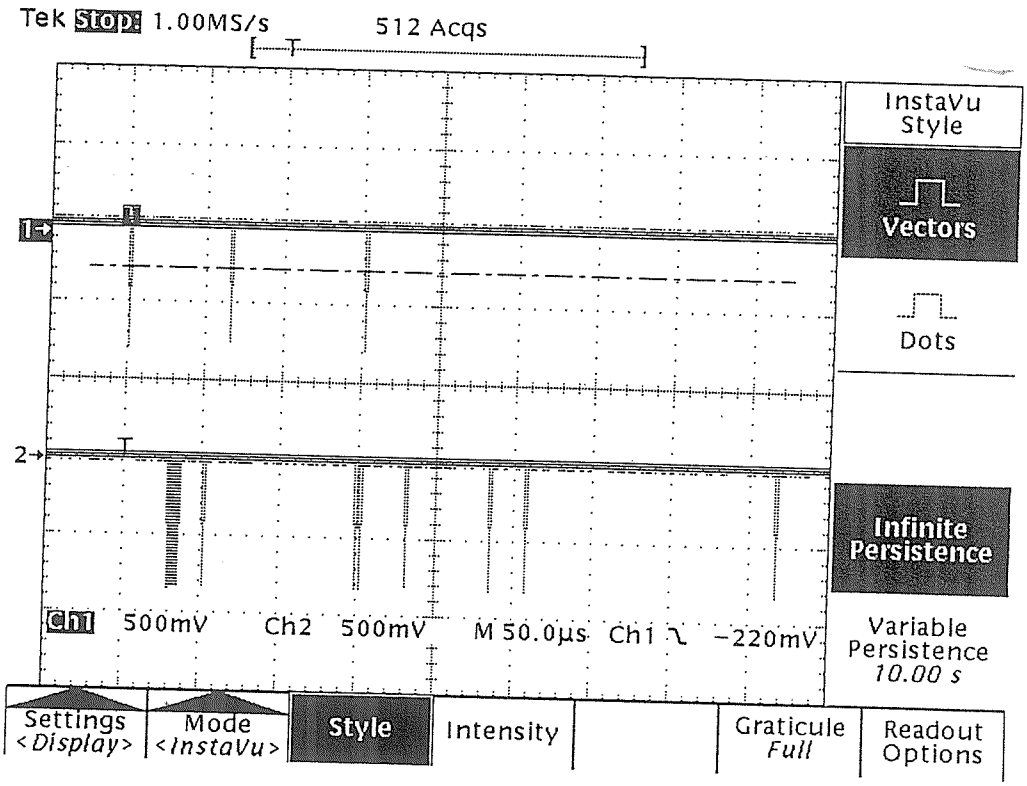
Tek Stop: 1.00MS/s
512 Acqs
InstaVu Style
Vectors
Dots
Infinite Persistence
Variable Persistence 10.00 s
Ch1 500mV Ch2 500mV M 50.0µs Ch1 ∖ −220mV
Settings <Display>
Mode <InstaVu>
Style
Intensity
Graticule Full
Readout Options

図6-2

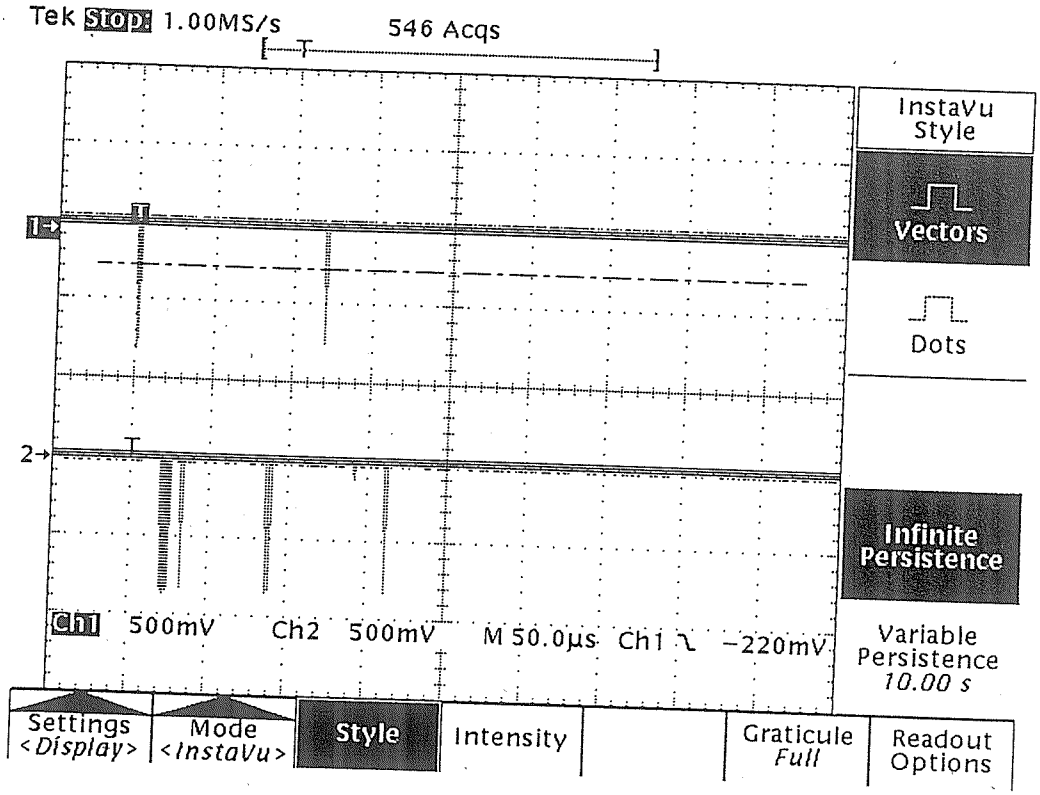
Tek Stop: 1.00MS/s
546 Acqs
InstaVu Style
Vectors
Dots
Infinite Persistence
Variable Persistence 10.00 s
Ch1 500mV Ch2 500mV M 50.0µs Ch1 ∖ −220mV
Settings <Display>
Mode <InstaVu>
Style
Intensity
Graticule Full
Readout Options

