第 2, 3, 4 回

ビームを使ったコミッショニング

Itaru Nakagawa, Rachid Nouicer, Maya Shimomura, Genki Nukazuka,
Raul Cecato, Joseph Bertaux, Mai Kano, Manami Fujiwara,
Cheng-Wei Shih (remote)

コミッショニング

第 2, 3 回: bitmodes スキャン

目的

- パラメータ bitmodes の値を変えながらデータを収集し、ビーム・ビーム衝突によるヒット検出に最適化すること

状況

- 2023/05/30, 6/2
- 100 GeV 金・金衝突 ❓
  - 28 × 28 ビームバンチ（run7314 - 8126）
- MBD トリガーを使ったデータ収集
- intt2 のみ（第 2 回）, 全 FELIX（第 3 回）

データ

- イベントファイル：/bbox/commissioning/INTT/beam/
- ROOT ファイル：/home/phnxrc/INTT/commissioning_5_30/
  /home/phnxrc/INTT/commissioning_6_2/

第 4 回: DAC スキャン

目的

- DAC 設定を変えながらデータを収集し、MIP ピークを見る

状況

- 2023/06/04
- 100 GeV 金・金衝突 ❓
  - 56 × 56 ビームバンチ（run 9301 - 9335）
- MBD トリガーを使ったデータ収集
- シフターが設定変更、ラン開始・終了した

データ

- イベントファイル：/bbox/commissioning/INTT/beam/
- ROOT ファイル：/home/phnxrc/INTT/commissioning_6_4/

第 2 回: modebits スキャン

- modebits だけを変えながらデータ収集
  - LV1 Delay: 25（十分短いディレイ）
  - n_collisions: 0 or 2
  - open_time: 35
- intt2 のみ使用
- ヒットレートが
- Golden data: commissioning_5_30/hit_files/calib_intt2-00008118-0000.root
- Joseph, 糠塚・加納のオンライン解析でタイムインを確認した
- 詳しくは加納が報告

modebits スキャンのラン一覧

Run	内容
8102 - 8115	n_collisions=2 でスキャン
8116 - 8125	n_collisions=0 でスキャン

第 3 回: modebits スキャン

- modebits だけを変えながらデータ収集
  - LV1 Delay: 25（十分短いディレイ）
  - n_collisions: 0, 2 or 4???
  - open_time: 35
- 全 FELIX 使用
- ヒットレートが極端に減少し始める値を探す

第 4 回: DAC スキャン

- DAC 設定を変えながらデータ収集
  - LV1 Delay: 25
  - n_collisions: 4
  - open_time: 35
  - modebits: 78
- スキャンデータを規格化して繋げば全 DAC 領域に渡って詳細な分布が得られる
- 今のとこ杉山、Cheng-Wei が興味を示しているらしい

