ATLAS 検出器前後方ミューオンシステムの ソフトウェアコミッショニング

東京大学大学院理学系研究科物理学専攻
坂本研究室

Kanega Fumihiko
金賀 史彦

2009 年 1 月 20 日

Introduction

スイスのジュネーブにある研究所 CERN で、高エネルギー (重心系エネルギー 14TeV) の陽子ビーム同士を衝突させる実験が進められている。その衝突点の一つに配置されているのが ATLAS 検出器であり、標準模型で未だ未発見とされているヒッグス粒子、また標準模型の枠組み外で予言される超対称性粒子の発見が期待されている。

ビーム衝突は今学期中に行われる予定だったが、あいにく延期されて、少なくとも 2009 年 4 月まで、ビームが出ることはなくなってしまった。しかし、ATLAS 検出器はシングルビームでのデータ収集が成功し、システムがきちんと動作することを証明した。

高エネルギーのハドロン同士の衝突なので、バックグラウンドは膨大となる。そこで、興味のあるイベントについてのみデータを収集しすることが必要となる。ATLAS 検出器は 3 段階のトリガー (ハードウェアによる Level1、ソフトウェアによる Level2、EventFilter) を用いて、データを収集している。

ヒッグスの終状態の一つである、高エネルギー μ 粒子に対してトリガーするミューオントリガーシステムはハードウェアの段階で運動量判別を行う画期的なシステムとなっている。

我々はエンドキャップ部分ミューオントリガー検出器である TGC(Thin Gap Chamber) のチェンバ、エレクトロニクス、ケーブリング、ソフトウェアの動作確認や開発、デバックなどを行ってきた。

TGC は全セクター動作し、データもきちんと収集でき、解析、デバッグが進められている。

本論分では、主に担当したオフラインソフトウェアなどについて中心的に説明し、それらを用いて得られた宇宙線のデータを使い、TGC の解析を行った結果を発表する。

目次

1　素粒子物理と LHC

この章では現在の素粒子物理の現状、そして現在最も最先端の発見が期待される LHC(Large Hadron Collider) についての概要について述べる。

1.1　素粒子物理の現状

1990 年代、素粒子物理は標準模型が実験により高い精度で検証され、力の源がゲージ対称性であることが示された。さらに、世代が 3 つであること、力の統一の可能性、Higgs 粒子の質量の予言 (図 1.1 - 1.5)、標準模型を超える超対称性の可能性 (図 1.6 - 1.7) などが得られている。

Higgs 粒子は未だに見つかっていない。また、超対称性についても未確認である。

1.1.1　標準理論 Higgs 粒子

Higgs 粒子は、質量を与える未知の粒子である。Higgs 粒子を発見し、自発的対称性の破れが質量起源として起こっていることを示せば、ゲージ対称性の正当性がより強く支持されることになり、宇宙の進化の解明などにも重要な役割を果たすことになる。主題の ATLAS 検出器は質量 100GeV から 1TeV の広範囲で Higgs 粒子を探索する能力を持つ。

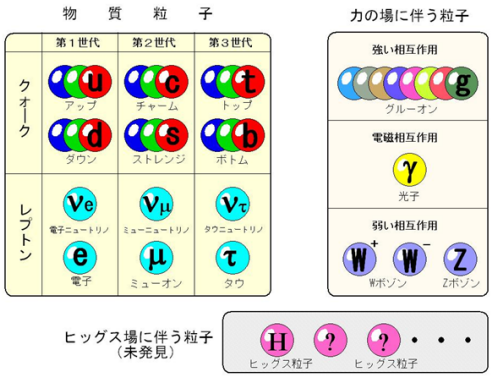


図 1.1　現在の素粒子 [11]

標準模型から考えられる粒子。ヒッグス粒子だけ未発見。

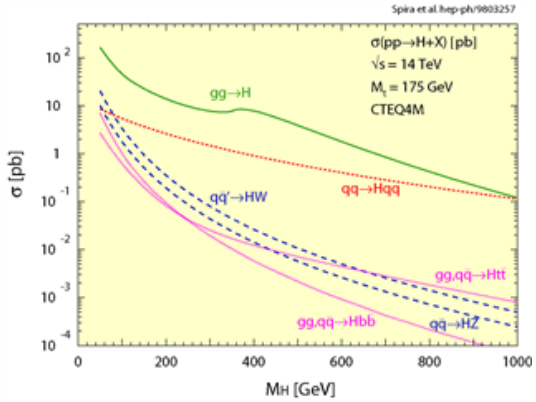


図 1.2　標準模型でのヒッグス粒子の生成断面積 [11]

横軸はヒッグス粒子の質量。縦軸が断面積。gluon fusion が最も生成断面積が大きい。

1.1.1.1　Higgs 粒子の生成過程

Higgs 粒子は重い粒子と結合しやすいため、主に次に挙げる 4 つの生成過程が考えられる。生成断面積と質量の関係を図 1.2 に、主な生成過程のファインマンダイアグラムを図 1.3 に示す

1. $gg \to H$ (gluon fusion)
   トップクォークやボトムクォークのループを介した過程で、最も断面積が大きい。その反面、Higgs 粒子が崩壊して出来る粒子以外に大きな p_T を持つ粒子がなく、バックグラウンドとの選別が非常に難し

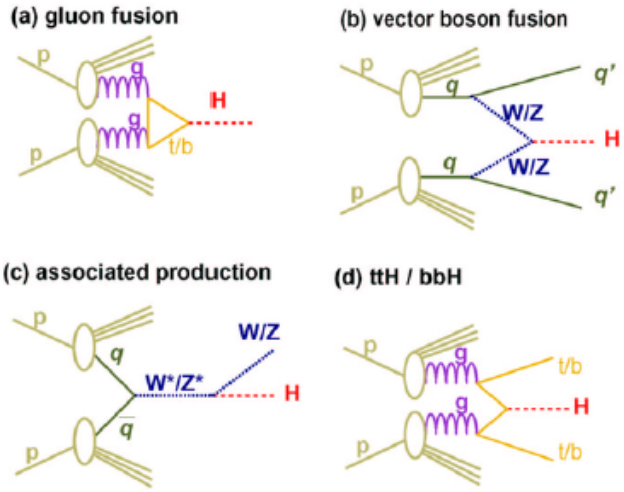


図 1.3　主要な Higgs 粒子生成ファインマンダイアグラム

重い粒子と結合しやすい。

い。$H \to \gamma\gamma ZZ(\to llll), W^+W^+(l\nu l\nu)$ だけが有望な崩壊過程。

2. $qq \to qqH$ (W/Z fusion)
   クォークから放出されたゲージボソンから Higgs 粒子が生成されている。断面積も比較的大きく、反跳したクォークに起因する大きな p_T を持つジェットが 2 本観測される特徴があり、イベントの選別が比較的行いやすい。さらに、イベントに関わる 2 つのクォークの間ではカラー交換が行なわれないので、QCD バックグラウンドによる影響は少ない。この生成過程では、様々な崩壊過程での Higgs 粒子の探索が期待されている。
3. $qq \to (W/Z)H$ (W/Z associate production)
   クォークの対消滅で生成されたゲージボソンから、更に Higgs 粒子が放射される過程。終状態にゲージボソン（W/Z）が観測される特徴がある。このゲージボソンがレプトンに崩壊した場合は、シグナルとバックグラウンドの識別が容易にできる。
4. $qq/gg \to ttH$ (top associate production)
   対生成されたトップクォークから、Higgs 粒子が放出される過程。断面積は小さいが、特徴のあるトップクォークペアを終状態に含んでいるので、QCD バックグラウンドを減らすことができる。またこの反応には、トップクォークの湯川結合（Higgs とクォークとの結合）という重要な情報を含んでいる。

1.1.1.2　Higgs 粒子の崩壊過程

Higgs の崩壊過程は質量に依存している (図 1.5)。以下に各領域での特徴的な崩壊過程を簡単に説明する。

1. $H \to \gamma\gamma$ ($m_H < 150$GeV)
   この質量領域では、実は $b\bar{b}, c\bar{c}, \gamma^+\gamma^-$ が支配的であるが、陽子陽子衝突から引き起こされる QCD

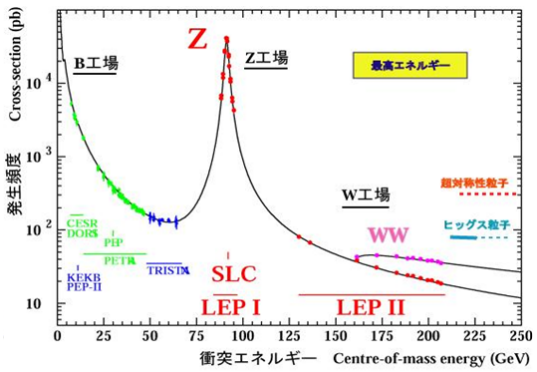


図 1.4　エネルギーと生成頻度 [11]

LEP で 200GeV 程度。

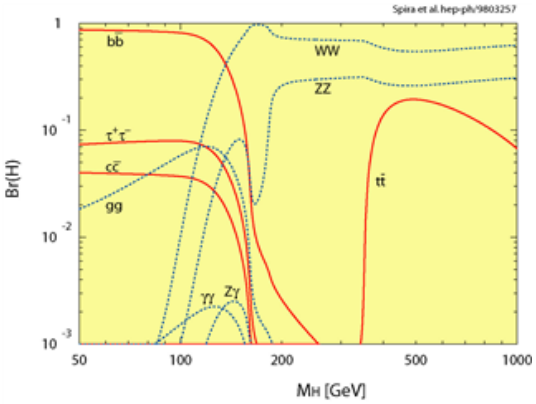


図 1.5　ヒッグス粒子の崩壊分岐比 [11]

横軸がヒッグス粒子の質量。縦軸が崩壊分岐比。崩壊分岐比が大きいイベントでもバックグラウンドが多いと発見が難しい。

ジェットバックグラウンドと区別することが難しい。そこで希崩壊ではあるが $H \to \gamma\gamma$ を観測し、不変質量 $M_{\gamma\gamma}$ 分布を求めると、Higgs 粒子の質量が鋭いピークとして存在する。エネルギー及び角度分解能の優れた電磁カロリメータが必要となる。

2. $H \to \tau\tau$ ($m_H < 150$GeV)
   Higgs 粒子が軽い場合、発見に有効とされているのがこのチャネルである。$\gamma\gamma$ よりも崩壊確率が高く、W/Z fusion の生成過程を考えることでバックグラウンドと区別することができる。この場合、Higgs のピークはバックグランドである Z のピークのテールに現れる。τ の崩壊にはニュートリノが含まれるので E_T^{miss} の精度が重要になる。
3. $H \to ZZ^* \to 4l^{\pm}$ (120GeV $\sim$ 180GeV)
   このモードは、最も綺麗なピークが得られるモードの一つである。一つのレプトン対に対しては、不変質量 m_Z に等しいという条件を課すことが出来るが、Z^* が仮想粒子であるため、もう一方のレプトン対の不変質量には制限が無い。そのため、検出器には運動量、エネルギーに対する高い分解能が求められる。バックグラウンドとしては、ZZ^*、$Z\gamma^*$、$t\bar{t}$、$Zb\bar{b}$ がある。このうち ZZ^*、$Z\gamma^*$ は減らすことは出来ないが、生成断面積もそれほど大きくない。$t\bar{t}$、$Zb\bar{b}$ はそれぞれレプトン対が、Z 起源または Z^* 起源であるという条件をつけることによって取り除くことが出来る。
4. $H \to ZZ \to 4l^{\pm}$ (180GeV $\sim$ 800GeV)
   このモードが最も綺麗なピークを得られる。2 組のレプトン対の不変質量が共に m_Z に等しいという条件を課すことが出来るため、信頼性の高いモードである。ただし、Higgs 粒子の質量が大きくなるにつれ崩壊幅が急激に大きくなるため、有効性が落ちる。
5. $H \to ZZ \to ll\nu\nu$ (400GeV $\sim$)
   この領域では、このモードの方が上 (4) よりも分岐比が約 6 倍も高い。$\nu\nu$ の不変質量は再構成することは出来ないが、これに起因する消失横方向エネルギー E_T^{miss} を精密に測定することが必要になる。
6. $H \to WW \to l\nu jj, H \to ZZ \to lljj$ (600GeV $\sim$)
   この領域ではこれらのモードが上 (4) に比べて、$l\nu jj$ の方は約 150 倍、$lljj$ は約 20 倍の分岐比を持つ。これらのモードでは、バックグラウンドと区別するために Higgs 粒子が W/Z 融合過程によって生

成された場合を考える。この過程では、散乱角前方にクォークによる 2 つのジェットが特徴的で、このジェットを指標とすることでバックグラウンドを排除することが出来る。

1.1.2 超対称性

超対称性の発見は、TeV 領域の新しい物理を拓き、図 1.7 にあるように 3 つの力の統一の可能性を示唆する。これは重力まで含めた統一理論への大きな一歩となる。

この超対称性は、ボソンとフェルミオンを交換する。つまり通常知られているボソンやフェルミオンに対し、スピンが 1/2 だけ異なりスーパーパートナーと呼ばれる超対称性粒子の存在を予言する。例えば、クォークやレプトン（フェルミオン）のスーパーパートナーとして、スクォーク（$\tilde{b}$）やスレプトン（$\tilde{l}$）（ボソン）があり、グルーオン（ボソン）のスーパーパートナーとして、グルイーノ（ $\tilde{g}$）（フェルミオン）がある。もし、この理論が正しければ、LHC では強い相互作用をするスクォークやグルイーノの対が大量に生成され、超対称性粒子の発見が期待される。

最終的に生成される、最も軽い質量を持つ LSP(LightestSUSYParticle) の候補として最軽量ニュートラリーノ $(\tilde{\chi_1^0})$ が考えられるが、直接観測にはかからない。しかし、E_T^{miss} として現れるので、ジェットと共に E_T^{miss} を指標として探索を行う。主な崩壊としては以下の三つがある。

1. Multijets + E_T^{miss} モード
   $\tilde{g} \to q\tilde{q}\tilde{\chi_1^0} \to jets + E_T^{miss}$
   $\tilde{q} \to q\tilde{\chi_1^0} \to jets + E_T^{miss}$
2. 同符号の 2 レプトン・モード
   $2\tilde{g} \to 2(q\tilde{q}\tilde{\chi_i^{\pm}}) \to 2(q\tilde{q}\tilde{W}^{\pm}\tilde{\chi_1^0}) \to 2(jets + l^{\pm} + E_T^{miss})$
3. 3 レプトン・モード
   $\tilde{\chi_1^{\pm}}\tilde{\chi_2^0} \to l\nu\tilde{\chi_1^0} + ll\tilde{\chi_1^0} \to 3l + E_T^{miss}$

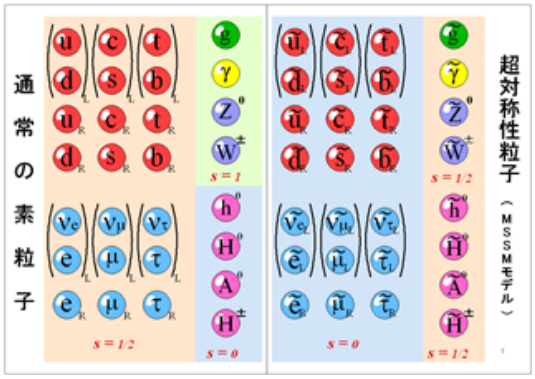


図 1.6　超対称性粒子と通常の粒子 [11]

左が通常の粒子、右が超対称性粒子。超対称性粒子とヒッグス粒子はまだ見つかっていない。

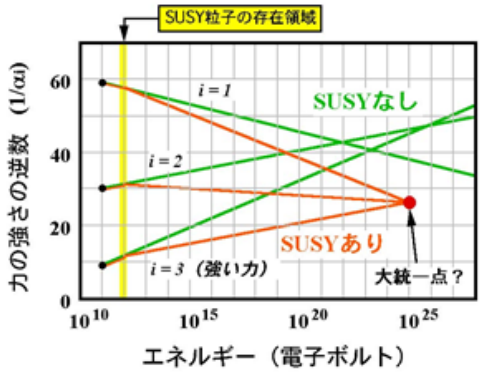


図 1.7　力の統一の可能性 [11]

横軸がエネルギースケール。縦が弱い力、強い力、電磁気力の結合定数の逆数となっている。超対称性が存在しない場合を緑、する場合を赤線で示している。存在する場合エネルギースケールが 10^{25} 辺りで一致することがわかる。

これらについての成果を主な目標として LHC 計画が始まった。

1.2 LHC 計画

LHC はスイス、ジュネーブ郊外にある研究機構 CERN の地下 100m にある大型陽子陽子衝突型加速器である。2000 年まで稼動していた、電子陽電子衝突型加速器 LEP と同じトンネル内に LHC は建設されている。図 1.8 に LHC の概要を載せる。

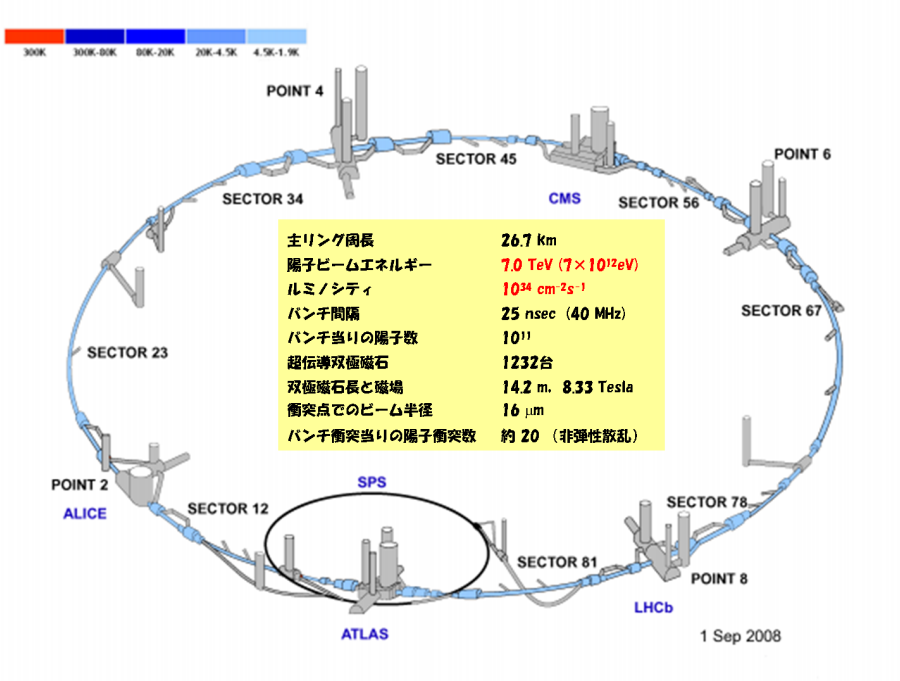


図 1.8　LHC の各種パラメータ

周長は山手線ぐらい。ビームエネルギーは現時点で世界最高となっている。バンチ間隔は 40MHz で、これにあわせてエレクトロニクスのクロックも 25nsec となっている。

LHC は重心系エネルギー 14TeV という現時点で世界最高のエネルギーで稼動する予定である。陽子同士を衝突させるハドロンコライダーになっており、ルミノシティも高いので、バックグラウンドも膨大になり、必要なデータを効率よく収集するすることが必要となる。陽子はメインリングまでに PS (Proton Synchrotron) と SPS (Super Proton Synchrotron) によって 450GeV に加速されてから、LHC のメインリングで 7TeV まで加速される。バンチ間隔は 40MHz で、エレクトロニクスもそれにあわせ、25nsec のクロックを使用する。

LHC には 4 つの衝突点が存在する。後述する大型汎用検出器 ATLAS (A Toroidal LHC AppratuS)、ATLAS より小型の汎用検出器である CMS (the Compact Muon Solenoid)、重イオン衝突実験用検出器の ALICE (A Large Ion Collider Experiment)、B-Physics に特化した検出器 LHC-b が設置される (図 1.9)。

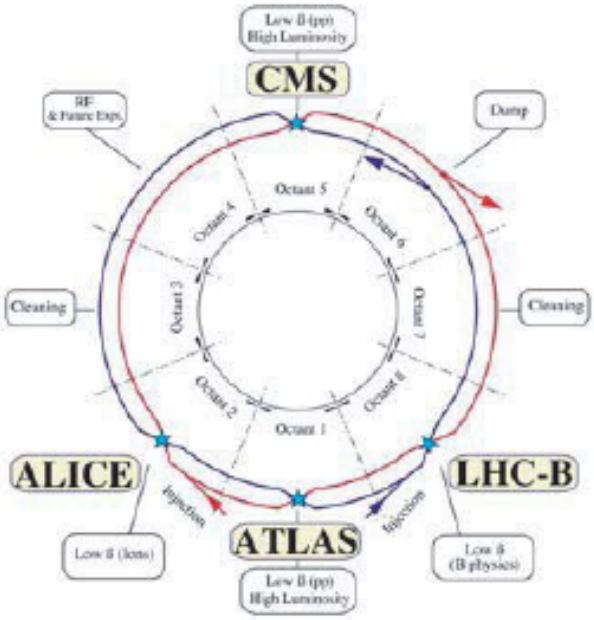


図 1.9　LHC における衝突点 [10]

衝突点のそれぞれに ATLAS、CMS、ALICE、LHC-B が配置されている。

2　ATLAS 実験

この章は LHC の衝突点の一つに設置されている ATLAS 検出器を用いた実験について概要を述べる。ATLAS 検出器は汎用検出器として、様々な物理に対応でき、興味のあるイベントを全て見るように設計されている。

2.1　ATLAS 実験の目的

第一章でも述べたとおり、Higgs 粒子は未だ未発見であるので、まずその発見が確実に期待されている。また、標準模型を超えた超対称性理論に関する物理成果も期待されている。

ATLAS 検出器は質量 100GeV から 1TeV の広範囲で Higgs 粒子を探索する能力がある (図 2.1 - 2.3)。

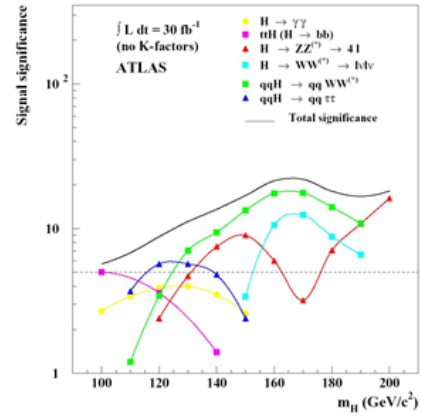


図 2.1　ヒッグス粒子の 30fb^{-1} での発見能力 ヒッグスの質量 100-200GeV [11]

横軸はヒッグス粒子の質量。縦軸は確実性。LHC の三年間のランで 30fb^{-1} のデータを得ることが可能。その時の ATLAS の発見能力。

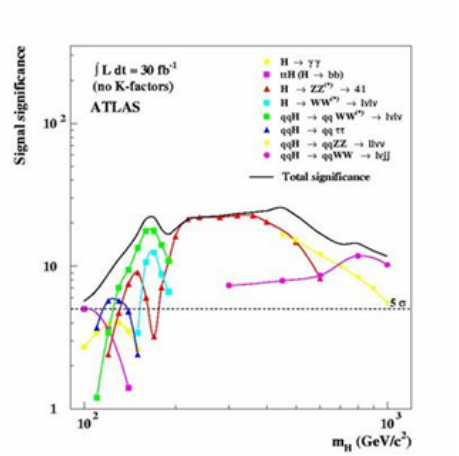


図 2.2　ヒッグス粒子の 30fb^{-1} での発見能力 ヒッグスの質量が大きい場合 [11]

横軸はヒッグス粒子の質量。縦軸は確実性。LHC の三年間のランで 30fb^{-1} のデータを得ることが可能。その時の ATLAS の発見能力。

2.2　ATLAS 検出器の構成

ATLAS 検出器の全体像が図 2.4 である。全長 44m、直径 25m、総重量 7000 トンという非常に巨大な検出器である。内側から内部飛跡検出器、カロリーメータ、ミューオンスペクトロメータという構成で、検出器の間にマグネットが配置されている。ATLAS は円筒形の筒に対応する部分 ($|\eta| < 1.05$) のバレル部分と円筒のフタに相当する部分 ($1.05 < |\eta| < 2.4$) の領域に分けられる。

座標系は図 2.4 に載っているとおりで、ビーム軸と粒子の相対位置は擬ラピディティ (Pseudorapidity)

$$\eta = -\log(\tan\frac{\theta}{2})$$

を用いることが多い。

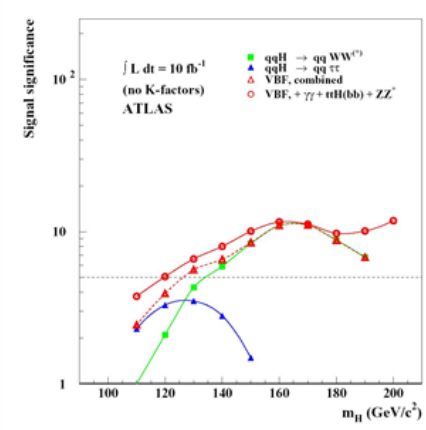


図 2.3　ヒッグス粒子の 10fb^{-1} での発見能力 質量 100-200GeV [11]

横軸はヒッグス粒子の質量。縦軸は確実性。LHC の一年間のランで 10fb^{-1} のデータを得ることが可能。その時の ATLAS の発見能力。

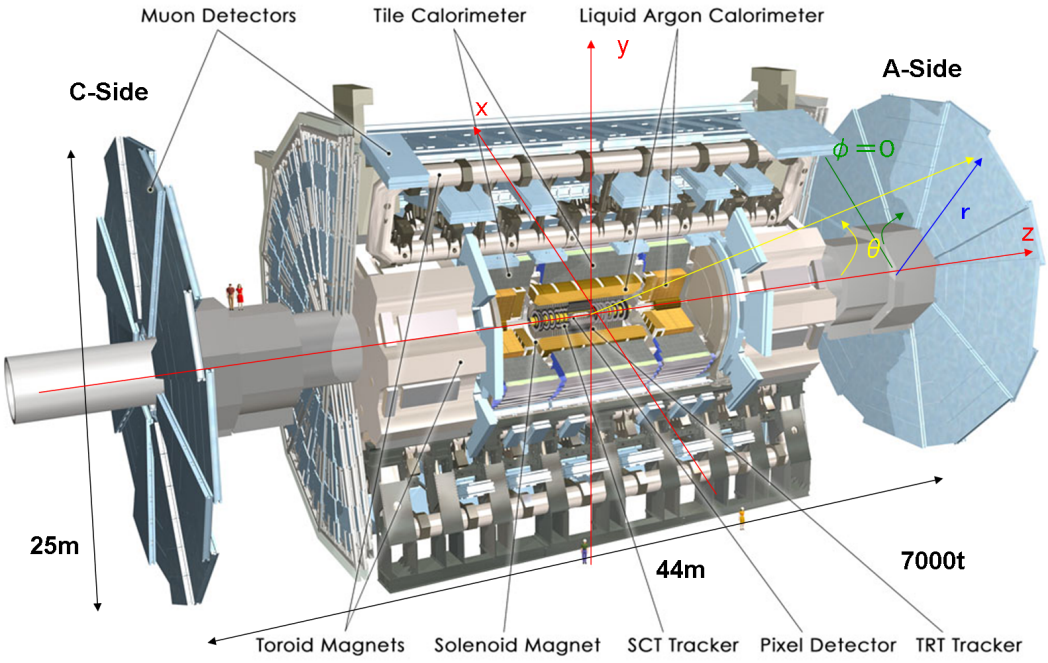


図 2.4　ATLAS 検出器 [13]