図6-3

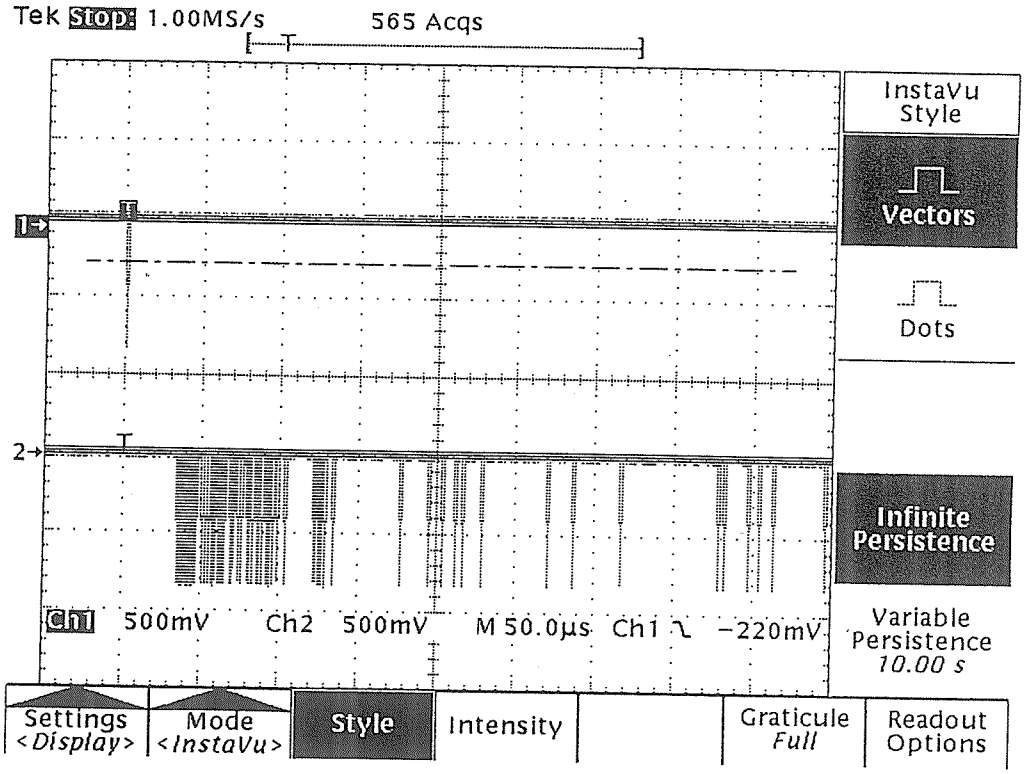
Tek Stop: 1.00MS/s 565 Acqs
InstaVu Style
Vectors
Dots
Infinite Persistence
Variable Persistence
10.00 s
Ch1 500mV Ch2 500mV M 50.0µs Ch1 ∿ −220mV
Settings <Display>
Mode <InstaVu>
Style
Intensity
Graticule Full
Readout Options

図 6-4

第7章　まとめ

実際にUNIDAQを使いデータをとり始めることができたが、まだ使い易いプログラムにはなっていないので、実験をする時のカスタマイズ・操作が簡単になるように手直ししていく。具体的には、新しい実験を始める時に手直しする部分を少なく分かりやすくする。また実験をする時、今はコマンドを打ち込んで命令を出すようにしているが、アイコンを置きそれをクリックすると実験を始められるようにしていきたい。

割り込み応答遅れ時間については、UNIDAQはKODAQに比べ50%程遅かったが、メリットも併せて考えると十分実用的だと思われる。他の命令を実行していた場合はやはり応答が遅れるので、測定中にはそのシステムで他に何もしないことが望まれる.。またディスクアクセスをした時は大幅に応答時間がながくなっているので、ネットワークを介してrecorderをcollectorと別のマシンで実行すると、反応時間は早くなるのではないかと思われる。

参考文献

文献

- Leo　Techniques for Nuclear and Particke Physics Experiments

マニュアル

- Installation Guide of CANAC Library and Driver package (Version 2.0.29c) for Linux OS and ISA/PCI-CC7x00
- R. Ball　UNIDAQ v2.3 User's Guide
- Masahara NOMACHI Y.TAKEYCHI

  NOVA buffer manager for SDC portable DAQ Users manual

www 関連

- 佐藤 一道　スケーラブルな量子線計測用データ収集・解析システムの開発研究

  http://sp8sun.spring8.or.jp/ kazusato/paper/paper.html