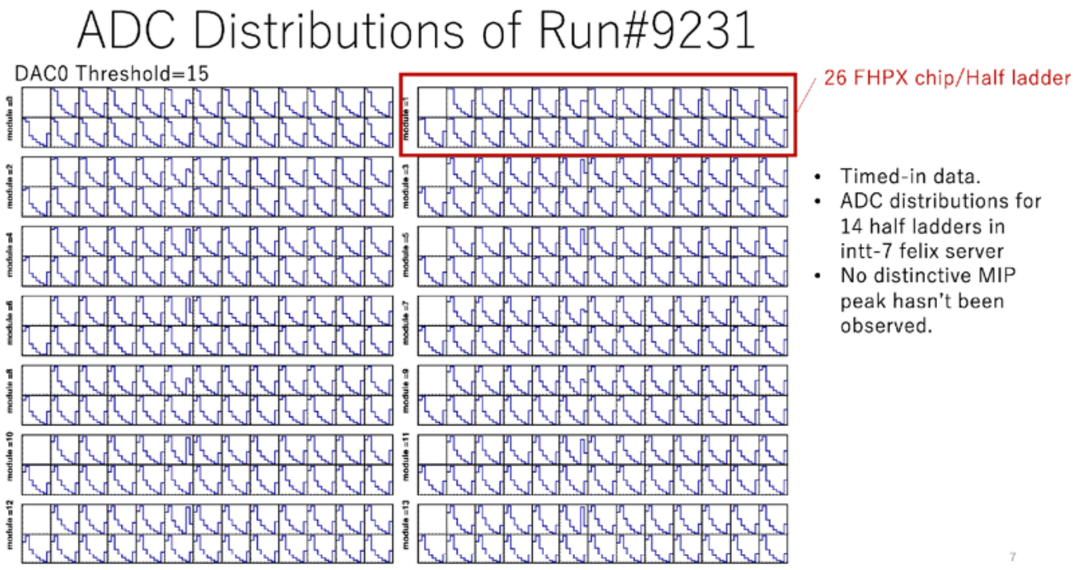
DAC 設定値

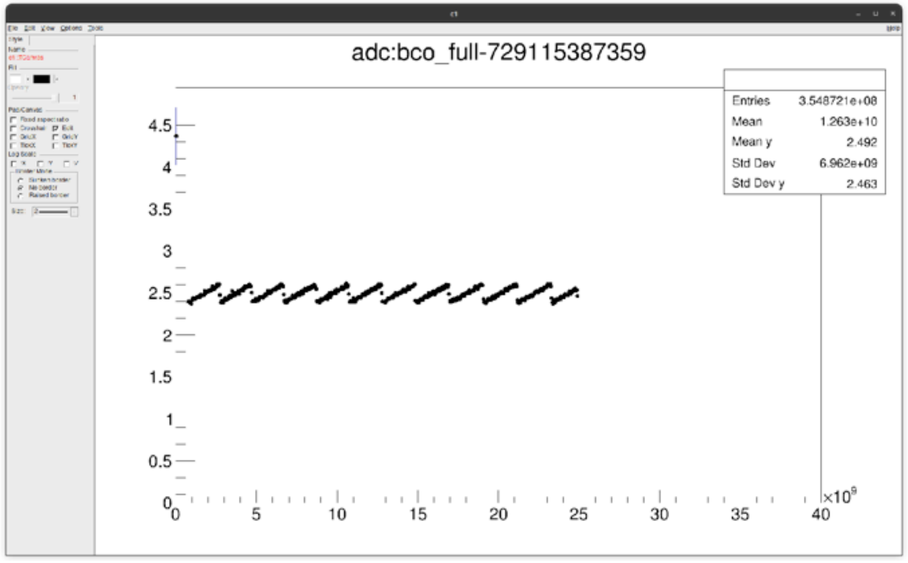
Run	Scan	DAC0	DAC1	DAC2	DAC3	DAC4	DAC5	DAC6	DAC7
9303	1	8	12	16	20	24	28	32	36
9314	2	28	32	36	40	44	48	52	56
9318	3	48	52	56	60	64	68	72	76
9319	4	68	72	76	80	84	88	92	96
9320	5	88	92	96	100	104	108	112	116
9322	6	108	112	116	120	124	128	132	136
9329	7	128	132	136	140	144	148	152	156
9333	8	148	152	156	160	164	168	172	176
9334	9	168	172	176	180	184	188	192	196
9335	10	188	192	196	200	204	208	212	216