内部飛跡検出器、カロリーメータ、マグネット、ミューオンスペクトロメータからなる。座標系は図の通りで、C-Side から A-Side に向かいビーム軸に沿って Z 軸をとり、右手系の座標を取る。

2.2.1 **内部飛跡検出器** (Inner Trackler)

内部飛跡検出器は衝突点から最も近い、2T の磁場を作る超伝導ソレノイドの内部に設置されて、内側から順にピクセル検出器 (Pixel)、シリコントラッカー (SCT)、遷移輻射トラッカー (TRT) の三つで構成されている (図 2.5)。

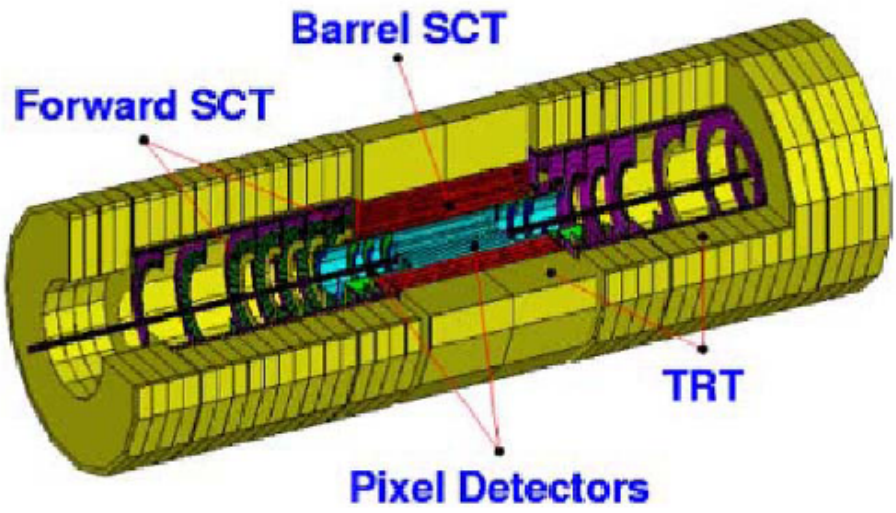


図 2.5　内部飛跡検出器 [2]

内側からピクセル検出器、SCT、TRT から構成されている。

ピクセル検出器は、最内層にある半導体検出器である。これは 1 つの要素が 50 μ m × 300 μ m の高分解能の半導体検出器である。この検出器の精度によって、バーテックスの精度が決められる。SCT はマイクロストリップと呼ばれる細長い有感領域をシリコン上に施した半導体検出器である。TRT は、半径 4mm のストローチューブ検出器で、トラッキングの他に遷移輻射を利用した電子の同定も行う。これらの検出器はいずれも非常に厳しい放射線下に置かれるので、高い放射線耐性が必要である。

2.2.2 **カロリーメータ**

カロリメータの主な役割は、電子や γ 線、ジェットなどのエネルギー、角度の測定である。ATLAS 実験に使用される 4 種類のカロリメータは、電磁カロリメータとハドロンカロリメータの 2 つのカテゴリーに分けられ、広い $|\eta|$ 領域をカバーする。

2.2.3 **マグネット**

ATLAS のマグネットは中央にソレノイドとバレル、エンドキャップにトロイダルの超伝導磁石が配置されている。トロイダル磁石は積分磁場強度としてバレル部分で 2〜6Tm、エンドキャップ部分で 4〜8Tm となる。トロイダル磁場はφ方向が主だが R 方向も存在してしまう。

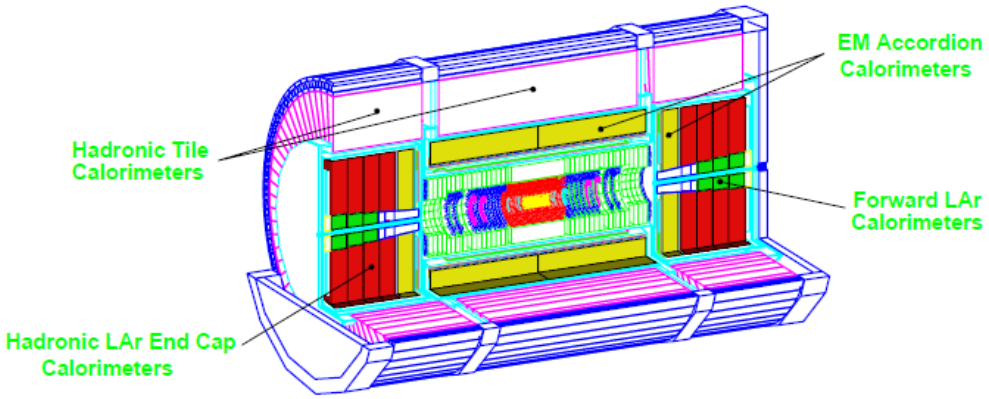


図 2.6　カロリーメータ [2]

電磁カロリーメータとハドロンカロリーメータから成る。電磁カロリーメータは鉛吸収体と液体アルゴンの組からなり、ハドロンカロリメータは鉄の吸収体とシンチレータ、銅の吸収体と液体アルゴン、そして銅とタングステンの吸収体と液体アルゴンの組から成り立つ。

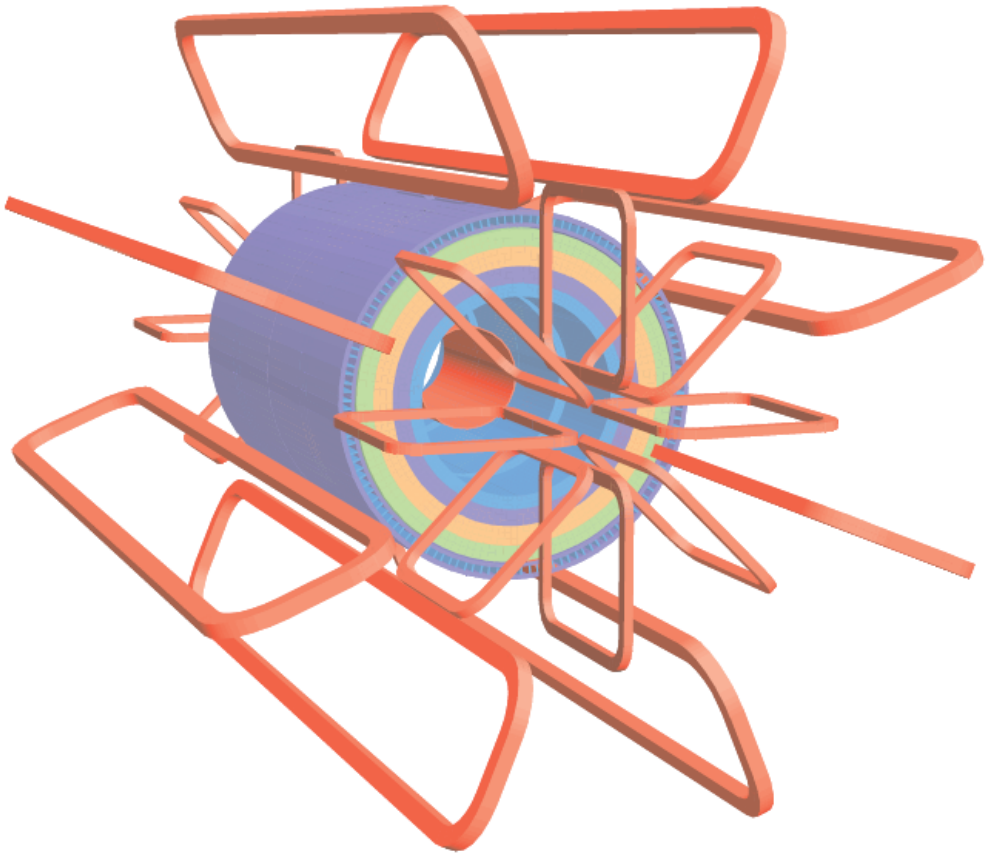


図 2.7　ATLAS マグネットシステム

中央にソレノイド磁石があり、バレルとエンドキャップ部にトロイダル磁石を配置している。

2.2.4 ミューオンスペクトロメータ

ミューオンスペクトロメータは Tracking 目的 (MDT、CSC) と Trigger 目的 (TGC、RPC) とに特化した検出器を用いる。空芯トロイド磁石により、ミューオンシステム独自で運動量が求めることができる ($\Delta P_T/P_T < 10^{-4}$Gev)。それにより運動量選択による Level1 トリガーが可能となる。

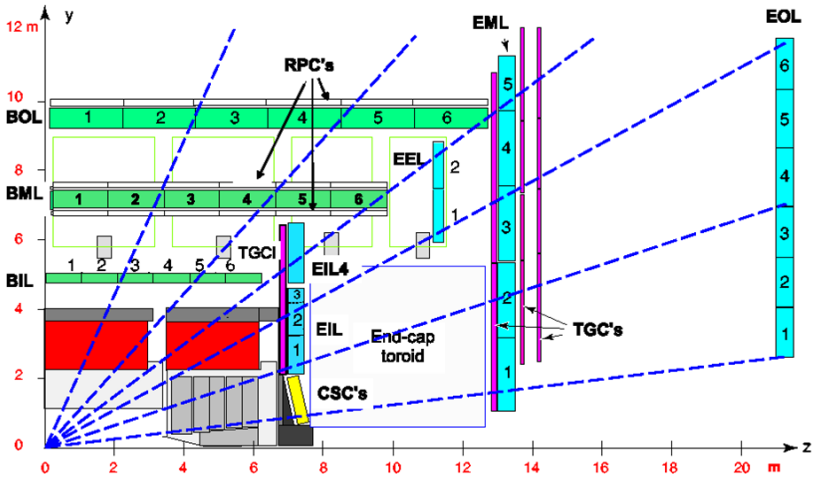


図 2.8　MuonSpectrometer Y-Z 平面

トラッキング目的の MDT が緑と青で、CSC が黄色で示されている。トリガ目的の TGC は紫、RPC は灰色となっている。

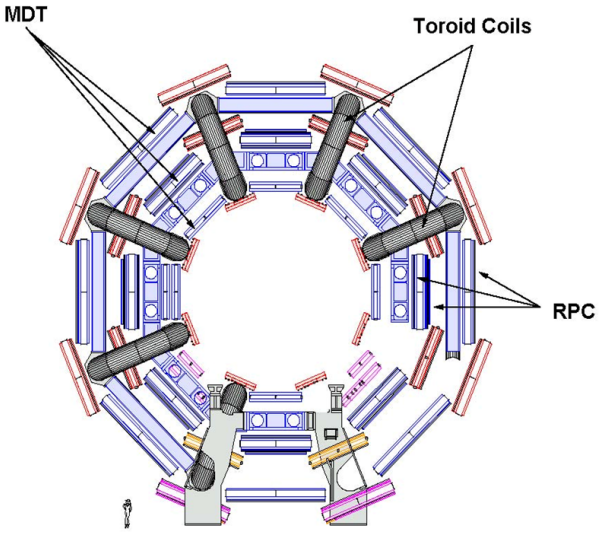


図 2.9　MuonSpectrometer Barrel 部分

RPC と MDT とトロイダルから成る。

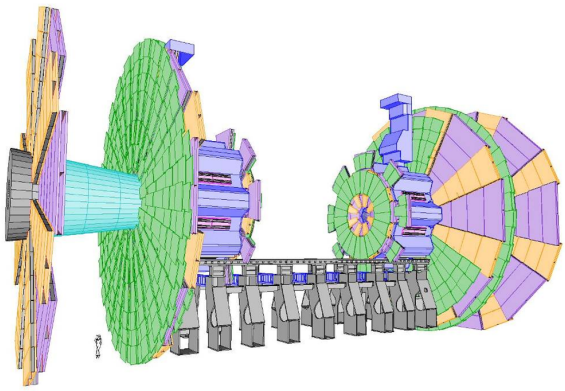


図 2.10　MuonSpectromter Endcap 部分

TGC と MDT とトロイダルから成る。

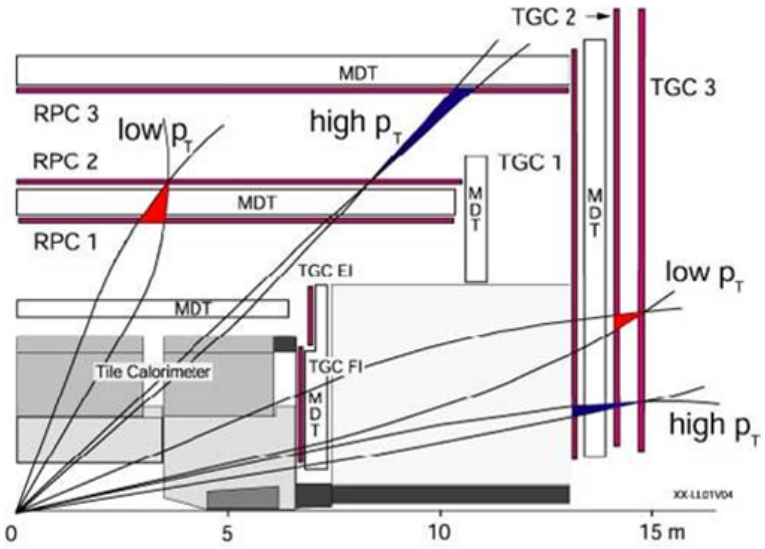


図 2.11　MuonSpectrometer Pt 判別

基準の層から無限大運動量トラックを想定し、それと他の層とのチャンネル差の大きさを見て、運動量を判別する。

2.2.4.1　Resistive Plate Chambers (RPC)

バレル部分を担当するトリガーチェンバである。686 枚のチェンバを用い、トータル 360k チャンネルある。2+2+2 の計 6 層あり、真ん中二層を中心にして、磁場による曲がり具合を見てやり、運動量を判定、トリガー信号を出す (図 2.11)。

チェンバは、2mm の resistive plate に 2mm のガスギャップがあり、二次元 ($\eta \times \phi$) の Strip 読み出しを行う。ガスは ($C_2H_2F_4$: iso-C_4H_{10} : SF_6) が (94.7 : 5 : 0.3) で、混合したものを用いて HighVoltage は 9400V から、大気圧に応じて自動的に変化するようになっている。Timejitter は約 15nsec で、位置分解能は 1cm となっている。

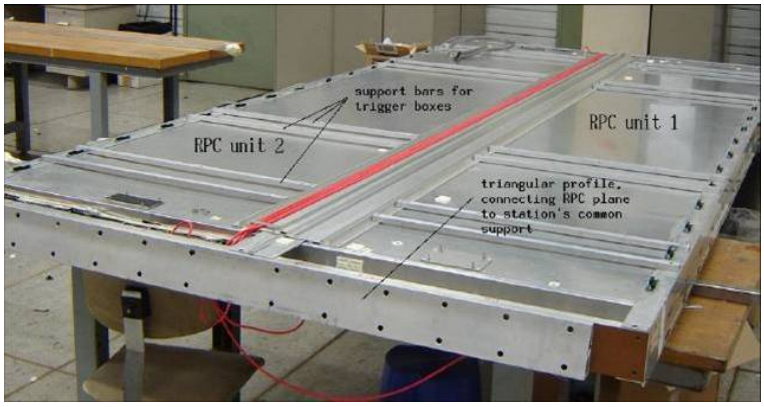


図 2.12　RPC の写真

ストリップが直行しており、二次元読み出しが可能。

2.2.4.2 Monitored Drift Tube chamber (MDT)

トラッキング目的のドリフトチューブチェンバであり、相対位置の変化、自分自身の変形をモニタし、トラックリコンストラクションに反映する (図 2.14)。バレル、エンドキャップの両方に設置されており、バレルの最も内側を BI、次に BM、最も外側を BO、エンドキャップも内側から EI、EM、EO と呼んでおり、それぞれ Large と Small タイプのチェンバがある。

全体で 640 の tracking station、340k チャンネルあり、チェンバはアルミニウムによる筒で直径 3cm、中心にある Wire が 50 マイクロメートルである (図 2.13)。ガスは (Ar : CO_2) が 3atm で (93 : 7) が用いられる。HighVoltage は 3080V、最大ドリフト時間は 700nsec で、Δp はチューブにつき 80 マイクロメートルである。

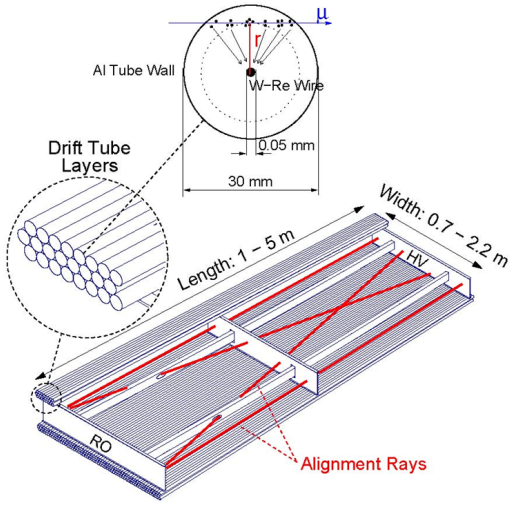


図 2.13　MDT Chamber

直径 30mm のドリフトチューブをフレームに固定した構造をしている。

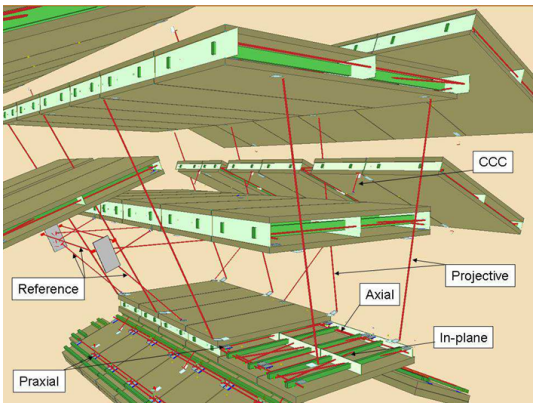


図 2.14　MDT Position

相対位置の変化、自分自身の変形をモニタし、リコンストラクションに反映する。

2.2.4.3 Thin Gap Chamber (TGC)

エンドキャップ部分のトリガーを担当する (図 2.16)。graphite Cathode の MWPC(Multi Wire Proportional Chamber) で、Wire、Strip による二次元読み出し ($\eta \times \phi$) を行う。次章で詳細を述べる。

2.2.4.4 Cathod Strip Chambers (CSC)

CSC はカソードストリップ読み出しの MWPC(図 2.17) で、放射線の多い高ラピディティ領域に配置されるトラキング目的用の検出器である。Wire 間隔が 2.54mm、Strip 間隔が 5.08mm で、ドリフト時間が 30nsec 以下、位置分解能は 60 μm、チャンネル数は 67k である。

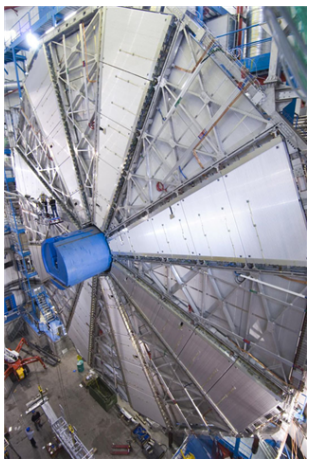


図 2.15　MDT の写真

Large タイプと Small タイプが
φ方向で交互に配置されている。

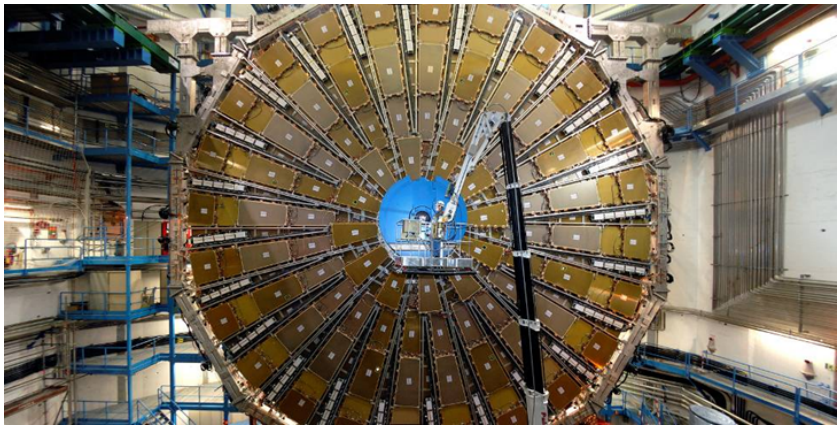


図 2.16　TGC の BigWheel の写真

台形状のチェンバを並べ、円形としてエンドキャップ部分を覆っている。

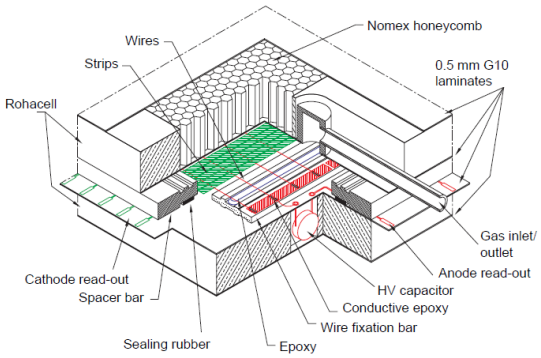


図 2.17　CSC の構造

カソードストリップ読み出しの MWPC。

2.3　ATLAS トリガ、読み出しシステム

LHC は 40.08MHz のビーム衝突ごとに最高で平均 23 個の陽子衝突が起こり、イベントレートは約 1GHz になる。1 イベントのデータ量は ~1.5MByte と見積もられているのでで、全てを記録するには一秒間に $1 \times 10^9 \times 1.5 \times 10^6 = 1.5 \times 10^{15}$Byte、つまり、1.5PByte を記憶しなければならない。それは不可能なので、膨大なバックグラウンドから重要な物理イベントだけを選び出し、記録することが必要となる。ATLAS は三段階のトリガーによってレートを下げ、最終的に 200Hz に落とされる。

2.3.1 概要

ATLAS のトリガは三段階でそれぞれ Level1(40MHz → 100kHz)、Level2(100kHz → 2kHz)、EventFilter(2kHz → 200Hz) と呼ばれている。Level1 はトリガによるデータ収集完了までの時間 (latency) が 2.5μsec という要求があるため、ハードウェアのみによって行われる。Level1 以降はソフトウェアによって行われ、HLT(High Level Trigger) とも呼ばれている。

Level2 はカロリーメータとミューオン検出器、内部飛跡検出器の情報をを用いるが、latency が 10ms と短めなので、この段階では ROI(Region Of Interest) という Level1 をクリアした粒子の大まかな位置情報を使用する。

EventFilter は全検出器の完全な情報を使ってトリガ判定を行う。

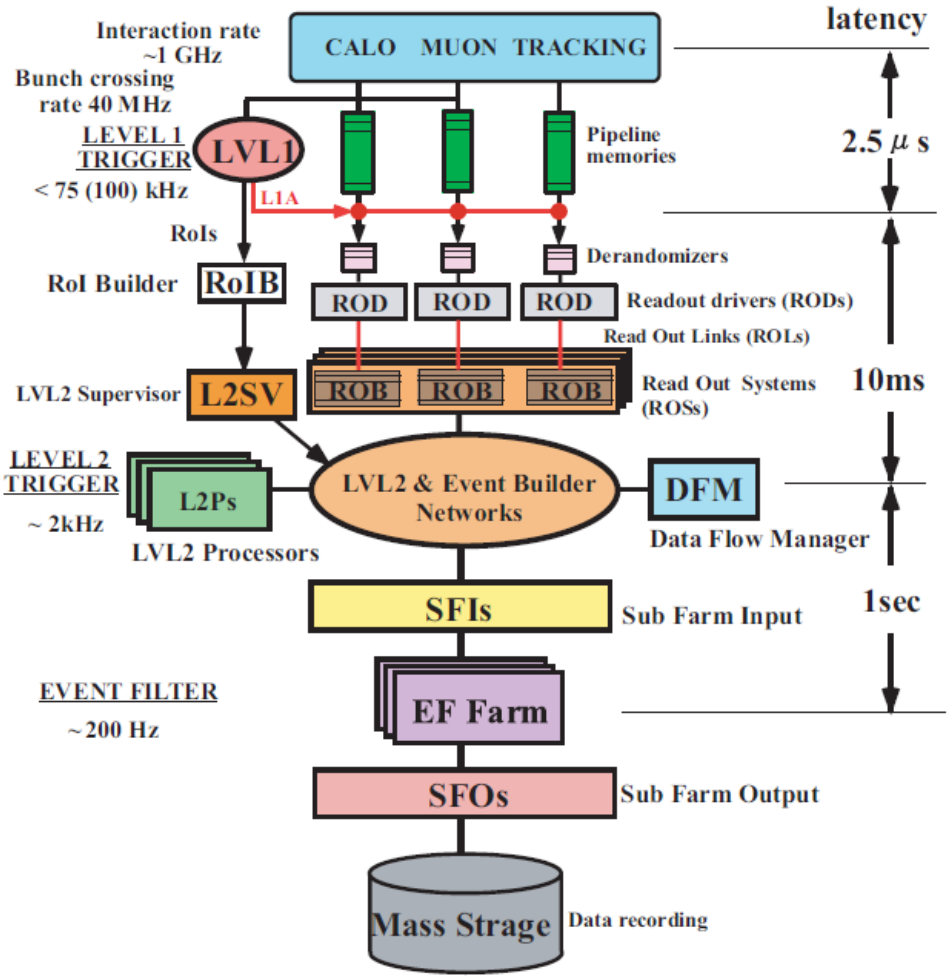


図 2.18　ATLAS Trigger & DAQ システム

ATLAS のトリガーと読み出しの全体図。Level1、Level2、EventFileter の三段階ある。Level1 はハードウェア、他二つはソフトウェアによるものである。

図 2.18 に ATLAS のトリガと読み出しの全体像を載せる。検出器からの信号は Level1 バッファ (L1 バッ

ファ) というパイプラインメモリに保存される。そしてトリガ検出器[*1]により、Level1 トリガーの判定がなされたら L1A(Level1Accept) 信号が発行され、それを受けたら L1 バッファから信号をデランダマイザ (Derandomizer)[*2]に送り、タイミングを調整、バンチと L1A の識別のための ID(BCID と L1ID) を付けられたデータを ROD(Read Out Driver) に送る。ROD は整合性を確認した後、S-Link(Simple Link Interface) という規格で、ROL(Read Out Link) を通して、ROB(Read Out Buffer) へと送る。ROB をまとめたものを ROS(Read Out System) と呼ぶ。ROB は Level2 判定が行われるまでデータを保持し、ROB から送られた信号はイベントビルダーで EventFilter のトリガー判定を待つ。それらをクリアーしたデータがストレージに保存される。

2.3.2 Level1 トリガー

本論文に特に関係する Level1 トリガについて詳細を述べる。Level1 トリガはカロリーメータ、ミューオン検出器 (TGC、RPC)、MUCTPI(Muon Trigger to CTP Interface)、CTP(Central Trigger Processer)、TTC(Timing Trigger and Control distribution system) から構成されている。カロリーメータからは e/γ、E_T^{miss}、τ、Jet のエネルギーなどの情報、ミューオン検出器からは高い p_T のミューオンの情報が CTP に送られる。各トリガー用の検出器からの情報は CTP に集められ、トリガ判定の結果 L1A 信号が出される。この L1A は TTC システムによって各検出器に配られる (図 2.19)。

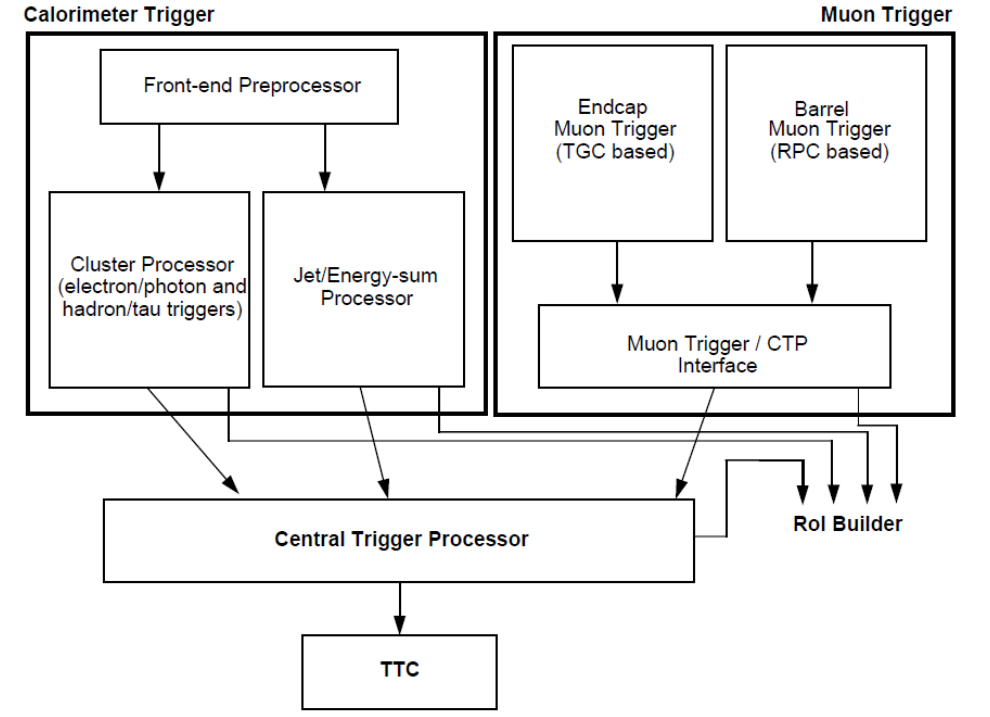


図 2.19　Level1 トリガ処理の流れ

カロリーメータとミューオン情報を CTP に送り、判定、TTC によって検出器に配られる。

[*1] TGC と RPC、カロリーメータ
[*2] ランダムにおこるイベントを一定の間隔で読み出せるようにするメモリ

2.3.2.1 MUCTPI

MUCTPI は TGC と RPC の情報を CTP に渡す役割を担う。。RPC と TGC は各々セクターと呼ばれる単位ごとに p_T の大きなミューオンの候補を挙げて、MUCTPI に送る。MUCTPI はこれらのトラックの候補を受け取り、境界部分での処理[*3]を行ってから、ミューオンの候補についての情報を、CTP へ送る。またその情報は Level2 トリガに送るため、RoIB（RoIBuilder）に渡される。

2.3.2.2 CTP

CTP の役割はカロリメータとミューオンの情報を統合して、最終的な Level1 トリガーの判定を行うことである。カロリメータでは、e/γ 、τ /ハドロン、ジェットのそれぞれに対し、数段階の閾値が設けてあり、同様にミューオン検出器では、ミューオンの p_T について、数段階の閾値が設けてある。CTP は最高 96 種類のトリガー項目を設定出来て、CTP が受け取る閾値を越えた情報とそのトリガー条件とを比較することで、L1A の有無を決定する。トリガー判定が終わると、CTP は TTC システムに対して、L1A とトリガーの情報を送信する。CTP でのレイテンシーは 4 バンチ（100nsec）以下と決められている。

2.3.2.3 TTC

TTC システムは、フロントエンドの各エレクトロニクスの同期をとるために、BC clock や L1A などの信号を分配するシステムである。また、TTC は各検出器固有のテストやキャリブレーション用のコマンドを受信し、実行する役割も担う。表 1 に TTC が扱う主な信号を挙げる。

表 1　TTC で使われる主な信号

BC Clock	Bunch-Crossing signal。40MHz に同期させるためのクロックパルス
L1A	CTP からおくられてくる Level1Accept
BCR	Bunch Counter Reset。BCID のリセットに使用
ECR	Event Counter Reset。L1ID のリセットに使用
EVID(L1ID)	EVent IDentifier。ROD、ROB でのバンチのチェックに使用
BCID	Bunch-Crossing IDentifier。ROD、ROB での Level1ID のチェックに使用

TTC は幾つかの partition に分かれており、その partition は TTCvi、LTP(Local Trigger Processer)、TTCvx、RODbusy というモジュール 4 つを必ず持つ。

LTP は partition の外部からの TTC で扱う信号を全て受信する。LHC からは 40.08MHz の BC クロックと周期 88.924 μ sec の ORBIT 信号を、CTP からは L1A 信号を受信する。L1A と ORBIT は TTCvi へ送信され、クロックは TTCvx へ送信される。

TTCvi は受信した L1A やテスト信号を TTCvx に送信し、TTCvx は受信した情報を加工した後、オ

[*3] TGC と RPC で重なった部分で一つの粒子を二つの候補にしないようにする。

プティカルリンクによってフロントエンドに設置される TTCrx と呼ばれる ASIC（Application Specific Integrated Circuit）まで分配する。TTCvi から TTCvx に渡される信号は、A-Channel、B-Channel という 2 種の信号に分配される。A-Channel で扱われるデータは L1A だけであるが、B-Channel では TTCrx に同期コマンド、非同期コマンドを送付することが出来、前者はテストパルスの発生等に用いられ、後者はパラメーターの設定などに用いることが出来る。TTCrx では、受信した信号をフロントエンドに配置される各エレクトロニクスに分配する。

また、RODbusy モジュールは、partition 内に属する ROD からの busy を集め結果を LTP に渡す。LTP が受け取った busy は CTP に渡される。

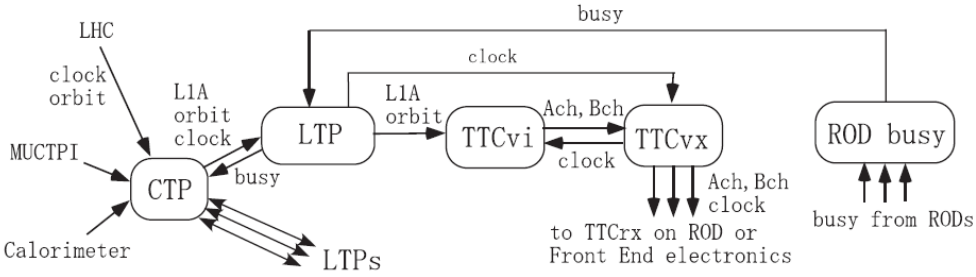


図 2.20　TTC の partition

TTCvi、LTP、TTCvx、RODbusy の 4 つのモジュールを持つ。

3 Thin Gap Chamber (TGC)

この章でエンドキャップ部のトリガーを担当する TGC についての詳細を述べる。

3.1 TGC の概要

TGC は R 方向として Wire、φ方向として Strip による二次元読み出しの台形状の MWPC チェンバである。それらを円形状に配置し 1 ステーションとなり、3+2+2 の計七層と EI および FI とよばれる二層から構成され、エンドキャップトリガを担当する。(図 3.6)。

チェンバはアノードとカソードとの間が 1.4mm と薄く、Wire 間隔は 1.8mm で 6 から 31 本をまとめて 1 チャンネルとして、計 320k チャンネルある。ガスは (CO_2:n-pentan) を (55:45) で混合したものを用いて、HighVoltage が 2900 程度で運用される予定である。Timejitter は 20 から 25nsec で、位置分解能は数十 cm となっている。

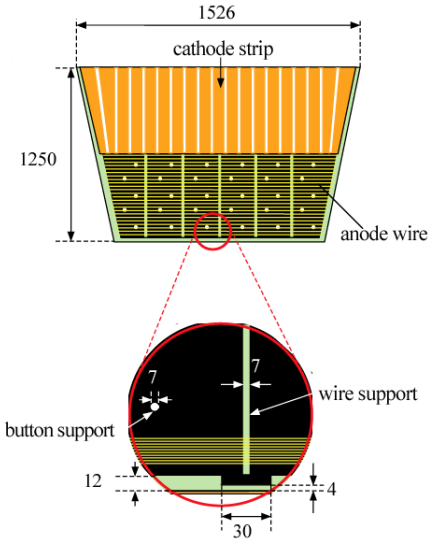


図 3.1 TGC 平面図

銅からなるφ方向の Strip とタングステンワイヤの R 方向の Wire からなる。Wire と Strip が垂直に走り、二次元読み出しが可能

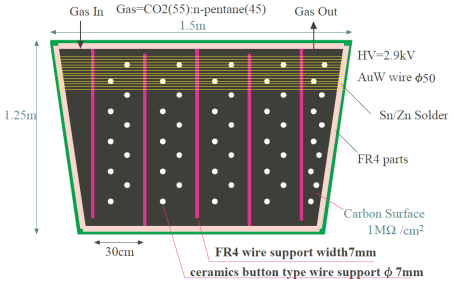


図 3.2 TGC チェンバー内部

ワイヤーサポートはたるみを防ぐだけではなく、ガス流路の形成やチェンバのゆがみも防いでいる。

TGC の内部構造は図 3.4 のようになっており、Wire は 50 μ m で広い範囲で電場を形成、アノード、カソードギャップが狭いので、高頻度入射での出力電圧低下を防いでいる。構造体は FR4 とよばれるガラスエポキシ基盤を用いており、グラウンドを与える高抵抗のカーボンを塗装することにより、反対面に張った Strip に信号が形成、二次元読み出しを行っている。この構造は ATLAS のトリガーチェンバに対する以下の要請にこたえられるように設計された。

- 99% 以上の高い検出効率
- 25nsec 以下の速い反応時間
- 1 C/cm の放射線耐性

- 1 kHz/cm^2 以上の高頻度入射耐性

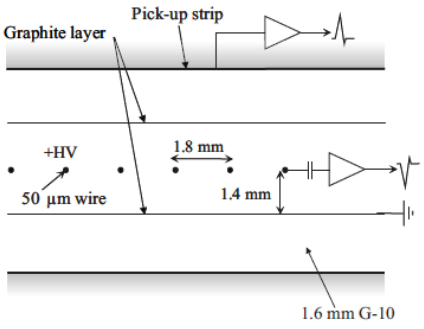


図 3.3　TGC の構造ワイヤ近傍断面図

ワイヤの間隔が 1.8mm、カソード面とワイヤの間隔が 1.4mm となっている。

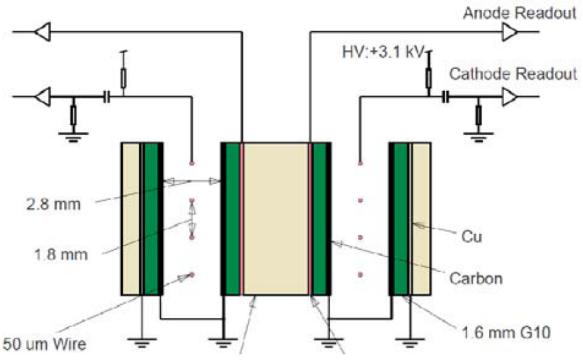


図 3.4　TGC の内部構造 [19]

ワイヤと黒で示したカーボン面、緑のグラファイト、そして Strip の銅が図示されている。

TGC は一層では用いず、図 3.5 に示すように、二層の Doublet と三層の Triplet として用いる。多層にし、コインシデンスをとることによって、バックグラウンドを減らす効果のほか、ワイヤサポートによる不感領域の影響も減らすことが出来る。また、各層でチャンネルが 1/2(Doublet)、1/3(Triplet) でずれて配置されているため、位置分解能が 2 倍、3 倍となる。

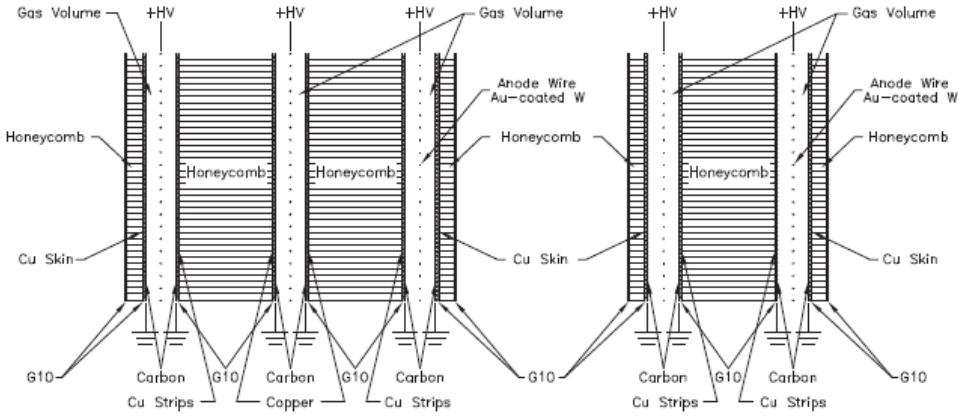


図 3.5　TGC の Triplet(左)Doublet(右) 構造

TGC は Triplet と Doublet という二種類が存在する。Triplet は三層の Wire と二層の Strip。Doublet は二層の Wire と二層の Strip から成る。

TGC の配置は図 3.6 のようになっている。TGC は片サイドに四つのステーションがあり、

最も内側に EI(Doublet) と FI(Doublet)[*4]、数メートル離れて TGC1(Triplet)、TGC2(Doublet)、そして TGC3(Doublet) のように配置されている[*5]。TGC3 は運動量判別でも基本の層となるので Pivot プレーンとも呼ばれている。

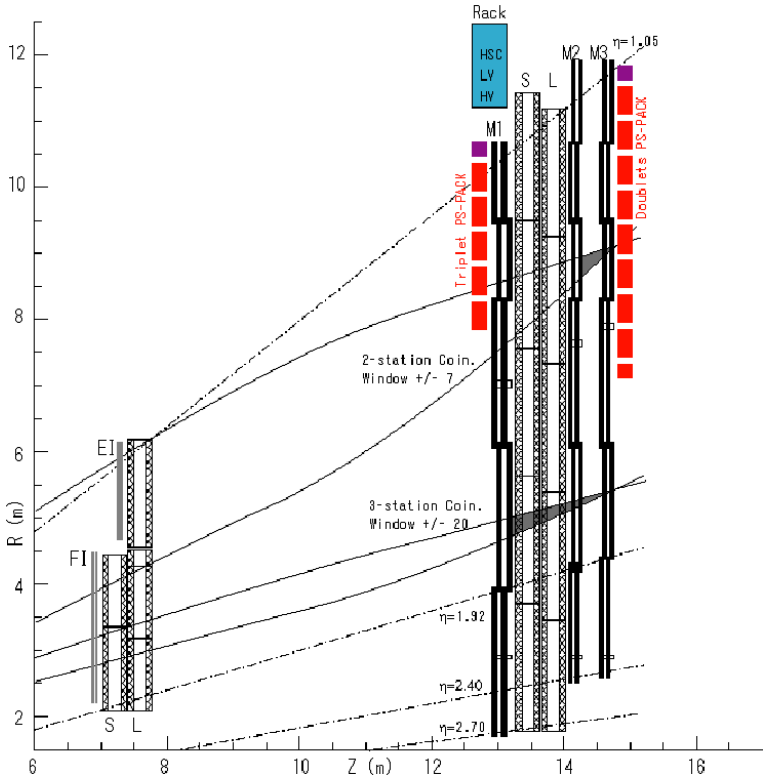


図 3.6　TGC の YZ 平面での配置 [1]

内側から EI と FI、TGC1、TGC2、TGC3 と配置されている。

TGC は R 方向で最も内側のチェンバ 1 枚を Forward と呼び、それよりも外側は Endcap と呼んでいる。Endcap 領域は Doublet の BigWheel ステーションで 5 枚、Triplet で 4 枚、EI で 1 枚となっている。EI に関しては図 3.9 のように、Endcap 部分何枚かがマグネットの配置の関係で削られている。5 枚のチェンバは内側から E5、E4、E3、E2、E1 と数え、Triplet の場合も E4、E3、E2、E1 と数える。

φ方向について チェンバ一枚を単位とすると、TGC の BigWheel は Forward で 24、Endcap は 48 あり、EIFI は Forward で 24、Endcap は 21 ある。

セクタの単位として図 3.8 の赤枠のように 1/12 単位 (Forward:2、Endcap:4) で数える。

[*4] SmallWheel と呼んでいる。EI は Endcap Inner で FI は Foward Inner の略称。他のステーションは BigWheel

[*5] TGC1、TGC2、TGC3 のことを M1、M2、M3 と呼ぶこともある。

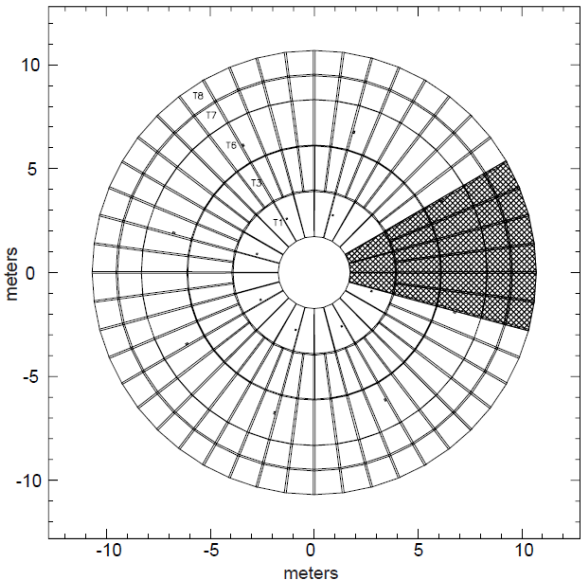


図 3.7　TGC1 の R ×φ平面 [1]

TGC3、TGC2 より Endcap 部分のチェンバが一枚少ない 4 枚で構成されている。

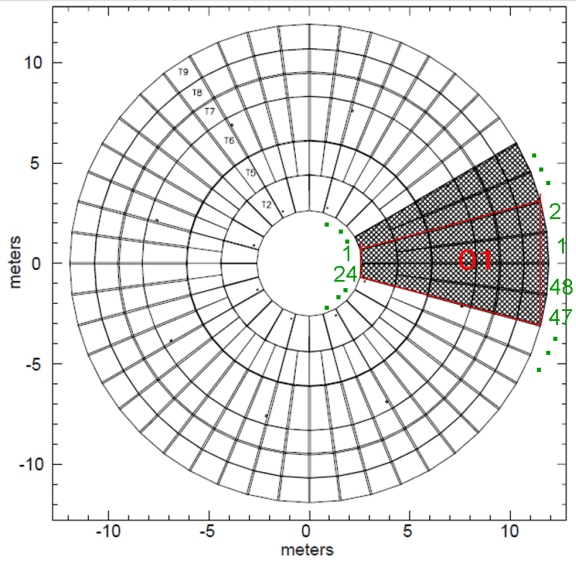


図 3.8　TGC3 の R ×φ平面 [1]

赤枠は例として 1/12 セクターの 01 を囲っており、緑の数字はオフラインで使用する stationPhi の番号。TGC1 も同様の数え方をする。昔は黒の網目部分のように 1/8 でセクターを数えていた。

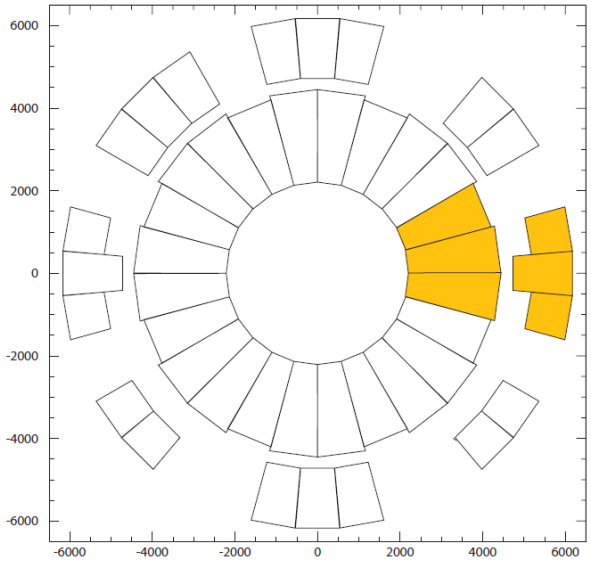


図 3.9　TGCEIFI の R ×φ平面 [1]

内側が FI、外側が EI で、EI はマグネットの配置上削られている部分がある。

TGC は分解能を増やせるように、レイヤーごとに η 単位でずらして配置されている (スタッガリング) (図 3.10)。これにより、Doublet では 1/2ch、Triplet では 1/3ch 単位の位置分解能でトリガーを行うことが出来る。

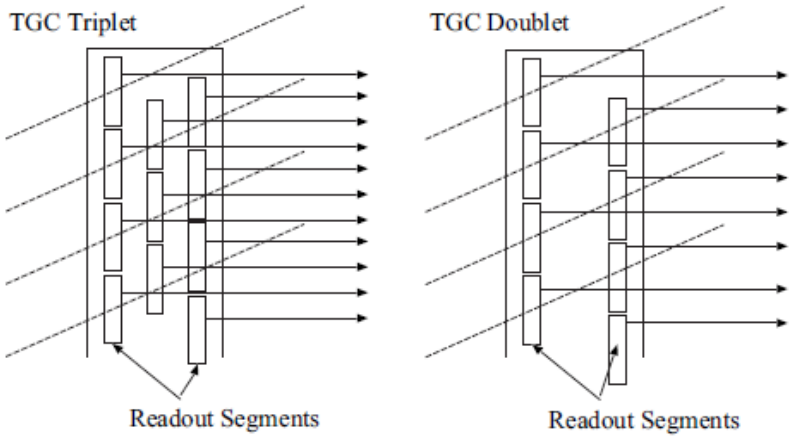


図 3.10　TGC のスタッガリング

レイヤー間でチャンネルをずらして配置されている。擬ラピディティ単位でずらしてある。

図 3.11 にガス増幅の模式図を載せる。ガス中を荷電粒子が通過すると、ガス分子が電離されイオン化する。電離した電子は電場により加速されながら、陽極へと向かう。その電子が電離エネルギーを超えると二次電子を生成する。これらを繰り返すことにより電子なだれを形成し、それらが電子雲として Wire を取り囲む。イオン雲がさらにその周りを取り囲み拡散していく。この電子なだれを Wire から読み取り、同時にカソード面で、カーボン面に電荷が誘起され、外側の Strip にも電荷が誘起され信号が読み出される。

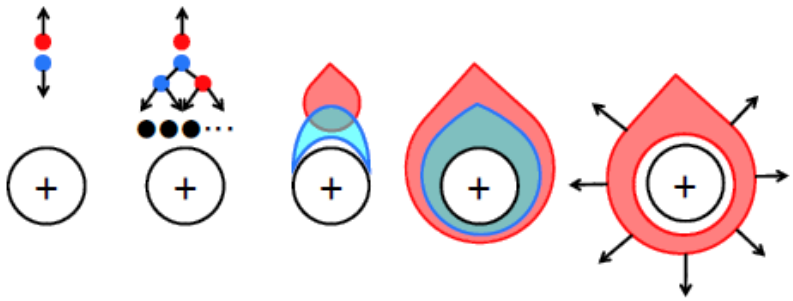


図 3.11　ガス増幅の模式図

赤がイオン 青が電子

TGC のトリガーロジックは TGC3 の Hit を基準に無限大運動量のトラックを仮定する。それと実際の TGC2 や TGC1 上の Hit とのチャンネル差 (これからこの差を Delta と呼ぶ) によって、磁場による曲がり具合を計測、それにより運動量を判別、その大きさに応じてトリガーを発行するというのが基本的な論理である (図 2.11、図 3.6 参照)。

具体的いうと、Wire、Strip で最初に外側四層で 3outof4 のコインシデンスをとり、それをクリアしたトリガ候補の TGC3 の 1/2ch 粒度の位置情報と Delta の大きさ送る。その後、TGC1 とのコインシデンスを取り、それが合えば Highpt フラグの立った Highpt 候補として、それ以外は Lowpt として、8ch 粒度の位置と Delta の大きさを送る。そして最終的に Wire、Strip の R-ϕ コインシデンスがとられ、Delta の大きさから運動量を判別、それらがトリガメニューに合えば、メニューラベルと ROI(Region of Interest) という 8ch × 8ch の大きさの TGC3 の位置情報を CTP へと送り、トリガ発行となる。(図 3.12)。