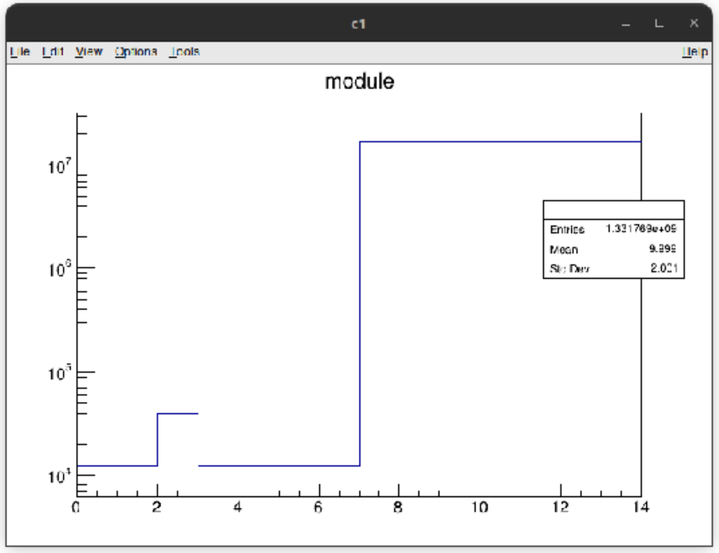
その他やれること

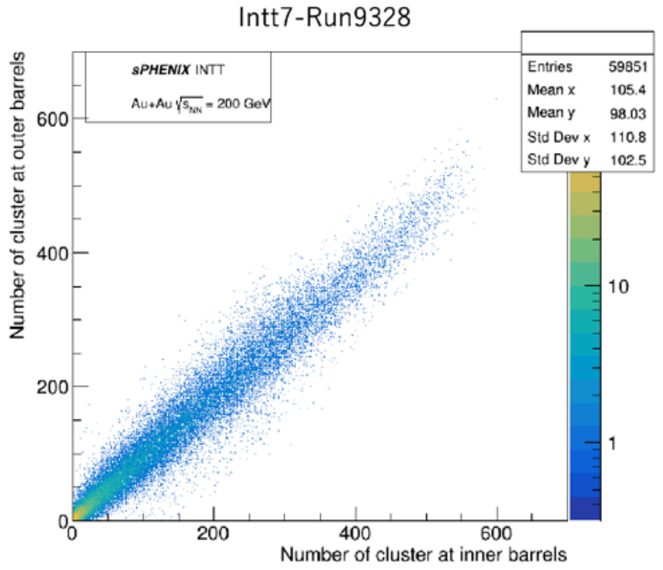
- ADC 分布の理解
- intt1 の挙動の理解
- タイムインしている・していないデータの比較
- イベントディスプレイ（DST が必要）
- 他検出器との相関

  等など









これから

- open time スキャン
- DAC0 スキャン
- Big Partition でキャリブレーション
- バイアス電圧を変えて測定

  等など

データ

TTree（ヒットベース）

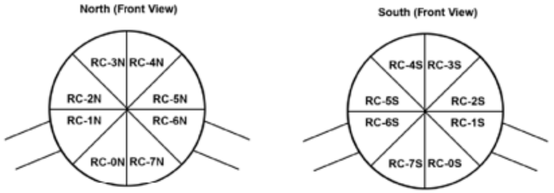
- ブランチは基本的に int 型
- FVTX テストベンチの構成を引き継ぎ、ブランチがいくつか追加されている
  - pid:パケット ID
  - adc
  - ampl
  - chip_id
  - module: FELIX readout ch
  - chan_id
  - bco
  - bco_full: Long64_t
  - event
  - roc
  - barrel
  - layer
  - ladder
  - arm: 0 (south), 1 (north)
  - full_fphx
  - full_roc

roc, barrel, layer, ladder, arm について

ROC の記述 (例 RC-2N) や

ラダーの記述 (例 B1 L0 14S) を再現できる変数

RC${roc}${arm}
B${barrel}L${layer}${ladder}${arm}



TTree（イベントベース）

- 蜂谷さんの InttEvent クラス（INTT/hachiya/convertInttRaw/test1/InttEvent.h/cc）、
  デコードプログラム（INTT/hachiya/convertInttRaw/test1/runConvertInttData.C）でとりあえず ROOT ファイルを作成している
- ファイル名：beam_intt?-${run}-${chunk}_event_base.root
- 解析のやりかた