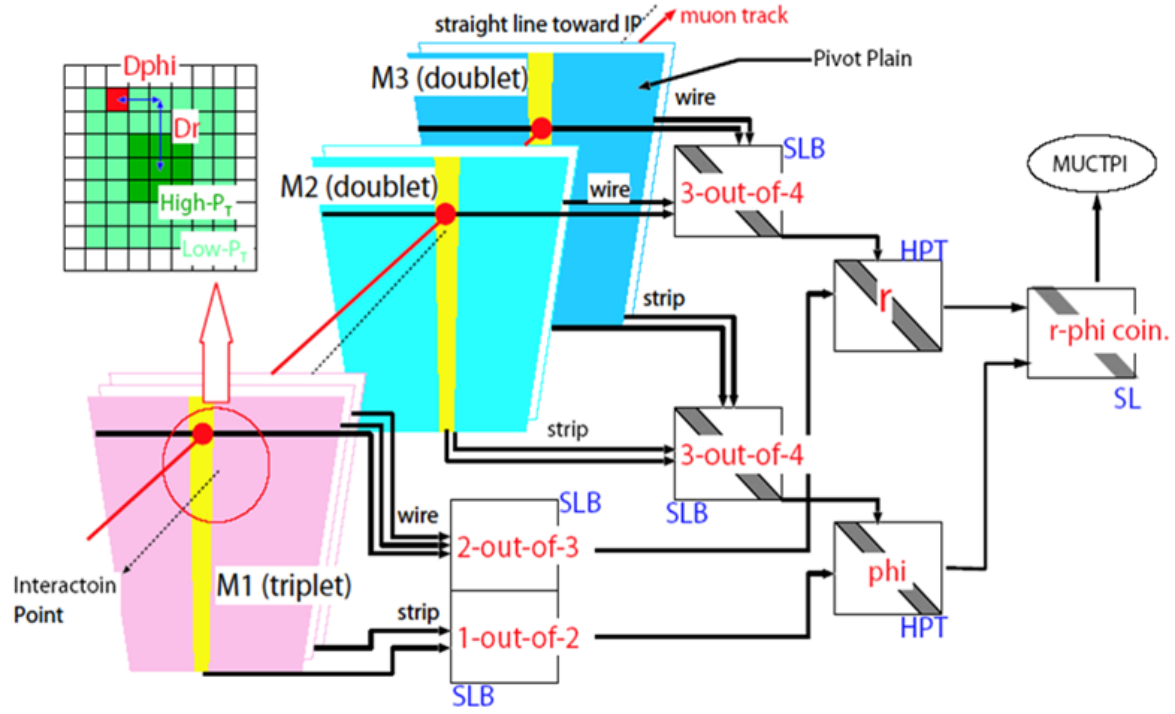


図 3.12　TGC のトリガー

SLB で TGC3 と TGC2 の 4 層のコインシデンスと TGC1 の 3 層のコインシデンスをとる。Highpt でそれらのコインシデンスを取り、SL で R とφのコインシデンスを取り、MUCTPI へと送られる。

3.2 TGC エレクトロニクス

TGC の信号を処理するエレクトロニクスについて述べる。

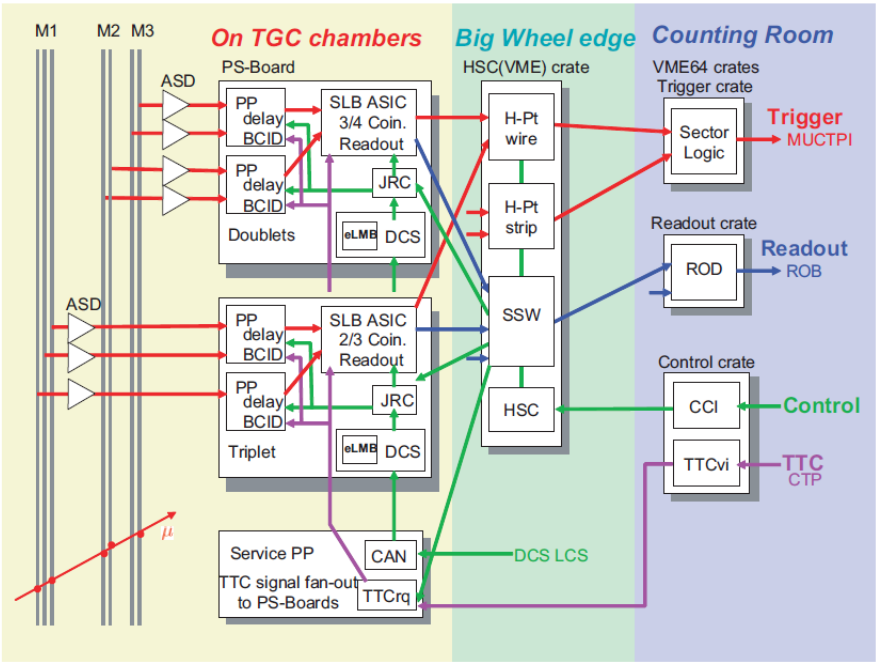


図 3.13　TGC のエレクトロニクス

赤矢印がトリガーライン、青がリードアウトライン、緑がコントロールとなっている。

TGC のエレクトロニクスは図 3.13 のようになっている。チェンバからの信号は ASD(Amplifier-Shaper Discriminator) により、増幅、デジタル化され、その後、PatchPanel によりタイミング調整が行われる。SLB(SLave Board) に入り、トリガーラインとリードアウトラインに仕事が分かれる。

データは SLB バッファにクロック周波数の 25nsec ごとに貯められる。それとは別に、3outof4 のコインシデンスが取られ、Highpt ボードに送られる。そこで TGC2、TGC3 と TGC1 とのコインシデンスをとり、SL ボードへと送り、R-ϕ コインシデンスがとられ、Pt によりラベルをつけトリガ候補を選出、そのラベルと ROI を CTP へと送り、トリガー発行となる (トリガーライン)。

Level1 のトリガー信号は Level1 Accept(L1A) と呼ばれ、それを受けると SLB のバッファは設定した深さののデータを送る。TGC は前後 25nsec 分の三つ (Previous、Current、Next) を送るようになっている。それらのデータは SSW で、圧縮し、ヘッダとトレーラを付け、その後 ROD へと送られる (リードアウトライン)。

3.2.1 Amplifier-Shaper Discriminator Board(ASD)

ASD は信号を増幅、デジタル化し、ある閾値電圧 (Threshold 電圧) を超えた信号だけを出力するエレクトロニクスである。また、エレクトロニクスやタイミングのテストのために擬似信号 (テストパルス) を出力する機能も持っている。1chip で 4 チャンネル分処理し、それらが 4 つで一つのボードとなる (1 ボード 16ch)。

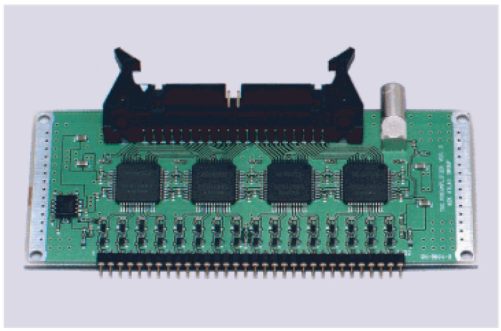


図 3.14　ASD

chip が 4 つあるのが分かる。これ一つで 16 チャンネル。

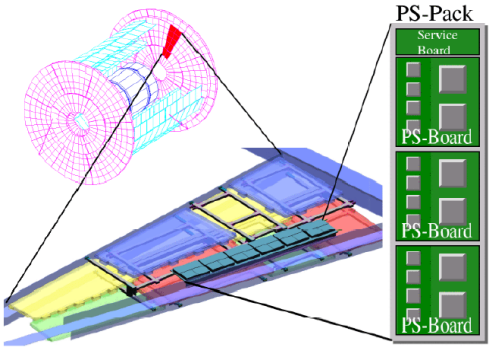


図 3.15　PS-Pack の構成と配置

PS-Borad をまとめて、一番上に ServicePathcPanel を付けて Wheel の脇に設置されている。

3.2.2　PS Board (PSB)

PS ボードは PP ASIC と SLB ASIC の両方がついているボードである。Endcap と Forward または Wire と Strip などの種類に応じて作られており、BigWheel の場合、それらをまとめて PS Pack として Wheel に沿ってセクターごとに設置されている (図 3.15)。EIFI では PS Board は専用のクレートに SSW と共に収められて、検出器の傍に配置されている。

3.2.2.1　Patch Panel (PP)

図 3.16 に PP チップのブロック図を載せる。PP は TOF(Time Of Flight) やケーブルによるタイミングのずれを調整、バンチの識別を行うのが仕事である。

3.2.2.2　SLave Board (SLB)

図 3.17 が Doublet、図 3.18 が Triplet Wire、図 3.19 が Triplet Strip の SLB の主な機能を図にしたものである。他にエレクトロニクスなどをチェックするためのテストパルスを出力する機能などがある。

SLB の仕事は大きく二つある。一つ目はトリガ系の仕事として、Doublet では 3outof4 のコインシデンス、Triplet Wire では 2outof3、Triplet Strip は 1outof2 のコインシデンスをとり、トリガ候補の位置情報や運動量判別に使用される無限大運動量とのチャンネル差 (Delta) などの情報を Highpt ボードへ渡している。

二つ目はリードアウト系の仕事として Level1 バッファに Hit データを貯め、L1A を受け取ったら、前後合わせて三バンチ分のデータをデランザマイザを通して SSW へと送っている。SSW に送るデータは 200bit あり、40bit 分をトリガデータ、他 160bit は Hit 分布のデータとなっている。

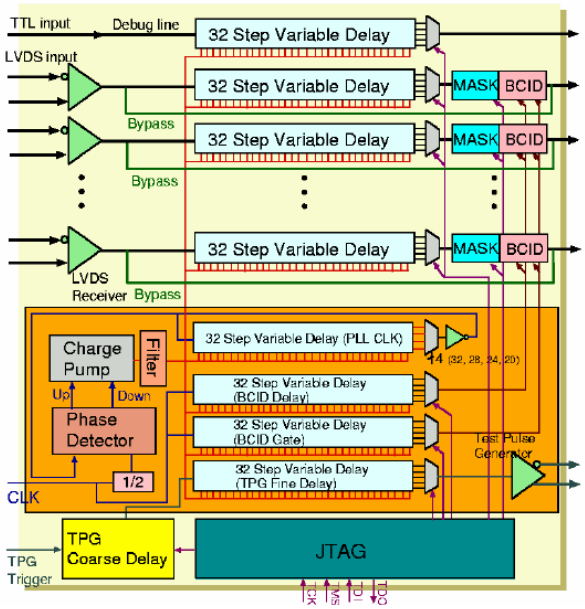


図 3.16　PP ASIC Block Diagram

タイミングのずれ、バンチの識別の機能が載っている。

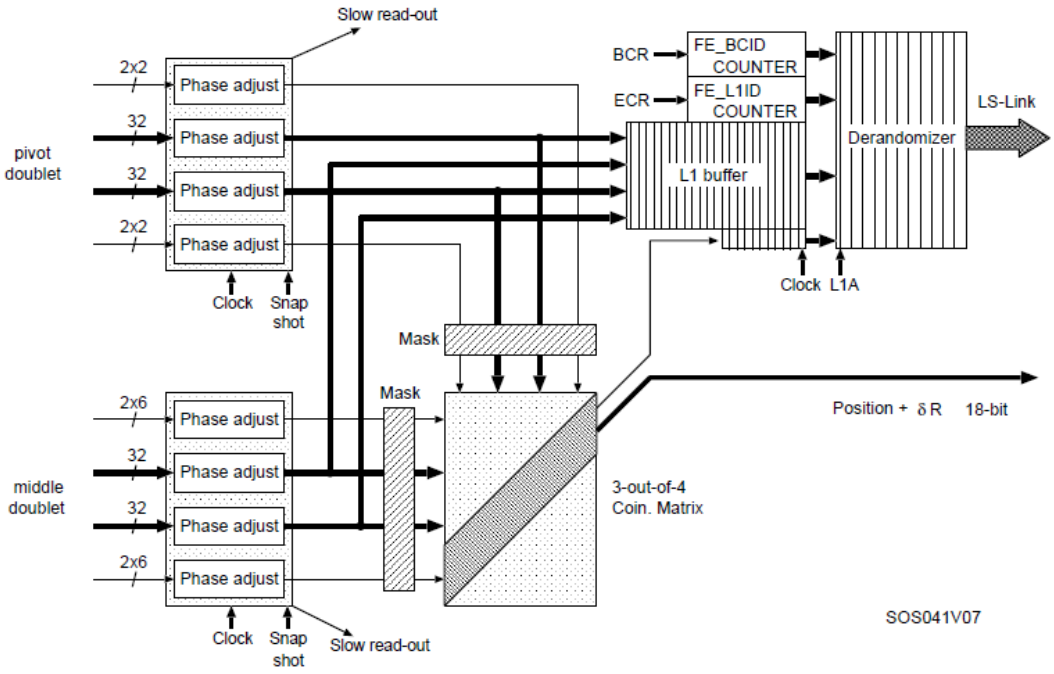


図 3.17　SLB Doublet の機能 [1]

pivot doublet とは TGC3 の事。middle doublet とは TGC2 の事。信号はトリガマトリックスと L1 バッファに送られる。マトリックスは Pivot の場所と無限大運動量トラックからのズレ Delta を送る。L1 バッファは L1A を受け取ったらデランザマイザにデータを送る。その際、BCID と L1ID も付与される。BCID と L1ID はある段階でリセットする必要があるので、ECR や BCR といったリセット信号を受け取る。

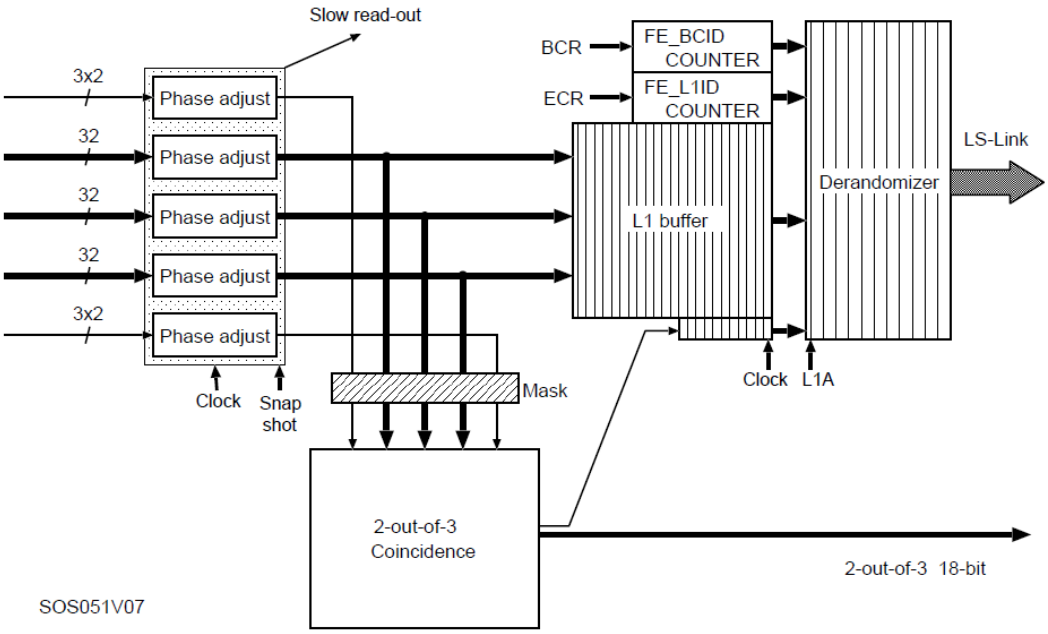


図 3.18　SLB Triplet Wire の機能 [1]

基本は Doublet と同じだが、コインシデンスのとり方が違い、トリガ部の送るデータも位置情報のみである。

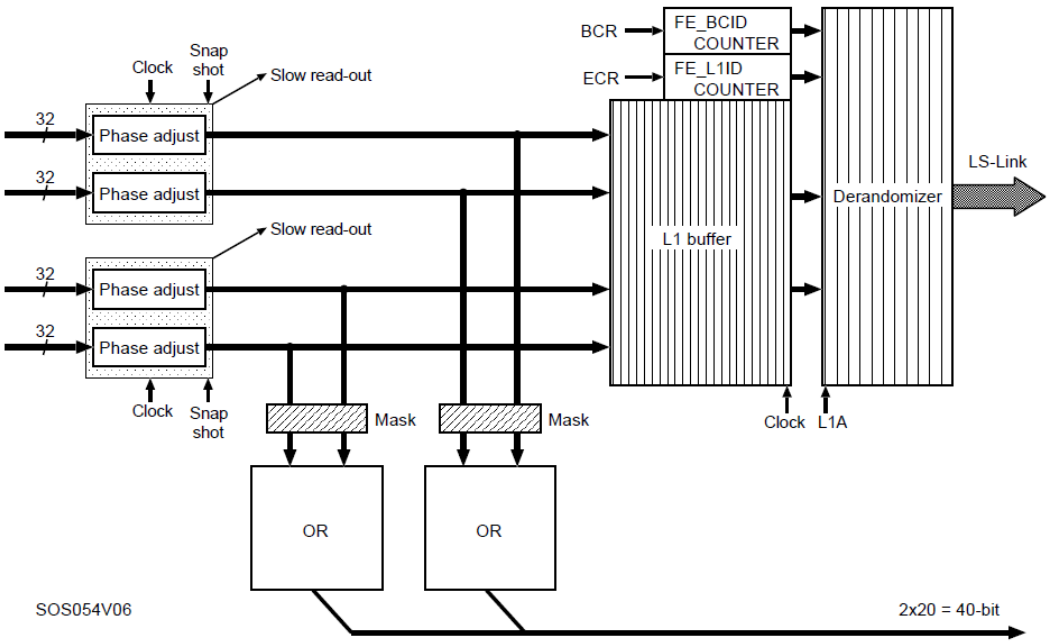


図 3.19　SLB Triplet Strip の機能 [1]

二層しかないのでコインシデンスが OR となっている。トリガ部の送るデータは位置情報のみである。

トリガ系

コインシデンスを取るときの詳細が図 3.20 である。一つのチップで最大二つの候補を選出するようになっており、もし複数候補が存在したら、その中で最も高い運動量の候補二つをを選出する。

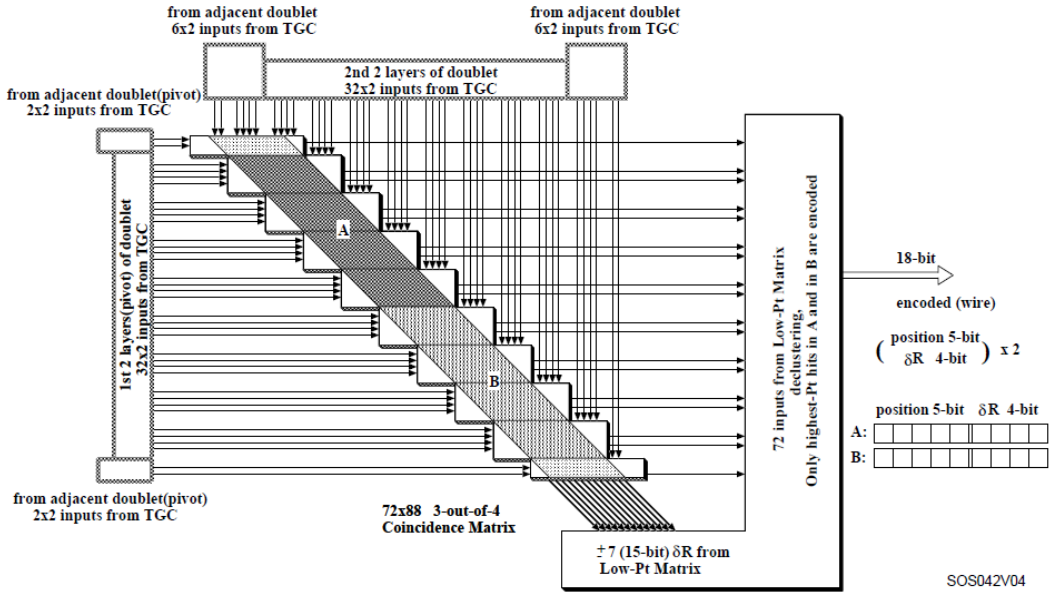


図 3.20　SLB Doublet Wire コインシデンス

A と B の範囲で一つの最も p_T の大きいトリガ候補が選ばれる。Delta は Wire の場合 $\delta R = \pm 7$、Strip の場合 $\delta\phi = \pm 3$

コインシデンスの論理は Doublet の場合図 3.21 のようになっており、Triplet Wire は図 3.22、Triplet Strip の場合は図 3.23 となっている。

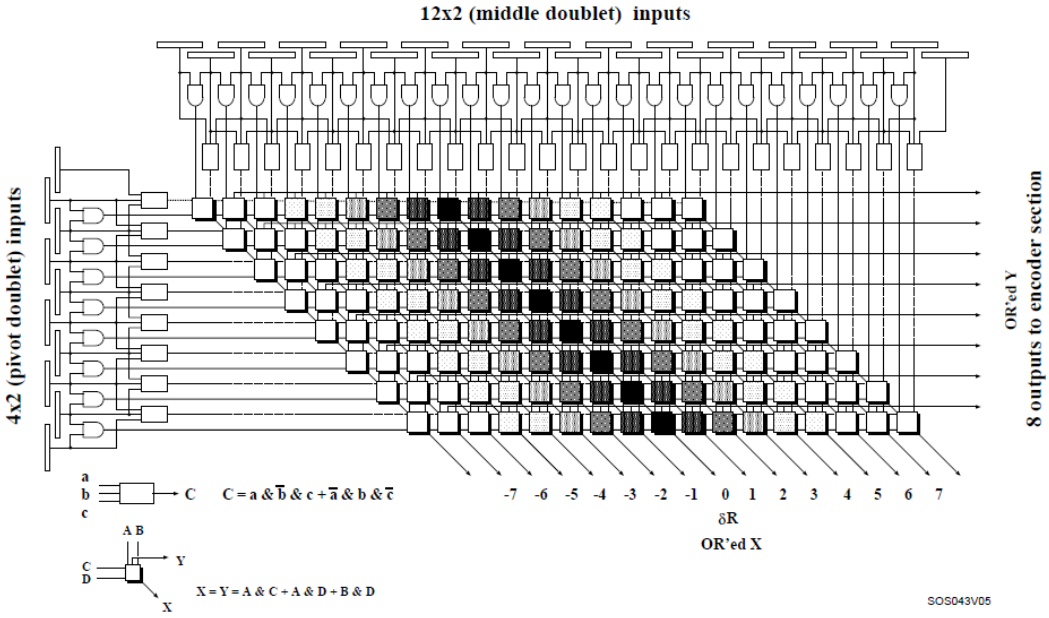


図 3.21　SLB Doublet Coincidence Wire マトリックス [1]

Wire のマトリックスの詳細。Triplet と違い、位置情報だけではなく、無限大運動量トラックとの差の Delta を送る。中心ほど運動量が大きい。

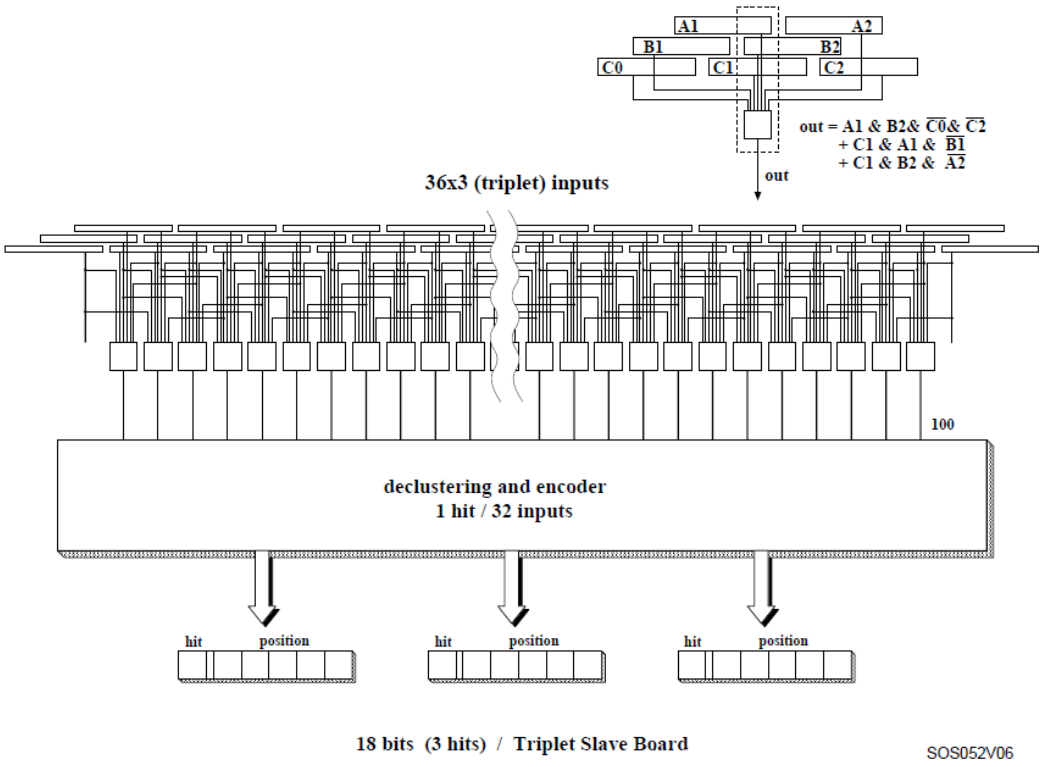


図 3.22　SLB Triplet Coincidence Wire マトリックス [1]

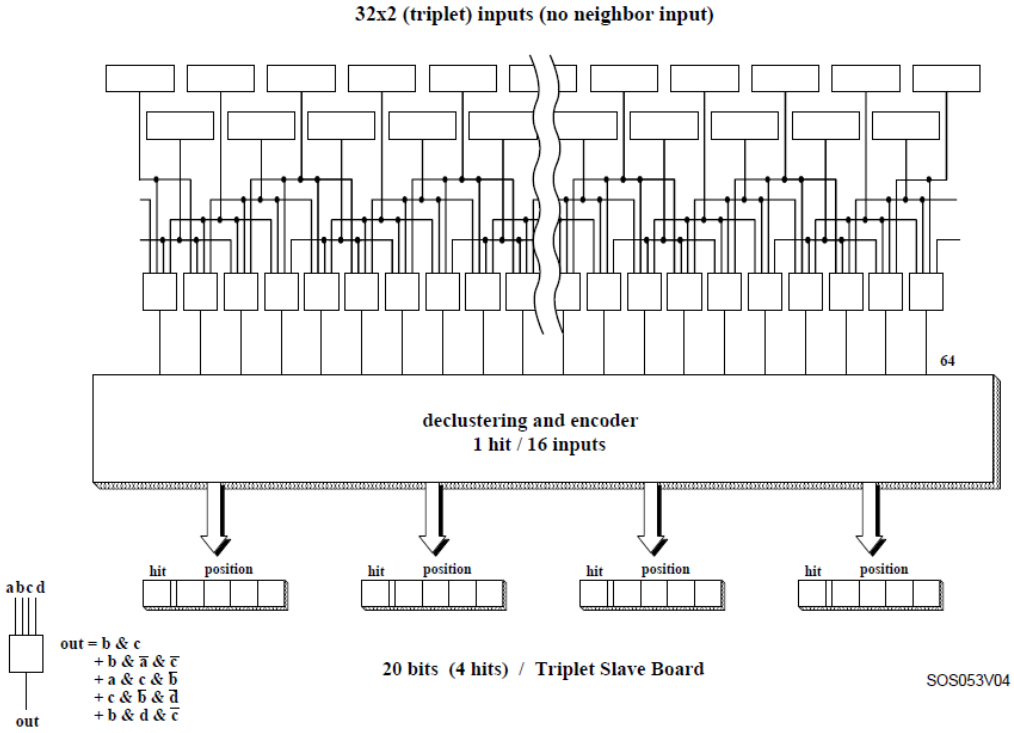


図 3.23　SLB Triplet Coincidence Strip マトリックス [1]

連続したチャンネルに複数の Hit があった場合には、図 3.24 のようにデクラスタリングと呼ばれる 1 チェンネルからの出力にする機能により、一つの粒子による複数の候補の生成を阻止している。

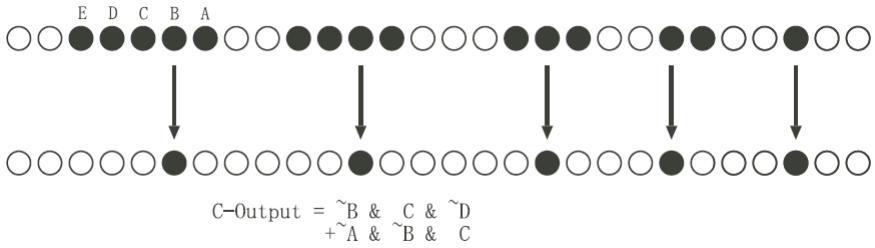


図 3.24　デクラスタリング

連続した Hit チャンネルは一つの粒子によるものと判別し、複数 Hit として数えることを防ぐ。

トリガ出力のデータフォーマットが図 3.25 である。

Doublet は Wire、Strip 共に一つのチップで TRIG0 と TRIG1 の二つの候補を出力し、位置を示す Position と粒子の符号、サジッタの大きさ Pt をそれぞれ 5bit、1bit、3bit で送り出す。

Triplet は Wire の場合、TRIG0、TRIG1、TRIG2 の三つの候補の位置情報 5bit を出力する。Strip の場合、TRIGA と TRIGB それぞれに四つの候補を持ち、位置情報 4bit を出力する。

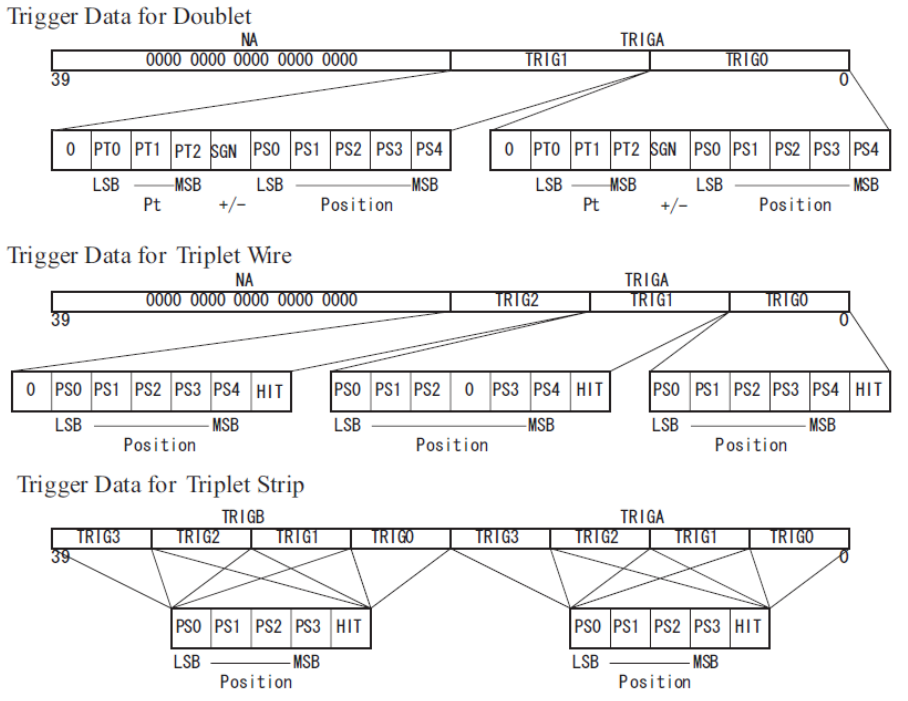


図 3.25　SLB トリガーデータ出力フォーマット

PSx は位置情報、PTx は Delta の大きさ、SGN は粒子 (Delta) の符号、HIT は Hit が合ったかどうかの判別のためのビットとなっている。

リードアウト系

リードアウト系は 1 チップの送る 200bit の情報の内、40 から 199bit の 160bit を Hit 情報として使用する (図 3.26)。160 ビットの情報は A、B、C、D インプットの四つに分けられ、それぞれレイヤーに対応している。例えば TGC3、TGC2 の Wire の A インプットと B インプットは TGC3 の外側のレイヤー (layer7) と内側のレイヤー (layer6) に、C インプットと D インプットは TGC2 の外側のレイヤー (layer5) と内側のレイヤー (layer4) に対応する。

Readout bit position

	Higher-R																																											Lower-R
D-Input	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199
C-Input	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155
B-Input					76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111						
A-Input					40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75						

図 3.26　SLB　リードアウトのビット配置

レイヤー別に A-Input、B-Input、C-Input、D-Input と分けている。

また、SLB には二種類のマスク機能が備わっている。一つはリードアウト系、トリガー系両方に対して、もう一つトリガーラインに対してだけマスクをかけることが出来る。これにより、通常 3outof4 であるトリガ条件をを 2outof2 にしたりすることができる。(図 3.27)。

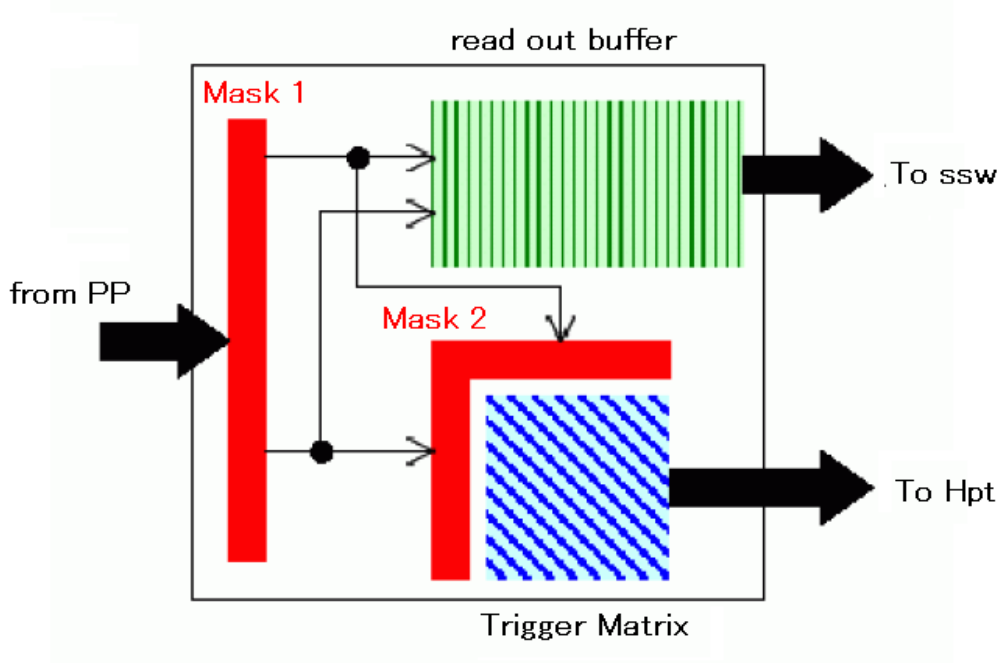


図 3.27　SLB Mask

Mask は二種類あり、リードアウト、トリガ両方にかけるものとトリガ系にのみかけるものがある。トリガ系のみのマスクを用いれば 3outof4 のトリガ条件を 2outof2 などに変更できる。

3.2.3 High Pt Board (HPT)

Highpt ボードは Wheel の脇にセクタごとに設置されている Mini ラックの中の HSC と呼ばれるクレートに SSW と一緒に配置されている (図 3.28)。TGC1 とのコインシデンスを行っており、出力を光で USA15 と呼ばれる部屋にある SL ボードへ送る。

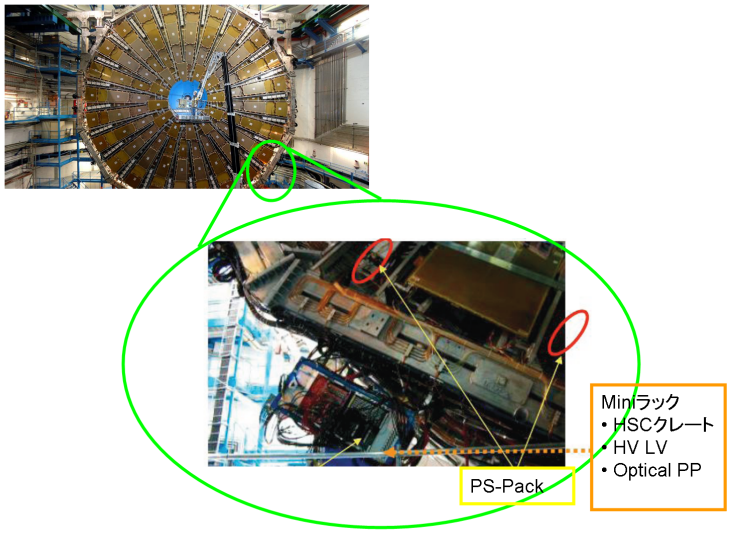


図 3.28　Mini ラックの位置

Mini ラックはセクタごとに Wheel の脇に設置されており、電源や HSC クレート、光信号のための OpticalPatchPanel などが入っている。

Highpt ボードはチップ一つで最大二つのトリガー候補を出力するようになっており、その出力は図 3.31 のようになっている。HitId、Pos といった位置情報や、Delta といった情報を送る。Highpt ボードは現在 Strip の Endcap を担当する部分の chip1 にバグを抱えており、HitId として 1、2、5、6 を送るはずが 1、2 しか送らないようになっている。この問題は Highpt ボードを取り替える他に解決方法は存在しない。

トリガ候補の位置に関連する情報は HitId と Pos で、1 チップが担当する領域を分割し、HitId として番号付けする。その領域の更に半分を Pos という bit で識別し、送り出す (広さ 8ch 分)(図 3.31)。HitId は最大 1 から 6 まであるが、Highpt ボードのチップによって HitId のとりうる値は違う。

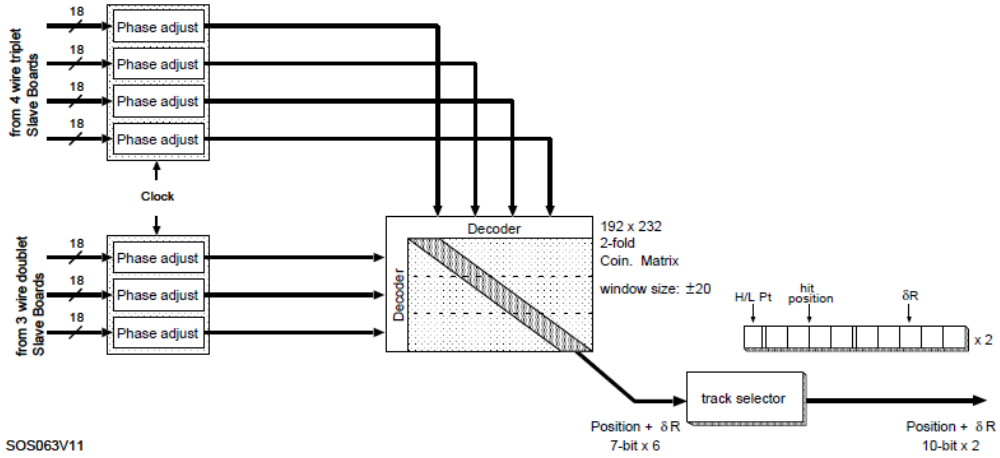


図 3.29　High pt Board Wire Block Diagram

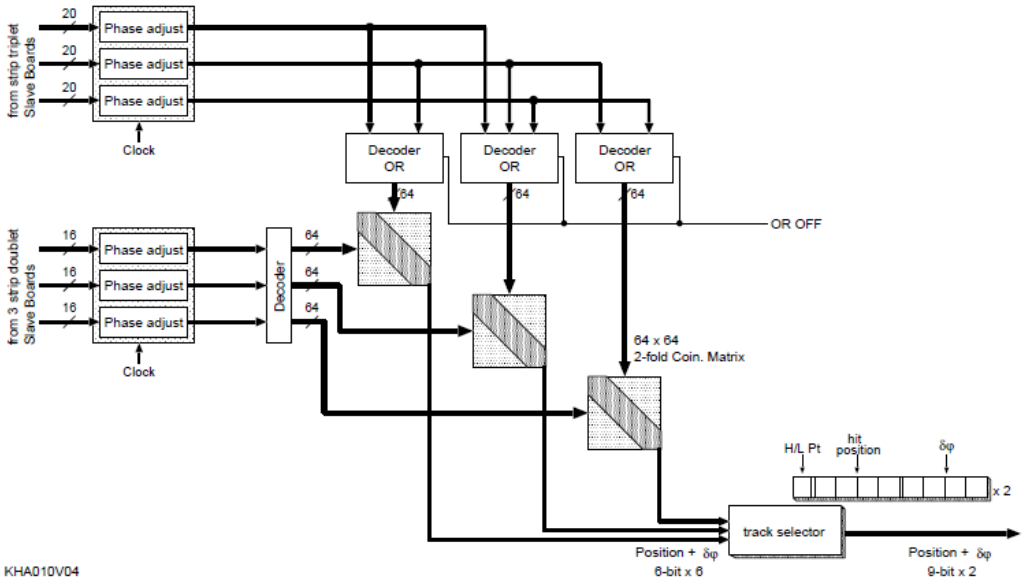


図 3.30　High Pt Board Strip Block Diagram

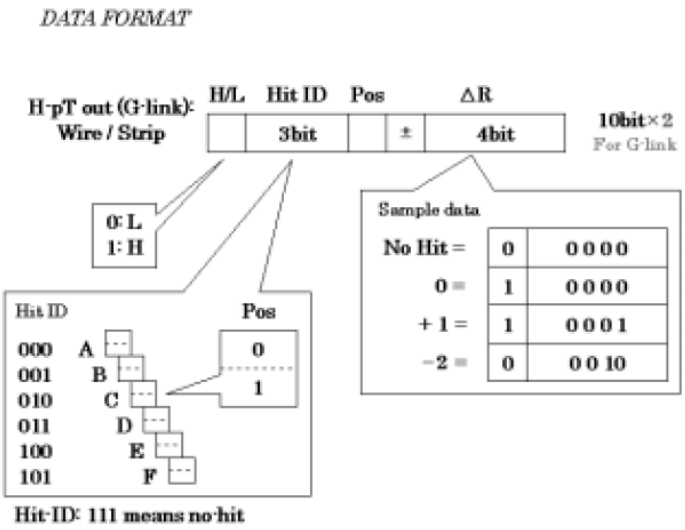


図 3.31　High pt Board Output

H/L は Highpt か Lowpt かの判別。TGC1 とのコインシデンスが取れたら H となる。HitID は位置情報を分割し、更に Pos ビットによって半分に分ける。± は符号で、Δ は SLB と同様の意味

3.2.4 Sector Logic Board (SL)

図 3.32 が SL ボードのブロック図である。Highpt ボードからの信号を受け取り、Wire と Strip のコインシデンスをとる。また、6 段階の Pt 閾値によって分類、フラグをつける。それらの Pt 判別フラグや ROI を CTP へと送る。また、それらのデータや Highpt から出力データをバッファに貯めて、L1A を受け取り、三バンチ分のデータを ROD へ送る。そのためのチップは SLB のチップと同一のものを使用している。

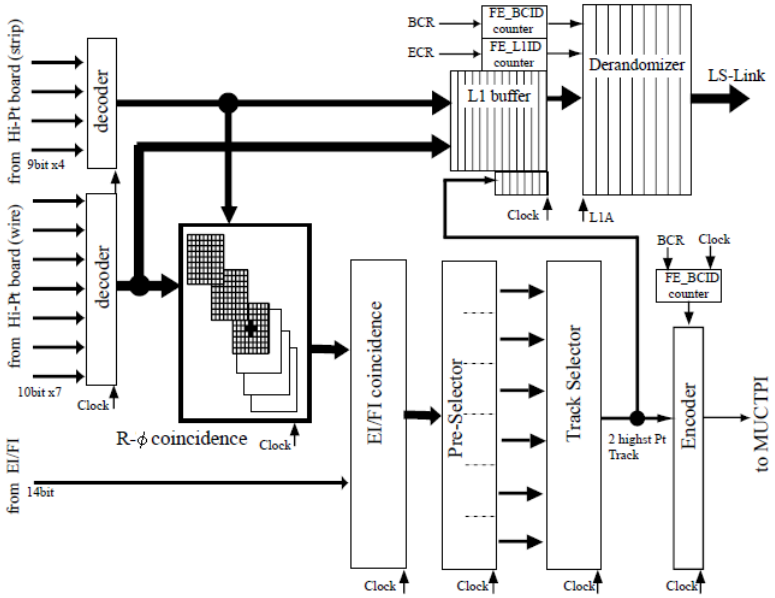


図 3.32　SL Block Diagram

R-ϕコインシデンスを取り、セレクトするのが主な仕事。あとは MUCTPI と ROD に信号を送っている。

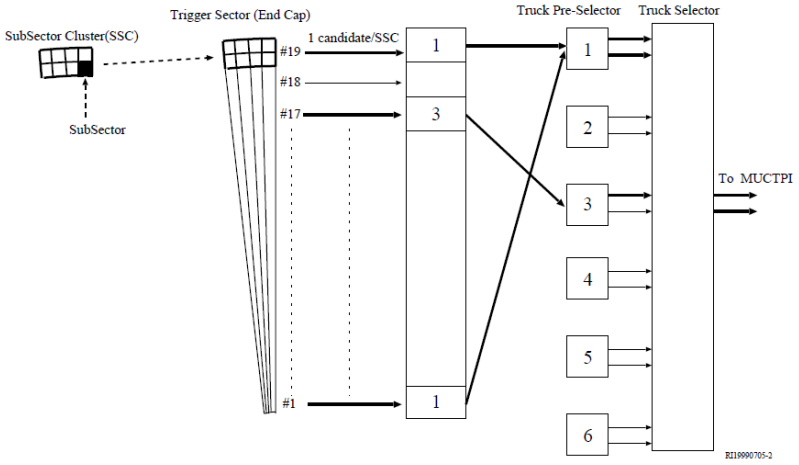


図 3.33　SL のトリガ処理

3.2.5 Star Switch Board (SSW)

SSW は SLB から送られたデータを圧縮し、効率を良くすることである。Wheel の脇にあるクレートに Highpt ボードと一緒に配置されており、光で USA15 に設置されている ROD へと信号を送る。

図 3.34 Star Switch Board

3.2.6 Read Out Driver (ROD) (TGC 用)

ROD は最終的にデータが集まるモジュールである。SSW や SL からのデータを格納し、トリガー情報からヘッダー、トレーラを付け、ROB へと送信する。現在 ROD は二種類のフォーマットでデータを記録しており、一つをデバッグ用途の RawData フォーマット、もう一つを Readout フォーマットと呼び、Athena でのデコードの際にどちらを使用するか、指定することが出来るように今のところはなっている。

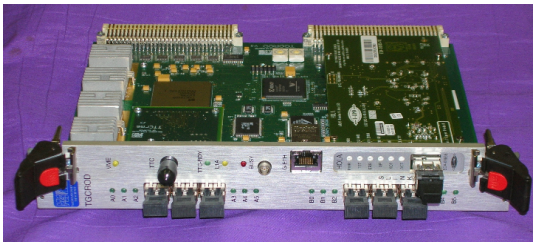


図 3.35 Read Out Driver (TGC)

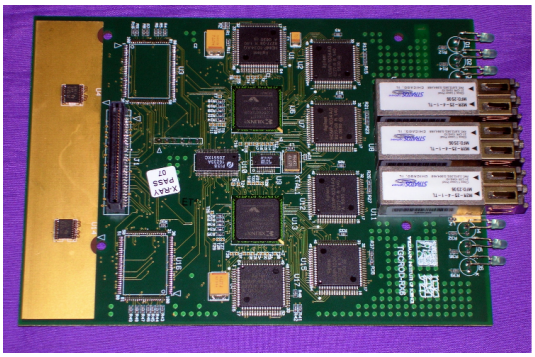


図 3.36 Read Out Driver mezzanine board

3.3 TGC の状況

ここ二年間を中心に、TGC の状況について述べる。

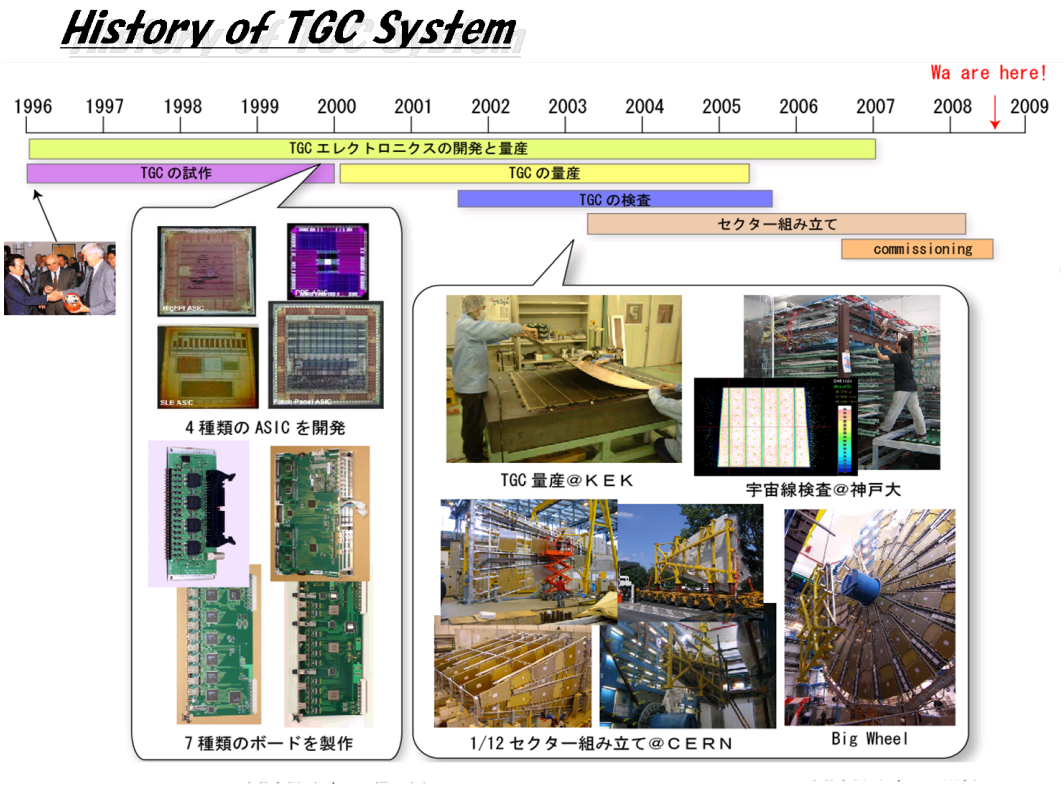


図 3.37　TGC のヒストリー [22]

TGC の歴史。試作から現在まで。

TGC はエレクトロニクスの検査やチェンバの試験、セクタの組み立てが行われていき、それらが完了した後、地上での宇宙線検査を 2007 年の秋初めまで行ってきた [23]。ここでアクセスが容易なうちに出来るだけケーブリングやチェンバ、エレクトロニクスをチェックし、地上動作試験が完了後、セクタごとに輸送、地下へと設置された (図 3.38)。

BigWheel については、2007 年の夏初期に地上動作試験が終わり、その後 SmallWheel について行われていった。その際に、大きな不具合は修正し、今度は地下へ移ってからも、TGC 単体で動作させ、宇宙線データやテストパルスからケーブリングミスやタイミングの調整などを検査している。

また、ATLAS 検出器は、一定期間だけ検出器同士を同時に動かす MileStoneRun を行ってきた。表 2 にここ二年間の Run を載せる。

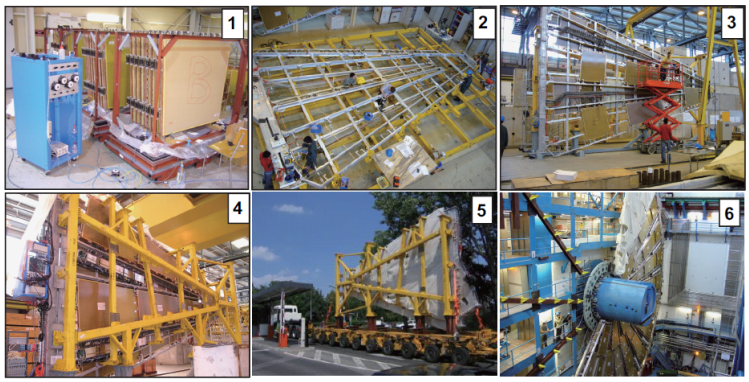


図 3.38　TGC インストール行程 [23]

チェンバのテストが終わったら、セクタ 1/12 ごとに製作。地上動作試験が完了したら、地下へと設置された。

表 2　ここ二年間で行われた Commissioning Run。Mx は全ての検出器が対象で、Px は MuonSpectrometer を対象としている。

Name	Start Date	End Date	TGC 動作 Sector
M3	2007/6/4	2007/6/18	C09
M4	2007/8/23	2007/9/3	C09
P1	2007/10/1	2007/10/7	C09 C10 C11
M5	2007/10/22	2007/11/5	C09 C10 C11 A09 A10 A11
P2	2007/12/3	2007/12/9	C09 C10 C11 A09 A10 A11
P3	2008/2/25	2008/3/3	C09 C10 C11 A09 A10 A11
M6	2008/3/3	2008/3/9	C09 C10 C11 A09 A10 A11
P4	2008/4/11	2008/4/16	C09 C10 C11 A09 A10 A11
P5	2008/5/19	2008/5/25	C03 C04 C05 C06 C07 C08 C09 C11 C12 A02 A04 A05 A06
M7	2008/5/30	2008/6/2	C03 C04 C05 C06 C07 C08 C09 C11 C12 A02 A04 A05 A06
P6	2008/6/17	2008/6/22	C03 C04 C05 C06 C07 C08 C09 C11 C12 A02 A04 A05 A06
M8	2008/7/10	2008/7/20	TGC 24 Sector
Aug Combined Run	2008/8/1	2008/8/31	TGC 24 Sector
Sep Combined Run	2008/9/9	2008/9/30	TGC 24 Sector + EIFI
Oct Combined Run	2008/10/4	2008/10/19	TGC 24 Sector + EIFI

今年九月には Beam が一周し (図 3.39)、陽子 Beam が回ったときのデータを得ることも出来た。2009 年には Beam 衝突が行われるはずである。