データ・イベントベースツリーの解析方法

```
#include "InttEvent.cc"
#include "InttCluster.cc"

int macro()
{

  TFile* tf = new TFile( file_path.c_str(), "READ" ); // ROOT ファイルを開く
  if( tf == nullptr )                                 // もしファイルが開けなかったとき
    {
      cerr << file_path << " is not found." << endl;
      return -1;
    }
  TTree* tr = (TTree*)tf->Get( "tree" ) ;             // TTree を取得する

  InttEvent* ev = new InttEvent();                    // InttEvent クラスのインスタンスを作る
  tr->SetBranchAddress( "event", &ev );               // 作ったクラスをツリーのブランチに対応させる

  for( int i=0; i<tr->GetEntries(); i++ )             // ツリー内のイベントについてループ
    {
      tr->GetEvent( i ) ;                             // i 番目のイベントを取得する。
      /* やりたいことを書く */                        // イベントの内容は ev に代入される
    }
```

データ処理・保存

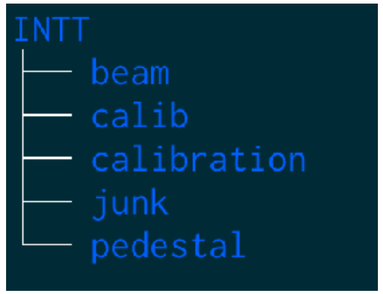
現在のデータ処理状況

- INTT DAQ サーバー内でイベントファイルをデコードし、ROOT ファイルを生成している
  - 場所：/home/phnxrc/INTT/commissioning_*_*
- ROOT ファイルを inttdaq で解析し、ADC 分布とチャンネル分布を見ている
- デコードプログラムのメモリ使用量が多い。

  大きいファイル (> 1GB) を処理するとそれなりの量 (サーバーの全メモリ128GB のうち 20%~30%) を消費する。
- デコード処理を 8 台のサーバーに分散させて行っているが、データ収集に干渉しそう

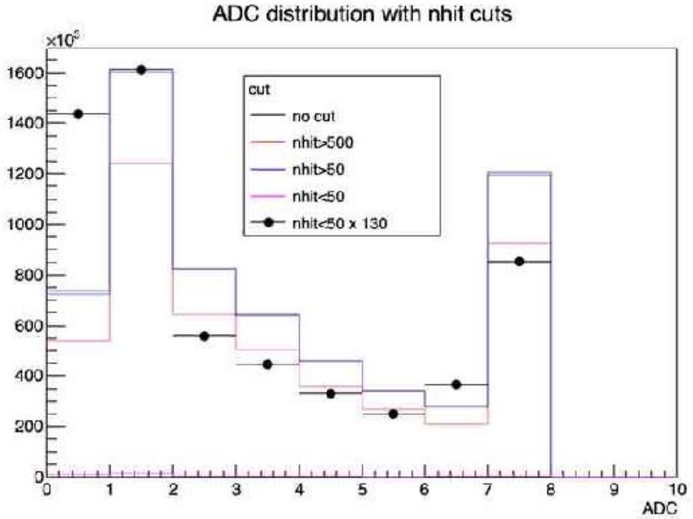
データ処理の予定・保存場所の予定

- バッファーボックスのイベントファイルを SDCC のディスク (?) に転送する
  - rcas 内から見れる場所へ置く
  - 予定地：/sphenix/lustre01/sphnxpro/commissioning/INTT →
  - sPHENIX 用の解析アカウントが必要
- rcas でデコード・ ROOT ファイルを生成する
  - 予定地：/sphenix/tg/tg01/commissioning/INTT/root_files (?)
- 準備はまだ始めていない