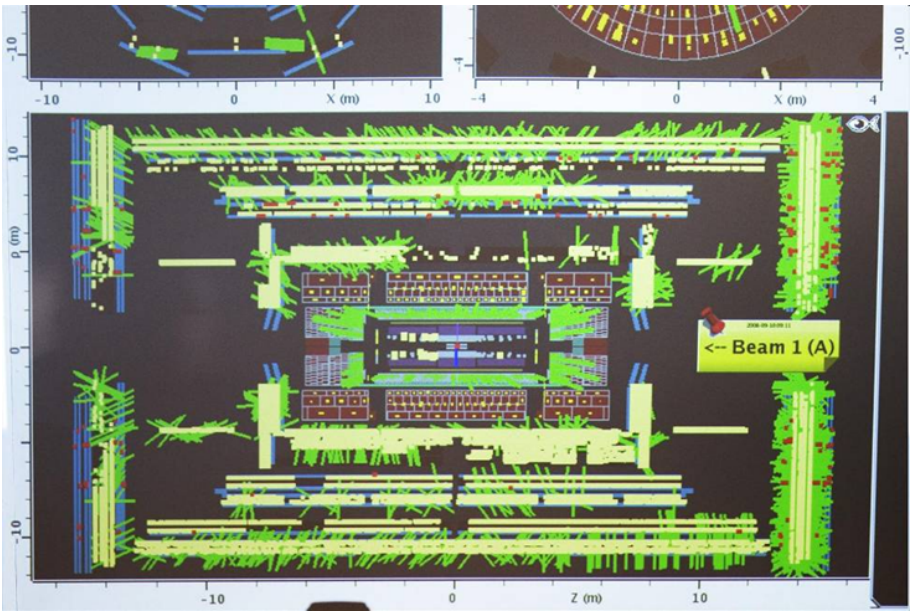


図 3.39　First Beam EventDisplay

TGC は最初 1Sector のみで MileStoneRun に参加し、その後徐々に稼動可能セクターも増え、SingleBeam が出たときは全セクター動作という状態に至った (図 3.40)。

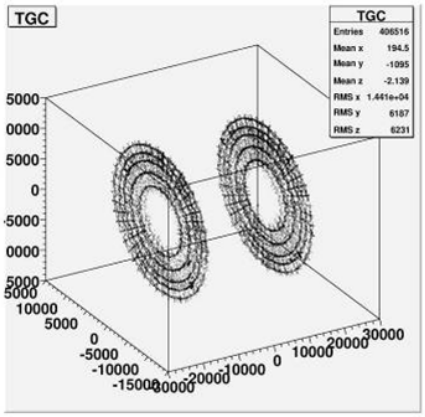


図 3.40　TGC BigWheel 全ての Sector に Hit があり、かけているところがないのがわかる。

現在は検出器の調整を行い、ケーブリングの再検査などを行っている。

4 ATLAS オフラインソフトウェア

ATLAS では一度ディスクに保存されたバイトストリームデータ (01 の羅列) からを切り出し、物理量に変換、物理解析を行うオフラインソフトウェアに Athena というフレームワークを使用する[*6]。Athena は言語として C++、スクリプトとして Python が使われており、各 subDetector や HLT などのグループごとに開発が進められている。Athena は、検出器シミュレーションに Geant4 [9]、パッケージ管理などに CMT [5] といった外部ソフトウェアも使用されている。

また、この章では UML(Unified Modeling Language) に基づいた絵が幾つかでてくる。UML はソフトウェアの開発を考える上で、システムの構造を表現する標準化された言語である。13 種類の図を必要に応じて書き分ける。システムを構成するクラスとそれらの関連の構造を表現するクラス図や、クラスを実体化して生成されたオブジェクト同士の関係を表現するオブジェクト図などがある。

4.1 Athena の概要

Athena(ATLAS Framework) はユーザに一般的な機能やアプローチ、コンポーネント間のやり取りやソースのリユーズなどを提供するフレームワークであり、Gaudi Common Framework Project から派生したものである。Gaudi [14] は元々 LHCb によって発展したものであるが、現在では両プロジェクトでのソフトウェアのカーネルとなっている。

Athena の主要なコンポーネントは図 4.1 の通りである。Algorithm が物理イベントなどの処理を行い、各種 Service が一般的な機能を提供する。例えば Message Service はメッセージ出力などを提供するし、Event Data Service は Transient Event Store からデータオブジェクトをやり取りする機能を提供する。データは Transient Store というメモリ上の領域に保持し、必要になったら取り出すというのが基本となる。また、ディスクに保存するときは Persistency Service が Converter を用いてデータを Persistent 用の形式に変換し保存する。

最も主要な機能は Algorithms がインプットデータを受け取り、計算し、新しい出力データを生成することである。図 4.2 は目的を果たすため、具体 Algorithm オブジェクトがどう他のコンポーネントと作用するのか示した図である。

4.1.1 CMT(Configuration Management Tool)

Athena はシステムやフレームワークで使用されるソフトウェアコンポーネント (アプリケーションやライブラリ、ドキュメントやツールなど) の集合体である Package という単位を用いて開発を行っている。そして、それらのセットを Project と呼ぶ。例えば、リコンストラクションやシミュレーションなどが Project である。その Project、Package 指向開発のユーティリティとして CMT が使用されている。

CMT は大規模物理実験の中でソフトウェア開発をサポートする為のアカデミックプロジェクトであり、LAL(Laboratoire de l'Accelerateur Lineaire - CNRS) という組織によってサポートされている。

ユーザは以下の設定をする。

- 他の Package との依存性

[*6] Athena はオンライン側でも HLT などに使用される。

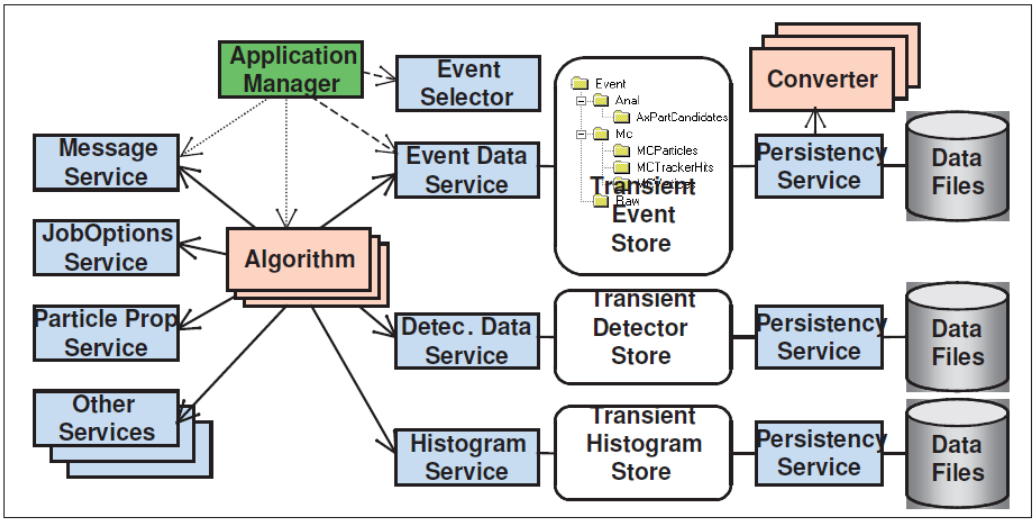


図 4.1　Athena の Object 図

Algorithm により、物理イベントの処理を行い DataObject を生成する。各種 Service によって一般的な機能が与えられる。Transient Store というメモリ上の領域に DataObject を保存し、必要なときに呼び出されるようになっている。ディスクへの保存は Converter により、Persistent 用の形式に変換され保存される。

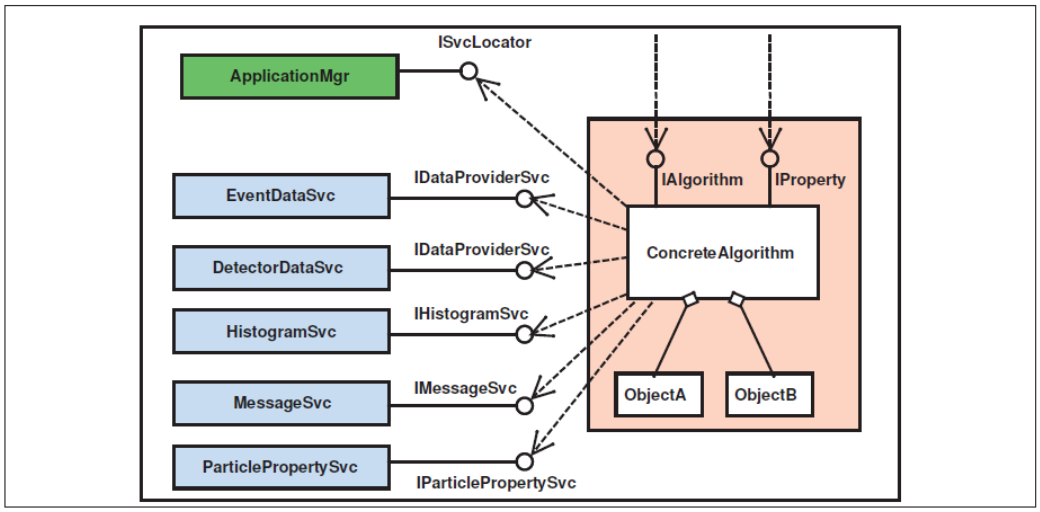


図 4.2　Algorithm が他のコンポーネントと作用するのかを示すコンポーネント図

図の丸はインターフェースを表し、破線矢印はそのインターフェースを実装したコンポーネントを利用することを表す。例えば図の ConcreteAlgorithm は ISvcLocator インターフェースを実装した ApplicationMgr を利用する。また白塗りの矢印は集約を表す。ObjectA と ObjectB という部分により ConcreteAlgorithm は成り立っている。

- プロジェクトのポリシーやジェネラルな Configuration の記述
- 環境変数やマクロなどのシンボル
- Package のコンポーネント (ライブラリなど)
- build や test ツールのパラメータ
- run 時の環境

これらを requirements というファイルに記述し、その設定の Configuration を行い、コンパイルなどを行う。

Package は図 4.3 のようなディレクトリ構造をとる。「cmt」に requirement、「src」にソースコード、「アーキィテクチャ名 (i686-... など)」に実行ファイルやライブラリファイル、「doc」にドキュメント、「Package 名」にヘッダーファイル、「share」にスクリプトや DB ファイルなどを入れる。

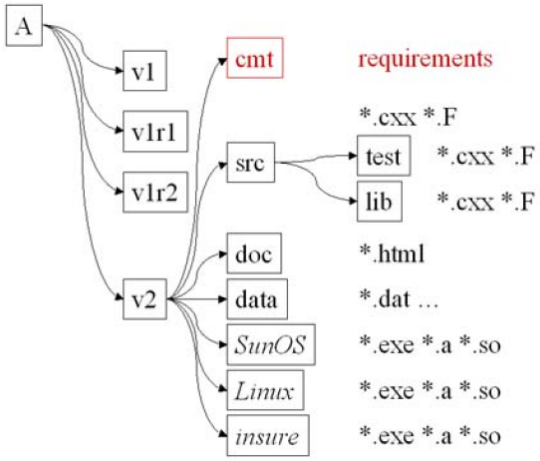


図 4.3　CMT のシンプルな Package 構造 [5]

4.2　Gaudi

Athena のカーネルとなっている Gaudi について、簡単に述べる。

4.2.1　システムの概要、デザイン

前に述べたとおり、Gaudi は元々 LHCb の物理データのプロセス、つまり物理、検出器のシミュレーションから High Level Trigger、リコンストラクション、物理解析、visualization などに適用可能なフレームワークを作成するプロジェクトである。これらのデータフローを図 4.4 に示してある。

Gaudi は、

- Data と Algorithms をはっきりと分ける
- Data は「event」、「detector」、「statistical」という三つの基本的なカテゴリに分ける
- persistent data と transient data とをはっきりと分ける
- logical なアドレスづけによって transient store にある Data の特定がどんな Algorithm からも出来るようにする
- 後々に加えられるユーザーコードのための place holder として Algorithms と Converters を想定

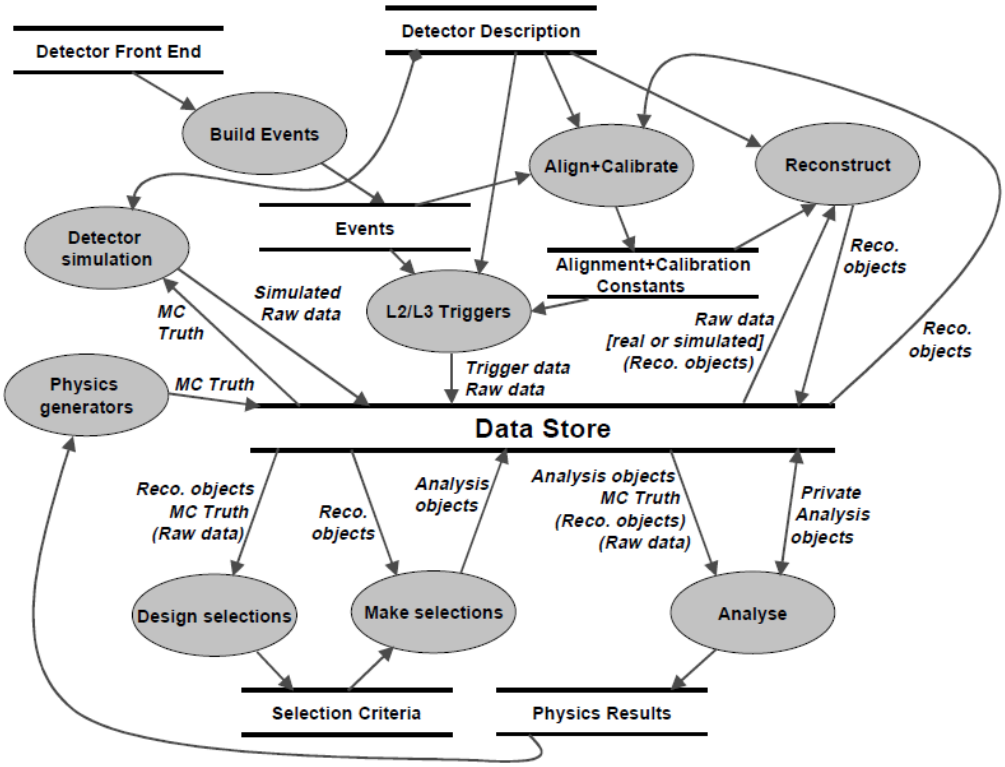


図 4.4　Gaudi のデータフロー [14]

- 全ての Components は interface(C++ の場合 pure abstract class) によって実装する
- 可能な限り Component を Re-Use すること

というのが主なデザイン戦略である。

4.2.2　アーキテクチャ

Gaudi のアーキテクチャの説明として、主要コンポーネントの Object 図を図 4.5 に載せる。主要なコンポーネントは以下のものがある。

Algorithm と Application Manager

Algorithm というイベントデータプロセスを担当するコンポーネントと、それらの生成や呼び出しを管理する、いわゆる「chef d'orchetre」の Application Manager がある。

Stores

データオブジェクトは三つの store に割り振られ、まとめられる (図 4.5)。一イベントのみの event data は transient event store に、ディテクタの description やジオメトリ、キャリブレーションなどの各イベントで変化しない detector data は transient detector store に、ジョブの後に生成されるヒストグラムや Ntuple な

どの statistical data は transient histgram store に保存される。

Services

Algorithm のテクニカルな面は深く考えずに、物理的な部分の開発に集中できるように Services というコンポーネントがある。 Services の幾つかの例が図 4.5 に載っている。例えば、上で述べた Stores について管理する Service(event data service, detector data service etc...)。他にも persistent data から transident data に変換、もしくはその逆を行う persistent service や job Option を扱う job Options service、メッセージを扱う Message service、Algorithm を扱う AlgFactory などが存在する。

Selector

選択を行うコンポーネントである。例えば event selector は処理するイベントを選択する機能がある。他にも transient store から algorithm や service で使われるオブジェクトを選択するための selector もある。

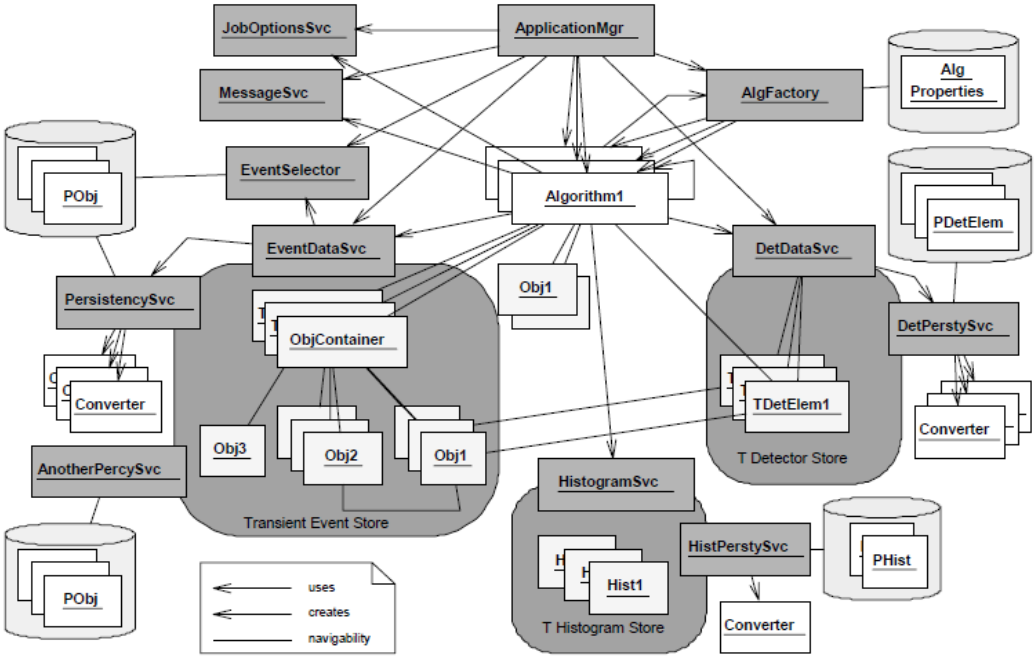


図 4.5　Gaudi のデータプロセスの Object 図 [14]

4.2.3　コンポーネント構成とクラス

システムを詳細に見るためには、ソフトウェア構造はどんなクラスを想定しているのか見る必要がある。アーキテクチャのカーネルを構成する主要なクラスを表 3 に載せる。

また、Services と Algorithms のクラス図を図 4.6 と図 4.7 に載せる。

全ての Services は「IService」インターフェースを継承している「Service」クラスから派生している。この

表 3　クラスの分類

Application Managers	application に対して一つ。algorithm の管理を行う。
Services	well-defined された interface で特定のサービスを提供する。機能に応じて違った実装となる。
Algorithms	物理イベント処理のコード
Converters	特定の event または detector データを他の表現へと変換する役割を持つ。
Selectors	event や detector データのイベント選択を処理するコンポーネント
Event/Detector data	algorithms や converters が使用するデータタイプ
Utility classes	algorithms を実装する助けとなる全ての utility classes

インターフェースは、参照されない変数を自動的に delete する reference counting mechanism など service で必要とされる一般的な機能を与える。また、service を特定する識別子も与える。

一般的にコンポーネントは一つ以上のインターフェースを実装している。そうすることにより、各々のインターフェースを専門化することができる。例えば、Algorithm コンポーネントは物理イベントを処理するための「IAlgorithm」インターフェースを実装しているが、「IProperty」インターフェースも algorithm の振る舞いを変えることができるようにするため実装している。

どんなコンポーネントも一つもしくはそれ以上のコンポーネントのインターフェースを参照しているが、決して具体オブジェクトを参照はしない。これはコンポーネント間の結合を最小にするための絶対条件である。それに加え、全てのインターフェースは「IInterface」という一般的なインターフェースから派生している。これは、コンポーネントのインターフェースに他のインターフェースへの参照のための問い合わせのメカニズムを提供するためである。

4.2.4　Transident Data Model

transient data store の中のデータは、ファイルとディレクトリ (ディレクトリはデータ属性を持つ) の関係と非常に良く似た木構造となっている (図 4.8)。データは木構造のノードとなることができ、自身のデータメンバやプロパティも持つことが出来る。例えば、Event オブジェクトは全てのイベントデータの root ノードとなっており、イベントナンバーやイベントタイム、イベントタイプなどのプロパティを持っている。

ほとんどのイベントデータオブジェクトはごく小さなものである。例えば、各イベントでの Hit は数バイトの大きさである。これらによるオーバーヘッドを防ぐため、識別可能なオブジェクトの中にそのようなオブジェクトをまとめたりする。

transient data store の中のオブジェクトに対して次のようなタイプを仮定している。

- 他のオブジェクトへのノードでもあるプロパティを持った識別可能なオブジェクト (directory with properties)
- 通常の識別可能なオブジェクト (files)
- 一つの識別可能なオブジェクトの中にある単純なオブジェクト (record)

図 4.9 にはデータオブジェクトのクラス図を載せる。

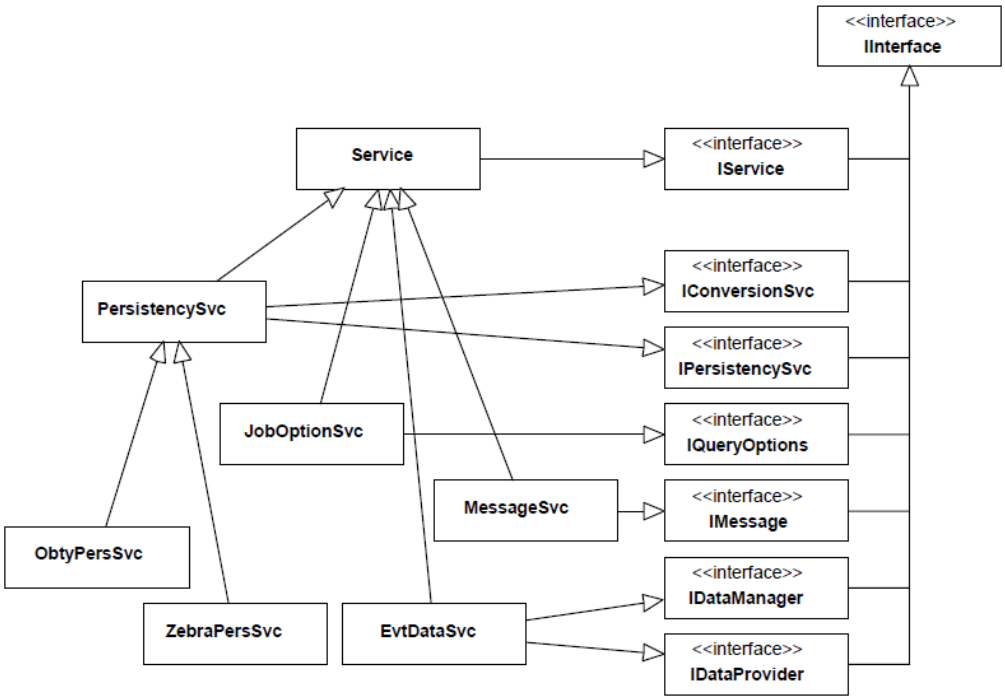


図 4.6　Gaudi の Service のクラス図 [14]

白抜きの矢印は継承を表している。例えば Service クラスは IService インターフェースを継承している。

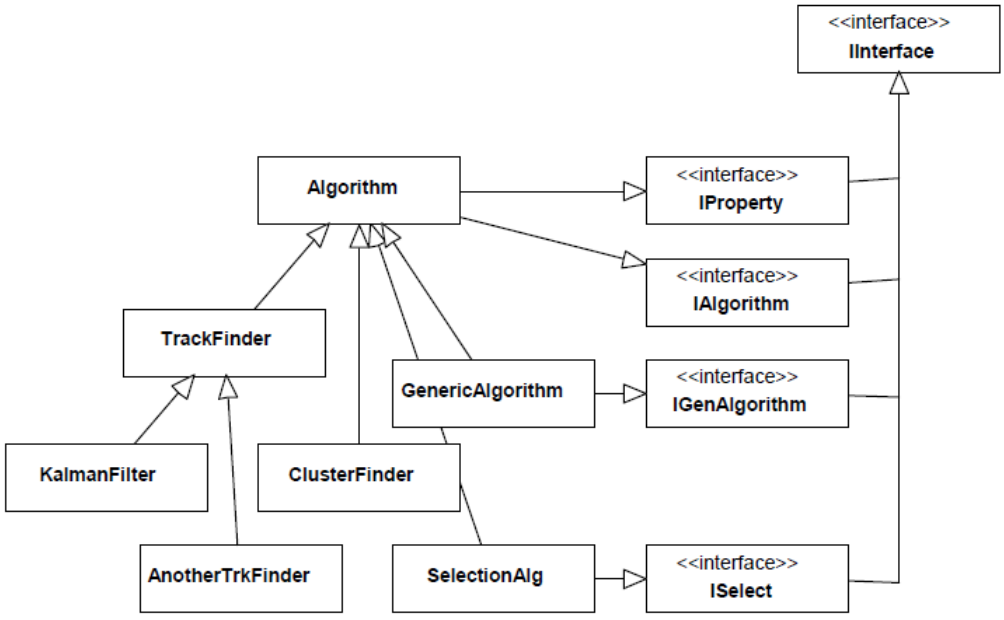


図 4.7　Gaudi の Algorithm のクラス図 [14]

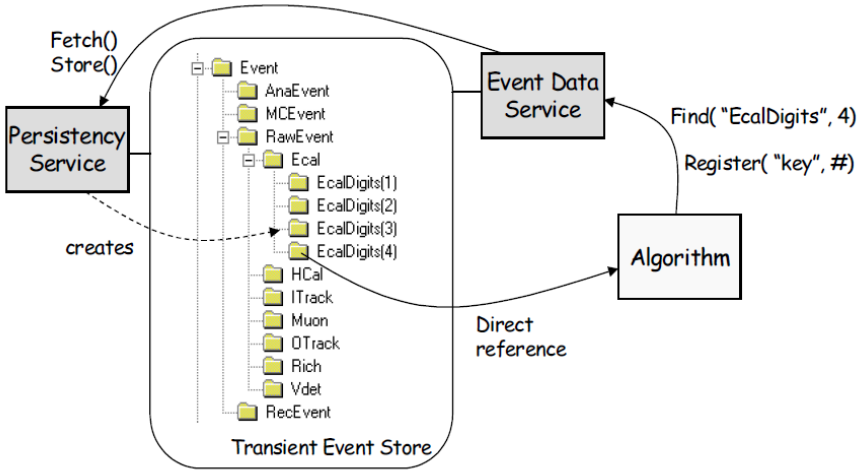


図 4.8　Transient Data Store の構造 [14]

Algorithm が Event Data Service を通して Store から EcalDigits などを取り出したり、登録したりする。Event Data Service も Persistency Service に対して取り出しや保存を要求し、ディスクとやり取りする。

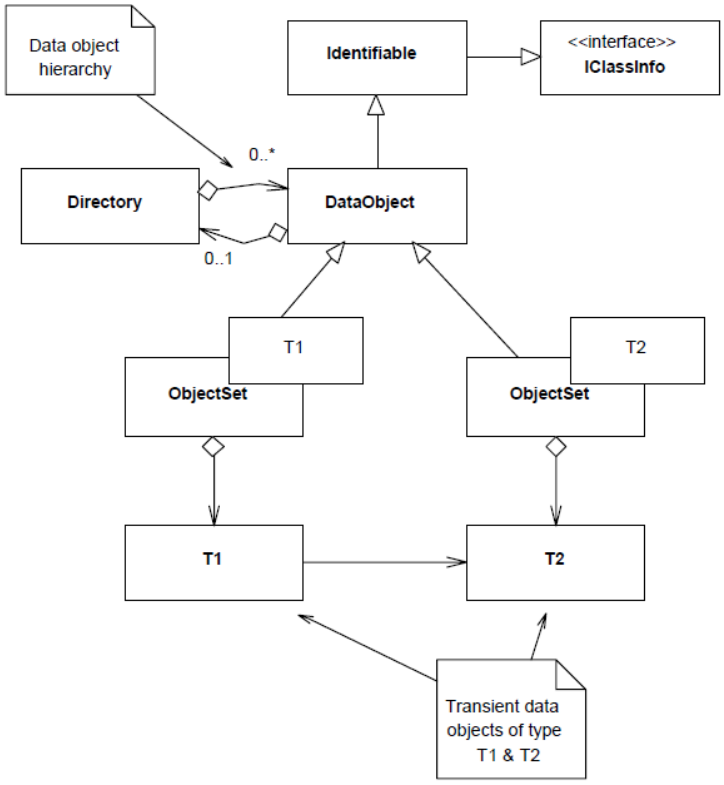


図 4.9　Transient Data Model のクラス図 [14]

白抜きのひし形は全体と部分のという概念を表す。白抜きひし形側が全体を表し、矢印側が部分を表す。ObjectSet は T1 オブジェクトという部分から構築されているのがわかる。それらは DataObject クラスを継承しており、階層構造を持つように設計されている。

4.2.5 Algorithms と Transient Data Store

複雑な Algorithm は一つもしくは複数の基本的な Algorithm によって実装される。このような結合された Algorithms はその役割を果たすために、データ交換を行う必要がある。そのコミュニケーション手段として、transient data store を使用する。図 4.10 に示しように algorithm はデータオブジェクトを data store の中に置き、他のがそれらを取り出す (retrieving) ことにより、algorithm から他の algorithm へデータを渡しているように見せている。

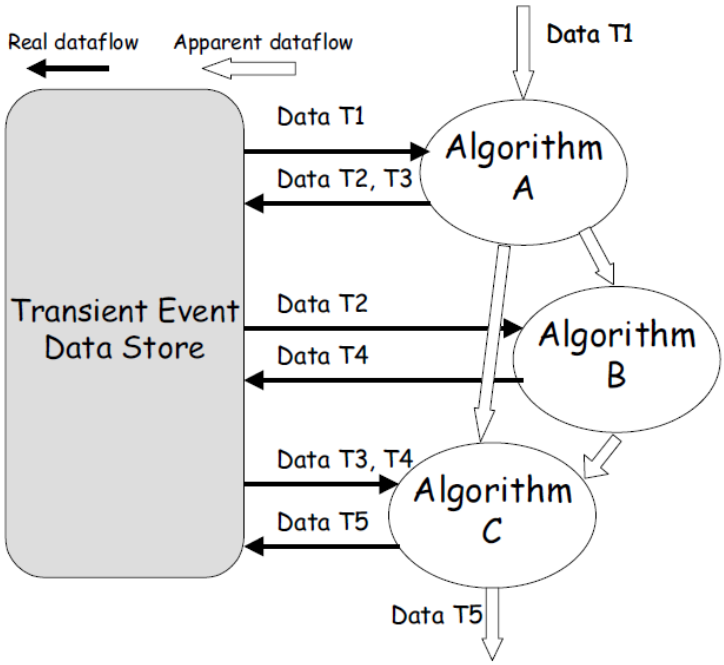


図 4.10　Algorithm と Transient Store Service の関係 [14]

Algorithm では直接別の Algorithm にデータを渡しているように見えるが、実際は Transient Store に保存、取り出しを行うことによって渡している。

4.2.6 Transient と Persistent data

デザインの節で述べたとおり、transient と persistent では異なるオブジェクト表現となっている。これにより、二つを独立に最適化でき、デスクに記憶する技術から切り離せるようにした。

そこで、オブジェクトをそれぞれに変換する必要がある。その方法は幾らかあるが、Gaudi は各々のデータタイプに対して変換専用のコーディングをする方法を取っている。また、transient クラスにも persistent クラスにも変換コードを配置したくない。そこで、共通のインターフェースを持ち、オブジェクトを変換する必要があるとき呼び出される、Converter と呼ばれるコンポーネントを使用する (図 4.11)。

上の仕様だと、ただ単に persistent を transient に変換するだけよりも、もっと複雑な処理を行うことができる。例えば、persistent ストレージ内で小さなオブジェクトを一つにまとめ、オーバーヘッドを最小化することなどができる。

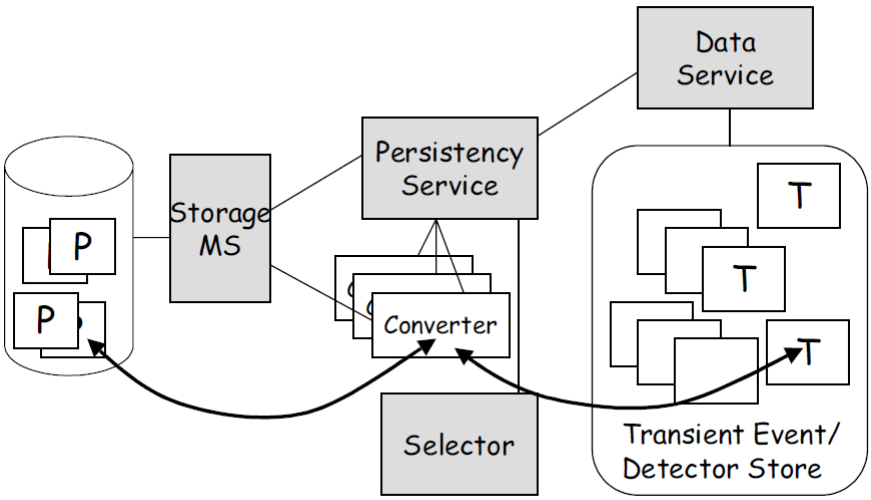


図 4.11　Persistent と Transient 表現 [14]

Transient にあるデータを Converter によって形式を変換しストレージへと保存する。これによりディスク保存の技術から transient のデータ形式などを切り離して開発できる。

4.3　Event Data Model

ATLAS 実験は何年という長期間続くプロジェクトであり、ソフトウェアもそれに耐えうるものでなければならない。ATLAS Event Data Model はディテクタやトリガーといったグループや、online データ処理と offline リコンストラクションなどに共通性、一貫性を持たせ、メンテナンスを容易にするためのモデルである。たとえば、Track クラスというモデルは Muon 単体での Track だけでなく、Inner Detector(ID) や Combined Track で使用される。

ATLAS は一年で PetaByte 単位のデータを生成するので、サイズがでかいまま広範囲で扱うのはいただけない。そこで、解析を行うのに二つのデータセットを導入する。

1. ESD　ディテクタリコンストラクションの出力、つまり粒子識別のための情報や、トラック、jet キャリブレーション、などが入っている。ESD は 1Event で 500kB を目指している。
2. AOD　一般的な解析のための情報が入っている。ESD から作られ、raw データから処理する必要がない。AOD は 1Event で 100kB を目指している。

この二つのデータセットは RDO(Raw Data Object) と PRD(Prep Raw Data) と呼ばれるデータを元に生成される。RDO は検出器から得た 0 と 1 のデータ (ByteStream) から、ビットの値を切り出した値が入っている。それから OfflineID という Identifier を持った DriftCircle や Cluster といった意味を持つ PRD オブジェクトを作る。Identifier(OfflineID) は IDHelper から Detector の場所や種類などをとってくることができる 32bitWord の ID である。

ATLAS は大きく分けて荷電粒子の運動量を測定するトラッカー (Inner Detector と Muon Spectrometer) とエネルギーディポジットを測定するカロリーメータ (Tile と Liquid Argon) の二つがある。前に述べたとお

り、EDM はこれらのシステムでのコードの共有が大きな目的である。

4.3.1 カロリーメータ

二つのカロリーメータでは raw データの時点ではフォーマットは違うが、図 4.12 のように raw データが CellMaker Algorithm を通って CaloCells データオブジェクトを生成した時点で、二つのカロリーメータデータクラスは共通化されている。隣り合う CaloCells は「towers」を作り、それらは「Clusters」を生成するのに使用される。

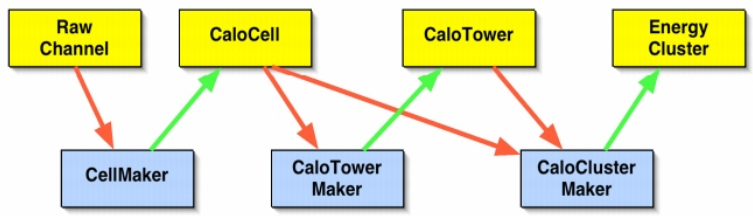


図 4.12　カロリーメータデータフロー　黄色がデータオブジェクト、青が Algorithm [12]

4.3.2 トラッカー

先のカロリーメータと同様で、異なるディテクタでコードシェアをすることが EDM に対する要求である。そこで Track クラスには Track Parameter、Hit Cluster や Drift Circle などのインターフェースを用意する。

図 4.13 が Track のリコンストラクションの様子である。Bytestream Converter はディテクタからデータを受け取り、Raw Data Object(RDO) を生成する。RDO は Prep Raw Data(PRD)、すなわち Cluster や Drift Circle を生成するのに使われる。そして PRD は Track を見つけること、vertex を見つけることや AOD レベルで使われる TrackParticle を生成するのに使用される。

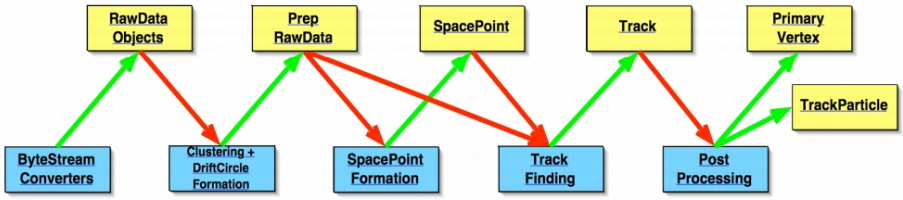


図 4.13　トラッカーデータフロー　黄色がデータオブジェクト、　青が Algorithm [12]

図 4.14 が Inner Detector のクラスを図示したものである。TRT や SCT、Pixel の RDO は InDetRawData クラスを継承し、PrepRawData として、DriftCircle と SICluster が TrkPrepRawData を継承する。

図 4.15 が Muon Detector のクラスを図示したものである。RawDataObject は共通のクラス MuonDigit を継承し、PrepRawData は ID と同様に DriftCircle と Cluster が TrkPrepRawData を継承している。

EDM は決して静的なものではなく、年月と共に変化することはあるが、基本方針は変わらないものである。これらは抽象クラスやシェアデータオブジェクトを使用し、大規模開発を容易することにある。

4.4 Athena ソフトウェアチェーン

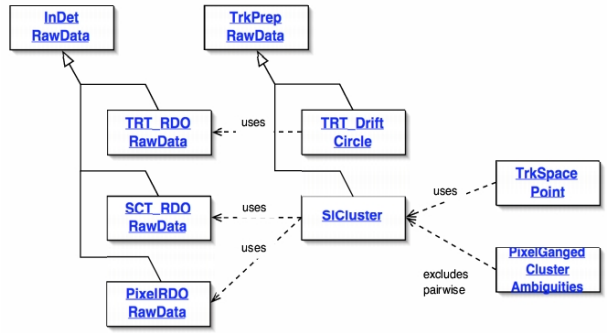


図 4.14　ID Input Class [12]

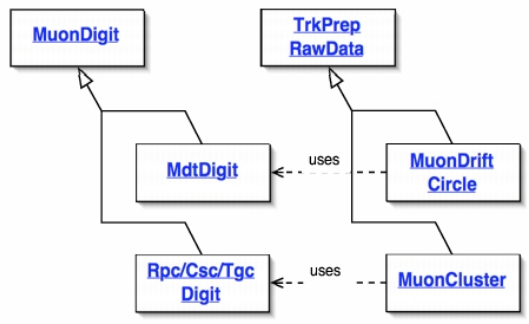


図 4.15　Muon Input Class [12]

実際にシミュレーションから解析までの流れが図 4.16 である。シミュレーションは粒子を生成する Generation の過程を経て、モンテカルロ Truth を生成する。その後、Geant4 により、検出器ののシミュレーションを行い、信号を生成、リコンストラクションを行い、ESD(Event Summary Data)、AOD(Analysis Object Data) を生成する。それら 4 つのデータオブジェクトから、Ntuple を生成し、Root マクロや、Athena 上でヒストグラムを作成するなどの解析を行う。

4.5　Athena Project build

ATLAS は多くの人の開発が必要であり、その為、タグづけは重要になるし、開発重視か安定性重視かなどの目的によっても使用する Package は異なる。そこで目的に合わせた Project をつくり、ソフトウェアの開発やライフタイムを別にして開発できるようしている。その為の Working Model の説明をする。

図 4.17 に Project の依存関係が載っている。

表 4 に Offline Project の概要を載せる。

本論分の解析に関しては、AtlasTier0 の 14.4.0.2 を使用してデコードを行ったデータを使用している。

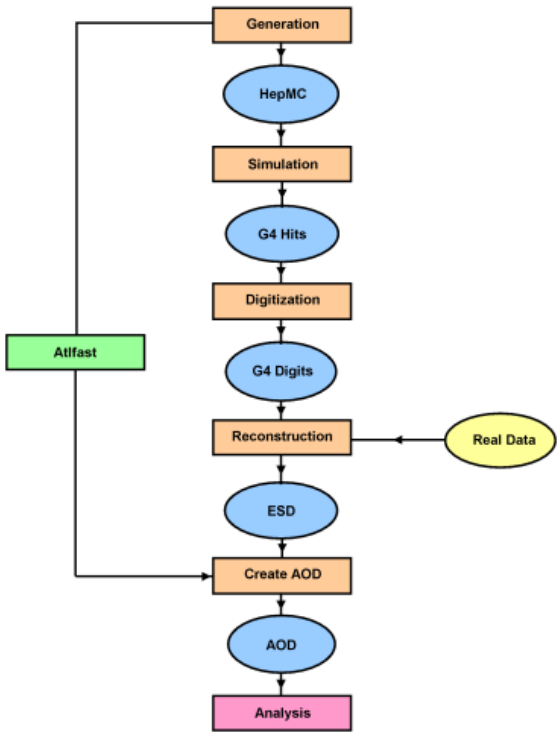


図 4.16　ソフトウェアチェーン

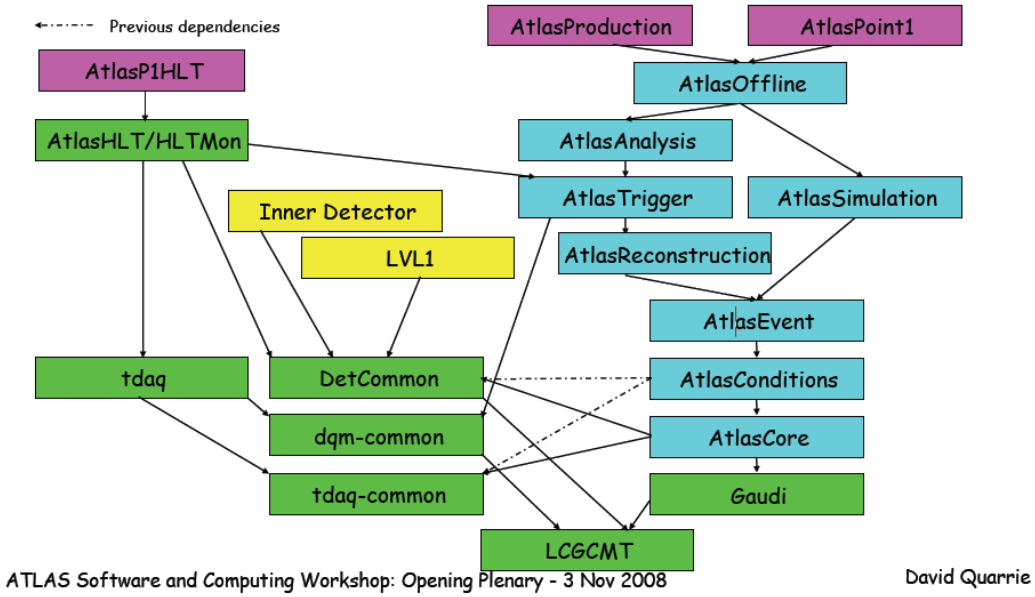


図 4.17　Offline Release Project の構造と依存関係

表 4　Atlas Offline Project

ProjectName	依存先	説明
AtlasAnalysis	AtlasTrigger	物理解析、モニタリング、イベントディスプレイに関連する Algorithm や Service や Tools が入っている
AtlasConditions	AtlasCore	Detector の description、geometry、calibration 情報が入っている
AtlasCore	Gaudi、LCGCMT	中心となるコンポーネントや Services が入っている
AtlasEvent	AtlasConditions	Event Data Model とそれらのサポートクラスが入っている
AtlasOffline	AtlasAnalysis、AtlasSimulation	インタラクティブに使用するのための Project 階層構造のデフォルトの導入点
AtlasPoint1(AtlasTier0)	AtlasOffline	Point1 で使用される online のためのパッチを扱うための Project[*7]
AtlasProduction	AtlasOffline	Offsite Production のためのパッチを扱うための Project
AtlasReconstruction	AtlasEvent	リコンストラクションと Atlfast という Simulation パッケージの為に使用される Algorithm や Service、Tools が入っている
AtlasSimulation	AtlasEvent	Geant4 シミュレーション、ジェネレータ、pile-up、digitization の Algorithm、Service、Tools が入っている
AtlasTrigger	AtlasReconstruction	High Level Trigger に関連する Algorithm、Service、Tools が入っている

5 Muon ソフトウェア

TGC を含む Muon オフラインソフトウェアは MuonSpectrometer 内で開発されている。図 5.1 に Muon ソフトウェアのリコンストラクションの流れを載せる。

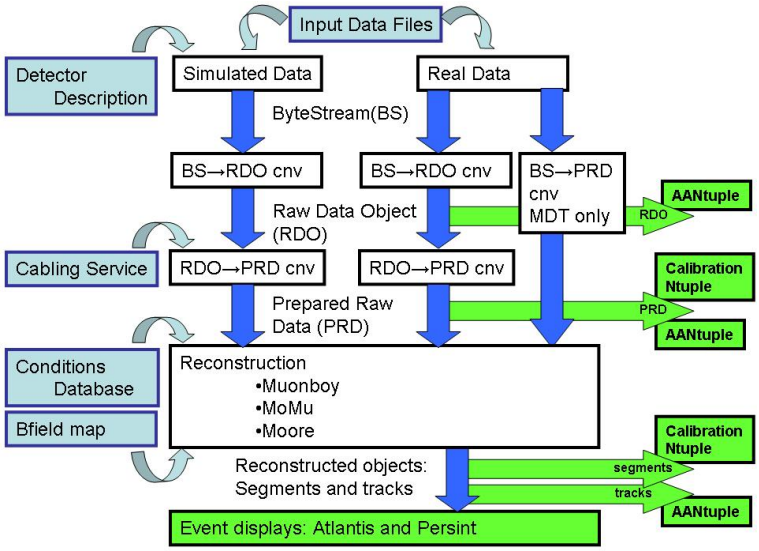


図 5.1 Muon Reconstruction Flow Chart [7]

BS(ByteStream) から RDO、PRD を生成、それを元にトラッカーを生成。それらを Ntuple に入れ、解析を行うことが出来る。

ByteStreamConverter により、バイトストリームを RDO へと変換、それを RDOToPRDConverter により PRD へ、その後、MuonBoy や Moore といった Tracker パッケージへと行き、Track などが作り出される。

その各段階での値を Ntuple へ入れ、解析を行う。特に TGC に関して解析を行う場合、一般的には PRD を用いる。

5.1 TGC ソフトウェア Package

TGC グループで担当した MuonSpectromter の Package で PRD 生成まで行う上で、主なものは以下である。

- MuonCnv/MuonTGC_ CnvTools
- MuonCablings/TGCcablingInterface
- MuonCablings/TGCcabling
- MuonCablings/TGCcabling12
- MuonCommissioning/MuonCommAlgs
- MuonRDO
- MuonReconstruction/MuonRecEvent/MuonPrepRawData
- MuonReconstruction/MuonRecExample

これらの Package の役割をざっと説明する。

5.1.1 MuonTGC_ CnvTools

この Package は TGC のバイトストリームから PRD までのコンバータをまとめてある。バイトストリームから TGC のデータを切り出し、RDO オブジェクトを作り出し、コンテナへ入れる。また RDO から PRD への変換も行い、コンテナへ入れる。

TGC については現在、フォーマットが二種類存在し、それぞれ「Readout」と「RawData」と呼んでいる。ROD で二種類のデータを送り出すようになっており、両方ともディスクに保存されている。それを jobOption により、どちらを使うか設定できるようになっている。つまり、この Package には「Readout」と「RawData」の二種類のコードが存在していることになる。「RawData」フォーマットの目的はデバッグ用で、SSW からのデータをそのまま送り出しているものである。本論分は「RawData」によるデコードデータを使用している。

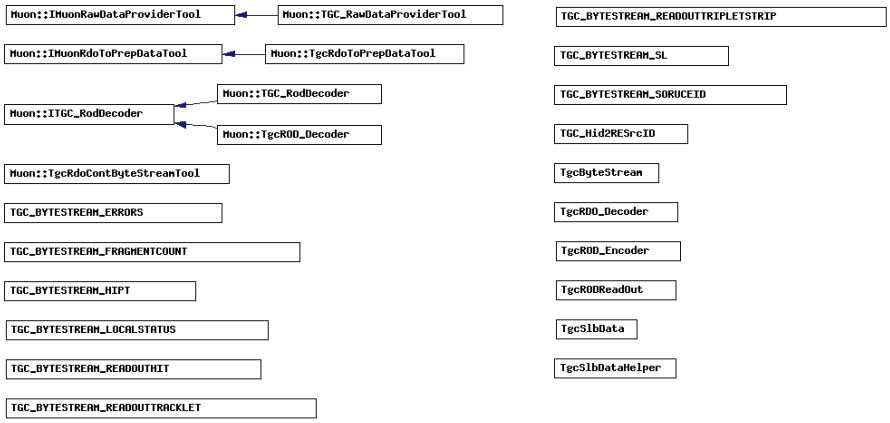


図 5.2　MuonTGC_ CnvTools の現在のクラス図 [15]

継承関係のみを黒塗りの矢印で表示している。

5.1.2 TGCcablingInterface TGCcabling TGCcabling12

この中の TGCcablingSvc によって全ての TGC エレクトロニクスシステム (ASD-PP-SLB-HPB-SL and ASD-PP-SLB-SSW-ROD) のモジュールや出力、入力のオブジェクトを生成、それらの変換を行うといったソフトウェアのケーブリングを管理する。また、OfflineID、OnlineID、ReadoutID[*8]の三種の変換も行う。

TGCcablingInterface がベースとなっており、実装が TGCcabling、TGCcabling12 となっている。TGCcabling が 8-fold、つまり 1/8 セクターでの関数で、TGCcabling12 が 12-fold、1/12 セクターの関数となっている。

[*8] 他のグループは ReadoutID のことを OnlineID と呼ぶこともある。

5.1.3 MuonCommAlgs

この Package は MuonSpectromter の RDO、PRD の変数を Ntuple に入れる仕事をしている。TGC は PRD のみを対象としている。この Package を変換することで Ntuple に入る変数を変えることができる。

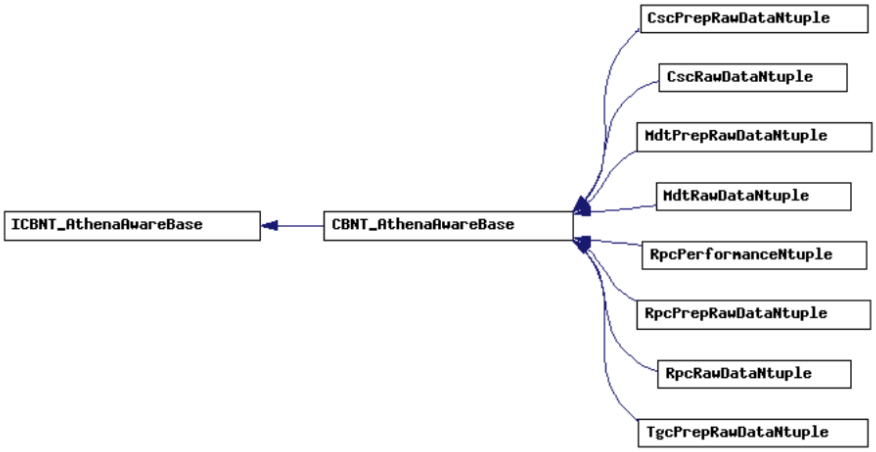


図 5.3　MuonCommAlgs の現在のクラス図 [15]

5.1.4 MuonRDO

MuonSpectromter の RDO について定義している Package である。RawData はコレクションに入っており、そのコレクションはコンテナというコレクションに入っている。例えば TgcRawData は TgcRdo というコレクションに、TgcRdo は TgcRdoContanier というコンテナに入っている。この Package の中を見れば RDO に何が入っているのか確かめることが出来る (図 5.5)。

5.1.5 MuonPrepRawData

PRD について定義している。PRD は OfflineID という、一意に決まる 32bit の ID を持っており、それから、各種情報 (channel やサイド、チェンバなど) を引き出すことができる。PrepRawData は RDO と同様にコンテナの中のコレクションに入っている。

5.1.6 MuonRecExample

MuonSpectromter 単体のリコンストラクションを実行するときに使用する python の JobOption ファイルなどのテンプレートなどが入っている。前までは MuonCommissining/MuonCommRecExample が使用されていたが、14.4.x からこちらを使用するように勧められている。MuonRecExample は、MuonSpectrometer ディレクトリ内だけでなく Reconstruction ディレクトリ内の JobOption も読み込んでいるので、そちらを書き換える場合もある。この Package の中の python ディレクトリと share ディレクトリに jobOption ファイルがある。この中で、athena の引数として実行するのは、実データの場合 MuonDataRec_ myTopOptions.py でデフォルトでバイトストリームからのリコンストラクションを想定している。