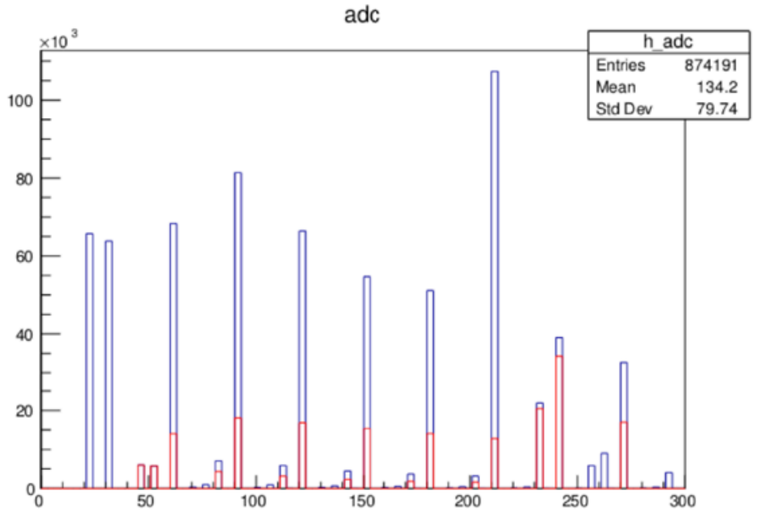
興味深い点

- 全ラダー（？）のチップ 26 でデータが全く取れていない
- ADC 分布に明確な MIP ピークが見えない
- ヒットのクローンがあるかもしれない



Multiplicity ごとに分割した ADC 分布

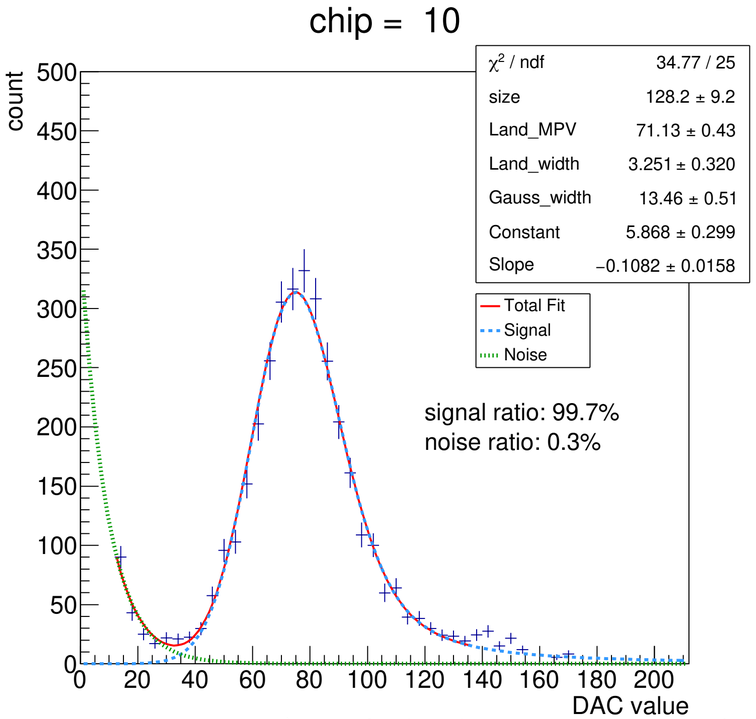
下村



クラスタリング後の DAC 分布

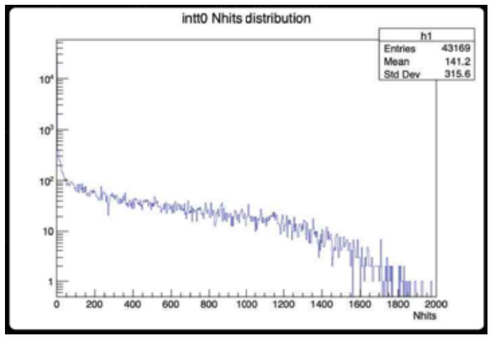
蜂谷

パラメータ名	値	パラメータ名	値
DAC0	23	DAC4	120
DAC1	30	DAC5	150
DAC2	60	DAC6	180
DAC3	90	DAC7	210



ELPH テストビーム実験で得られた DAC 分布

杉山

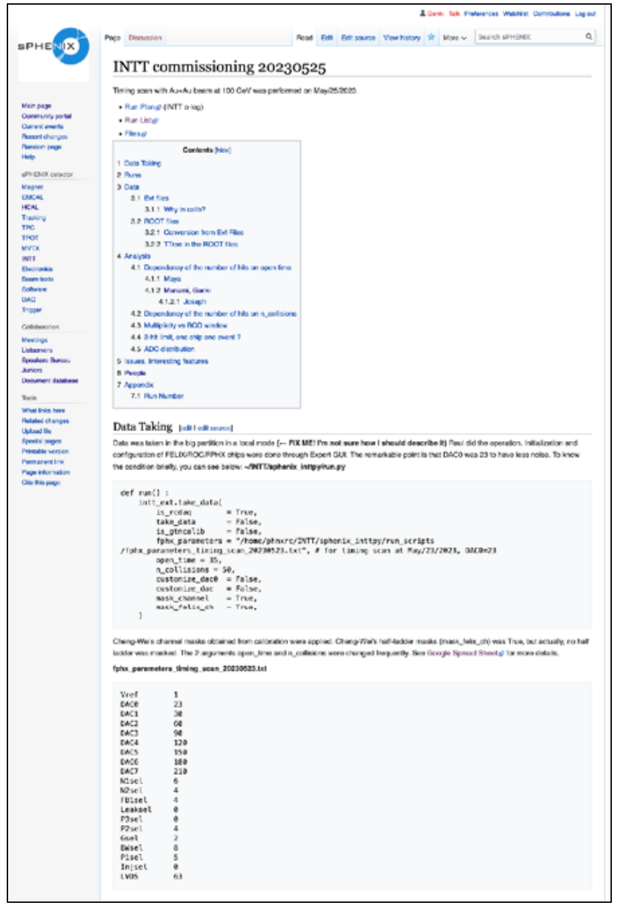


Multiplicity 分布

下村

sPHENIX Wiki

- リンク
- 執筆がデータ収集に追いついていない



INTT commissioning 20230525

Timing scan with Au+Au beams at 100 GeV was performed on May/25/2023.

- Run Plan@ (INTT e-log)
- Run List@
- Files@

Contents [hide]

1 Data Taking
2 Runs
3 Data
3.1 Evt files
3.1.1 Why in cols?
3.2 ROOT files
3.2.1 Conversion from Evt Files
3.2.2 TTree in the ROOT files
4 Analysis
4.1 Dependency of the number of hits on open time
4.1.1 Maya
4.1.2 M. Matsuda, Gianni
4.1.2.1 Joseph
4.2 Dependency of the number of hits on n_collisions
4.3 Multiplicity vs ROD window
4.4 3 Hit trial, one chip one event ?
4.5 ADC distribution
5 Issues, Interesting features
6 People
7 Appendix
7.1 Run Number

Data Taking [edit | edit source]

Data was taken in the big partition in a local mode (<-- FIX ME!! I'm not sure how I should describe it) Paul did the operation. Initialization and configuration of FELIX/ROC/FPHX chips were done through Expert GUI. The remarkable point is that DAC0 was 23 to have less noise. To knew the condition briefly, you can see below: ~/INTT/sphenix_intt/py/run.py

```
def run() :
    intt_ext.take_data(
        is_rcdaq        = True,
        take_data       = False,
        is_gtmcalib     = False,
        fphx_parameters = "/home/phnxrc/INTT/sphenix_intt/py/run_scripts
/fphx_parameters_timing_scan_20230523.txt", # for timing scan at May/23/2023, DAC0=23
        open_time = 35,
        n_collisions = 50,
        customize_dac0  = False,
        customize_dac   = False,
        mask_channel    = True,
        mask_felix_ch   = True,
    )
```

Cheng-Wei's channel masks obtained from calibration were applied. Cheng-Wei's half-ladder masks (mask_felix_ch) was True, but actually, no half ladder was masked. The 2 arguments open_time and n_collisions were changed frequently. See Google Spread Sheet@ for more details.

fphx_parameters_timing_scan_20230525.txt

```
Vref      1
DAC0      23
DAC1      30
DAC2      60
DAC3      90
DAC4      120
DAC5      150
DAC6      180
DAC7      210
N1sel     6
N2sel     4
FBlsel    4
Leaksel   0
P3sel     0
P2sel     4
Gsel      2
Bsel      0
Psel      5
Injsel    0
LVDS      63
```