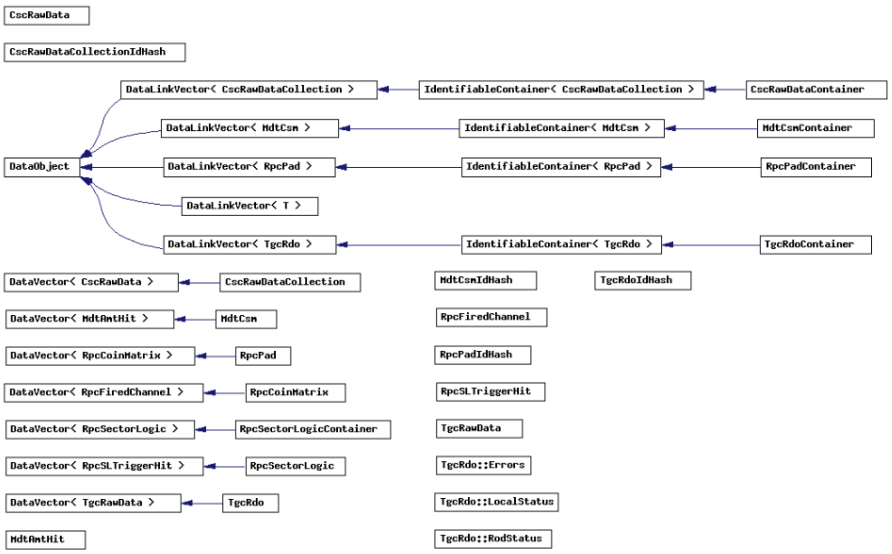


図 5.4　MuonRDO の現在のクラス図 [15]

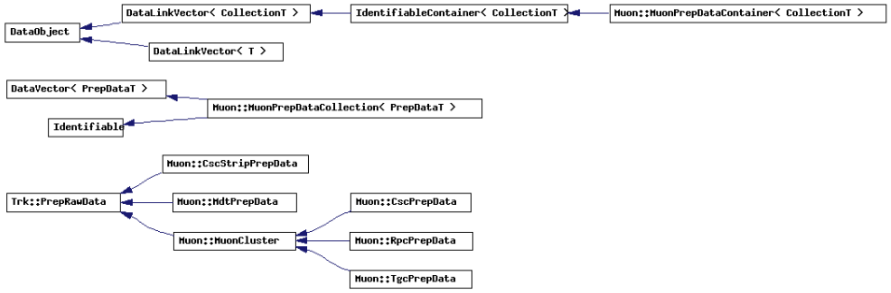


図 5.5　MuonPrepRawData の現在のクラス図 [15]

5.2　TGC のデータ

TGC の解析を行う際に使用する PRD とその元となる RDO について説明する。
TGC のデータは RDO、PRD ともに以下の四種類のオブジェクトがある。

- Hit (Readout)
- Tracklet (Coincidence)
- HighPt (Coincidence)
- SL (Coincidence)

Coincidence データはトリガーデータなどと呼ぶ場合もある。Hit はリードアウトラインで得られた TGC のデータ (図 3.26 参照) であり、Tracklet は SLB が Highpt ボードへ送る 40bit の情報 (図 3.25 参照) について、HighPt は HighPt が SL ボードに送る情報 (図 3.31 参照)、SL は SL ボードが CTP へ送る情報を切り出したものである。Highpt データは RDO の場合は TGC1 とのコインシデンスが取られなくても、生成される

が、PRD の Highpt データは H/L フラグが True、つまりコインシデンスが取れた場合にしか生成されないようになっている。

Coincidence データについては 2008 年の初めに PRD が作られるようになり、デバッグを行ってきた。TGC の RDO、PRD の全てのデータは 3 バンチ分 (Prev、Current、Next) のデータを保持している。

5.2.1 RDO

RDO というのはエレクトロニクスからの出力を切り出したそのままの値をもったオブジェクトであり、モジュールの ID や、bit の値などが入っている。我々のグループはこの RDO に入っている変数を ReadoutID と呼んでいる。TgcRawData は四種類 (HIT、TRACKLET、HIPT、SL) に対応できるようにコンストラクタやメンバが用意されている。

付録の TGC データの項に TgcRawData のコンストラクタに入る変数を載せる。

5.2.2 PRD

PRD は OfflineID という決して重複することのない ID を持っている。この ID を用いて、そのチャンネルや、R、φ方向のチェンバ識別、ステーション (TGC1、TGC2、TGC3、EIFI) の判別や、位置情報などが取れる。

タイミング情報を示す bcTag の様な変数をオブジェクト自体は持たず、3 バンチ (Prev、Current、Next) に対応する三つのコンテナが用意されていて、それぞれ Key を指定して transident store から取り出す。

上で述べたように、コンストラクタの持っている変数以外にも、TgcIdHelper というオブジェクトから OfflineID を用いて場所を特定する変数をとってくることが出来る。TgcIdHelper から得られる変数は、stationName、stationPhi、stationEta、gasGap、isStrip、などがある (付録 TGC データの項参照)。

5.3 TGC データの整合性

我々は TGC の RDO と PRD のデータについて、それぞれをヒストグラムで表示させ、とるべき範囲内に収まっているのか、また、Hit、Tracklet、Highpt、SL データの位置情報や、同じ意味を持つ変数の相関をとり、それぞれ四つの間で矛盾ない変数が入っているのかを調べた。また RDO と PRD について、RDO から問題なく PRD がそれぞれ四つのオブジェクトで生成されるのか、元と成る RDO の位置情報と PRD の位置情報が矛盾のない場所に変換されているのかなどを調べていった。

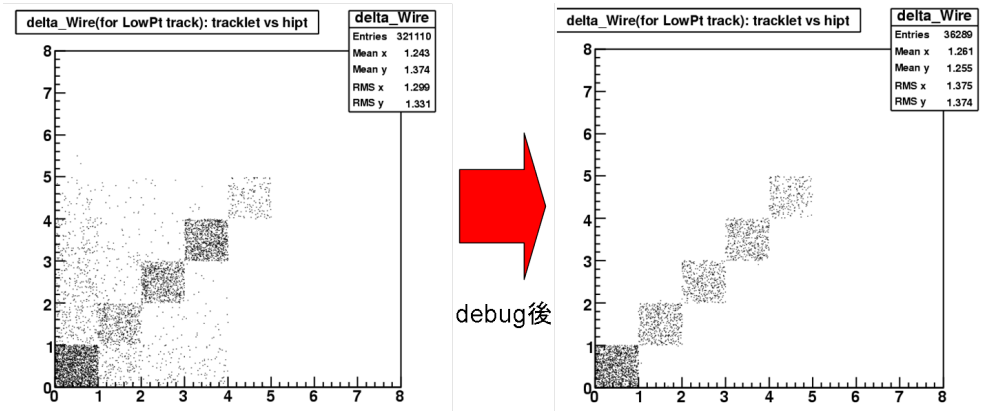


図 5.6　Tracklet と Highpt の Wiren の Delta 同士の相関

Lowpt のみを対象としたので、Delta は同一でなければならない。

2007 年の秋ごろにはまだ、BS から RDO へのコンバータに異常があったので、RDO の 4 種類と PRD の Hit データについて検証した。図 5.6 がその一例で、Tracklet と Highpt の RDO で同じ意味を持つ変数 Delta の相関を取ったときの図である。

位置情報の相関が取れている例として Hit と Tracklet のチャンネルの相関を取ったのが図 5.7 である。

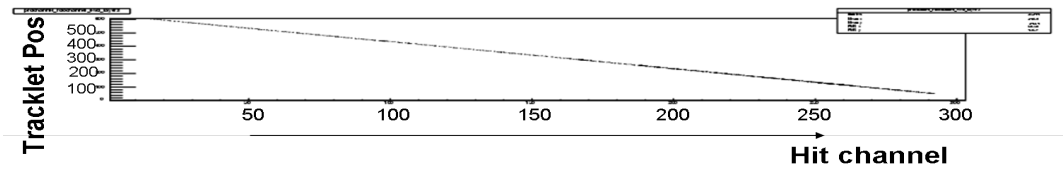


図 5.7　Layer7 の Hit チャンネルと Tracklet の position との相関

横軸が Hit のチャンネル、縦軸が Pos ビット (図 3.26) とモジュール ID から計算した 1/2 粒度のチャンネル。リードアウトとトリガ (Tracklet) で位置情報の整合性が取れている。

その後、PRD のトリガーデータを含めた整合性のチェックを行った。Coincidence の PRD は 2008 年に入ってから生成されたので、バグがかなり存在していた。その一例を図 5.8 にあげる。

現在、コンバータ自体の大きなバグは消え、コードの最適化が行われている。

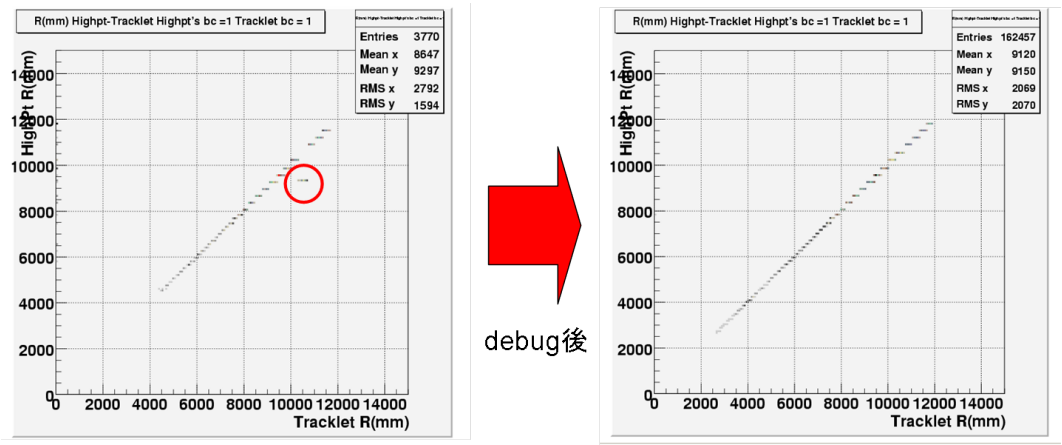


図 5.8　PRD の Tracklet と Highpt の TGC3 の R(mm) の相関。赤丸の部分がバグ

基本プロット

次に、そのトリガーデータを用いた、基本的なプロットの一例を載せる。宇宙線ランの Wire に関しての R 分布、Strip に関してのφ分布を図 5.9 に載せる。座標は ATLAS のグローバルポジションに対応している。

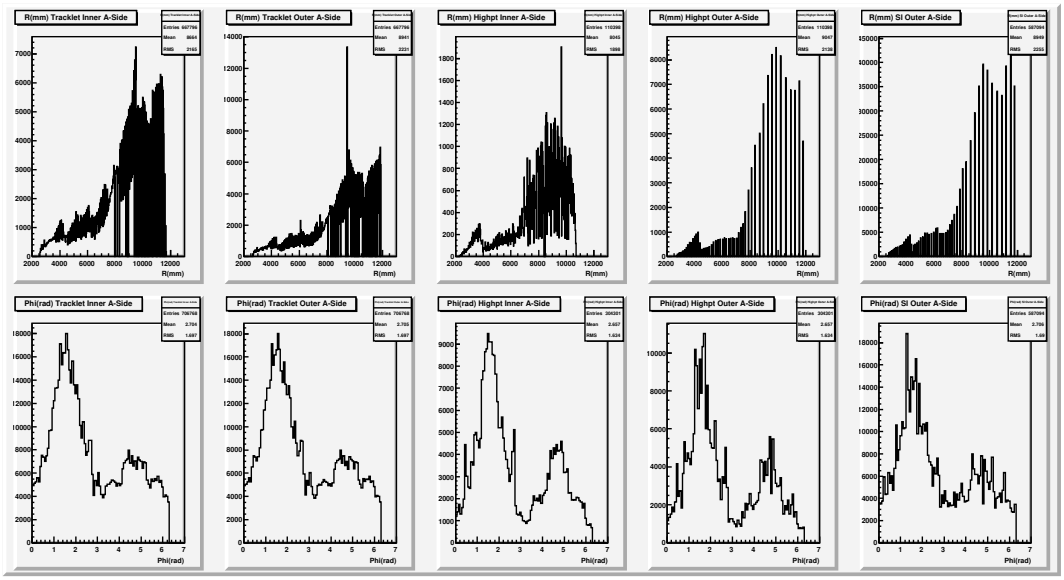


図 5.9　宇宙線のコインシデンスデータの R、ϕ 分布

上が R(Wire)、下が ϕ(Strip)。左から TGC2(Tracklet)、TGC3(Tracklet)、TGC1(Highpt)、TGC3(Highpt)、TGC3(SL) となっている。

上下方向のφの分布が多く、横方向はエントリが少ない。これは Strip トリガのために起こることで、地下は天頂方向からの宇宙線が多く (図 5.10)、横方向のチェンバは Strip のチャンネルがレイヤーによりかなり変わってしまう為である (図 5.11)。R 方向は外側ほどエントリが多い。これは外側のチャンネル幅が広いためと考えられる。

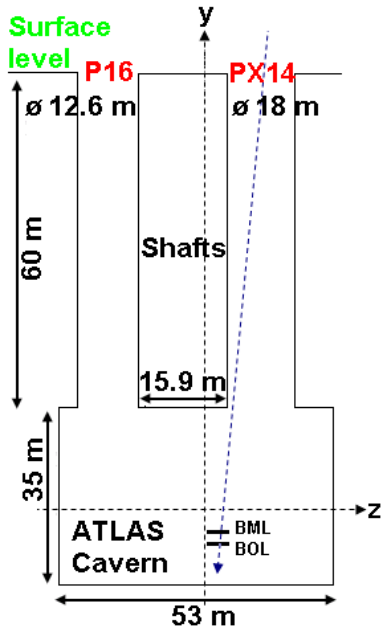


図 5.10　宇宙線の入射。天井から降り注ぐ宇宙線がほとんどである。

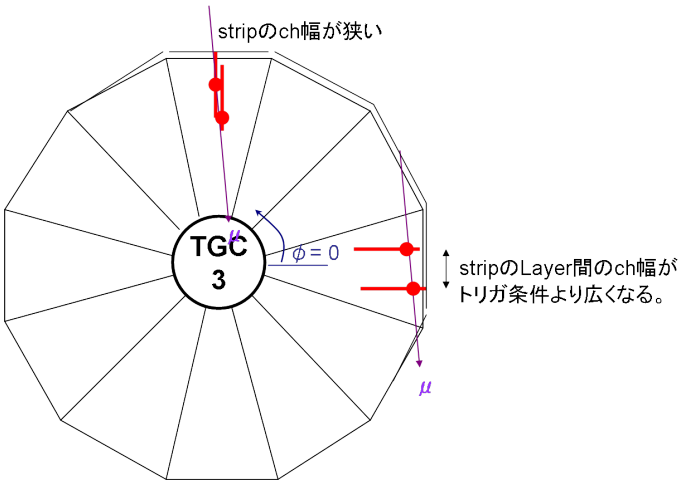


図 5.11　TGC に対する宇宙線の入射

TGC は Strip(φ方向) に関してトロイダルによる磁場の影響も少ないので Strip チャンネルに対するトリガー領域は小さい

EventDisplay

今度はあるイベントでの位置情報を示す。それが図 5.12 で、TGC3 上に ROI と Highpt の範囲、その内側に Tracklet と Hit の線が Wire、Strip 共に存在しており、データに入っている位置情報が Hit、Coincidence の両方で整合性が取れていることがわかる。

トリガーデータの階層構造

Tracklet、Highpt、SL はそれぞれ、SLB、Highpt、SL ボードからの出力をデータとしたものである。したがって SL の変数には元となる Tracklet、Highpt[*9]がいるはずである。また Highpt にも元となる Tracklet がいるはずで、Tracklet はトリガ候補となる Hit 分布がいるはずである。トリガデータ同士は PRD の変数 trackletid と side と phi が合えば、それが同一のトリガ候補となる。

それらの階層構造が取れているのかを調べたプロットの一つが図 5.13 である。これは Highpt に対して、元となる Tracklet が存在しているかどうかを調べ、

$$\frac{\text{元となる } Tracklet \text{ が存在した } Highpt \text{ の数}}{Highpt \text{ の総数}}$$

を stationPhi と R 方向 (16ch 粒度) で、場所別にプロットしたものである。stationPhi はエンドキャップで 1 から 48、Forward で 1 から 24 というチェンバをφ方向で識別する数である。

Highpt は RDO の生成時点で Endcap Strip ボード chip1 のバグの補正[*10]をするために、一つのトリガー候補から、バグの為にどちらか判断の付かない二つの候補のデータを両方生成するようになっている。その分

[*9] Highpt は TGC1 とのコインシデンスが取れた場合にのみ存在する。
[*10] TGC の説明のエレクトロニクスの Highpt ボード参照 (3,2,3 節)

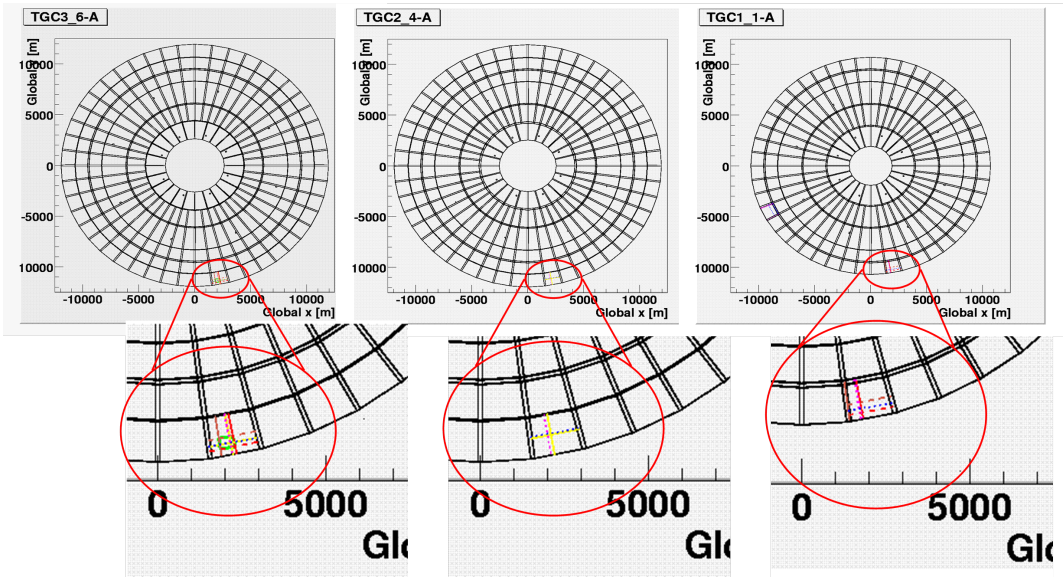


図 5.12　EventDisplay

左から TGC3、TGC2、TGC1。緑の枠が SL の ROI。赤と赤茶の線が Highpt の領域。黄色の線が Tracklet のチャンネル。青が Trigger 条件にかかった Hit 分布の Wire。ピンクが Hit 分布の Strip を示す。全て想定どうりの位置にある。

の対応が取れていないのが図に表れている。

SL オブジェクトに対して Highpt があるかどうかを調べたプロットが図 5.14 である。この最も内側のトリガ範囲で対応が取れていない場合が多い。これはハードウェア側の問題だと考えており、原因ははっきりとわかってはいない。

図 5.15 に Highpt と Tracklet の対応を載せる。この図の黒枠はラン中に対応する SSW モジュールが停止した為と考えられる。このようにトリガーデータはランのコンディションなどもサーチできる有用なソースとなっている。

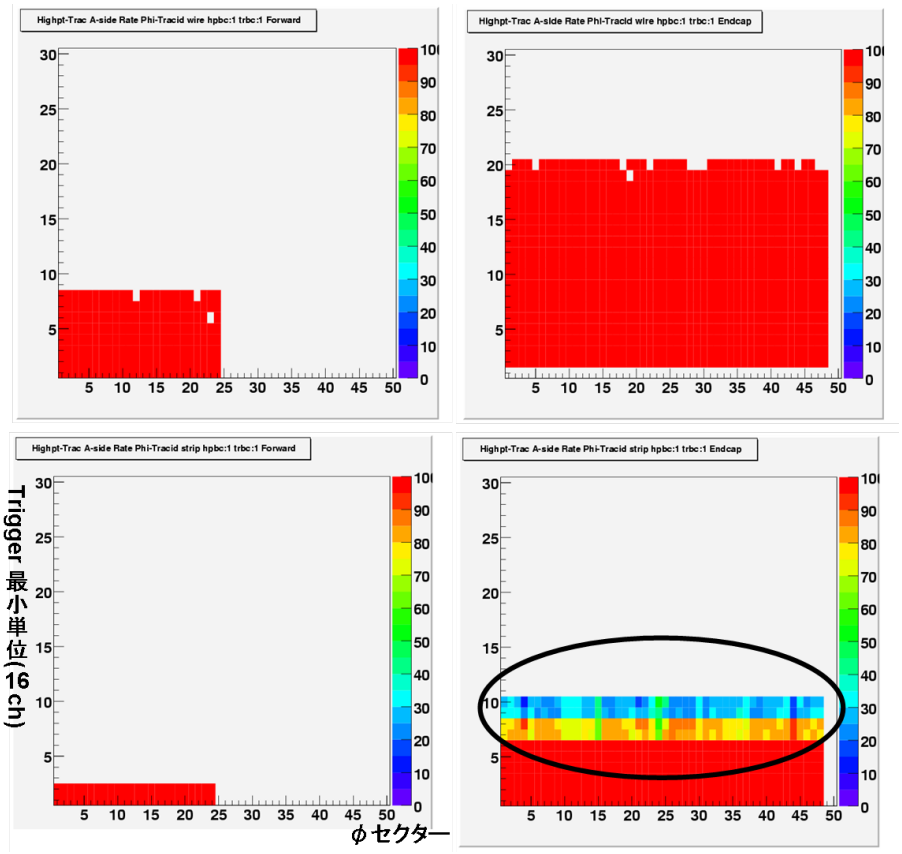


図 5.13　Highpt の元となる Tracklet があるかどうか

横軸が stationphi、縦軸は trackletid(16ch 粒度、上に行くほど SmallR となる)。左が Forward、右が Endcap、上が Wire、下が Strip となっている。黒枠で囲った部分は Highpt ボードの Endcap Strip chip1 が担当する部分。バグの為に対応が取れていない。

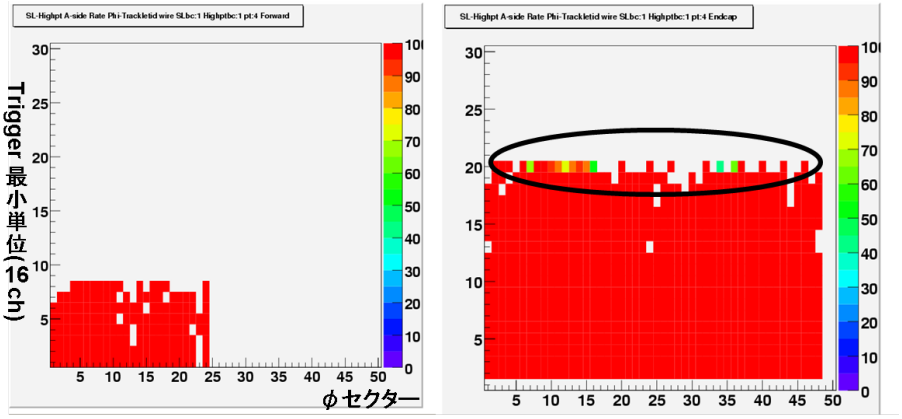


図 5.14　SL の元となる Highpt があるかどうか

Wire についてのみ。左が Forward、右が Endcap。Endcap の内側 1 ビン (16ch) 分の対応が取れていない。おそらくハードウェアによるもの。

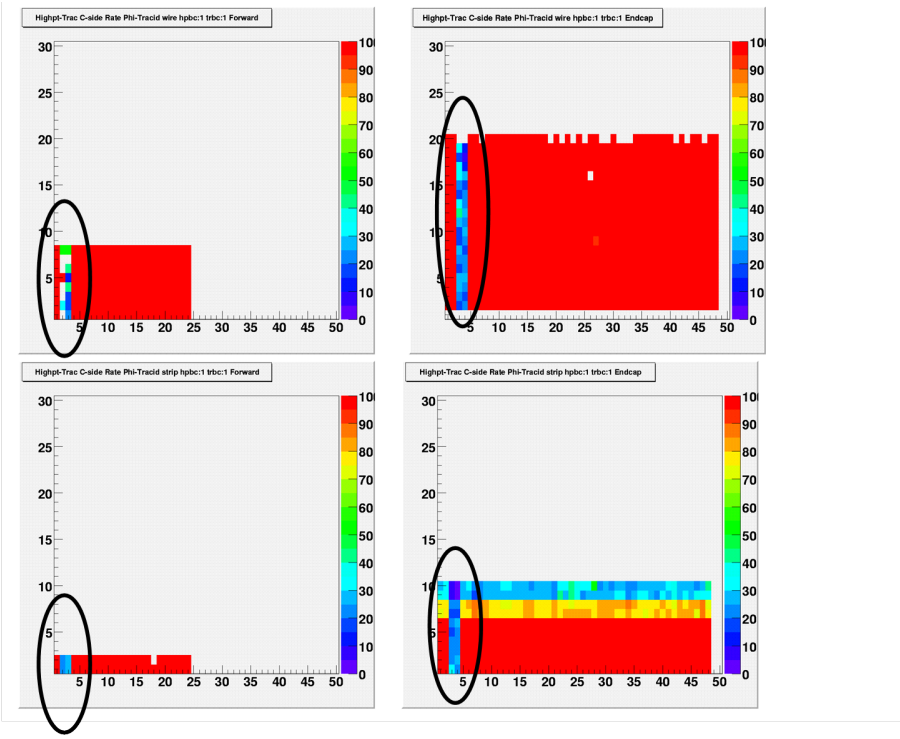


図 5.15　Highpt の元となる Tracklet がいるかどうか

黒丸で示す特定の stationPhi で Tracklet がいないことがわかる。これはランの最中に SSW モジュールにエラーが起こり、その部分のリードアウトラインのデータがなくなったためである。

上のバグの部分を除いてこのようなことを SL、Highpt、Tracklet、Hit について調べてやると、表 5 のような結果となった。

表 5　Coincidence データの階層構造割合。SL と Highpt の Strip は Strip の Highpt を要求するトリガメニューが存在しないので調べることが出来ない。

対応	基準の数 Wire	元となる数 Wire	割合
Tracklet-Hit Wire	996567	992010	0.9954273
Tracklet-Hit Strip	1053554	1011336	0.959928
Highpt-Tracklet Wire	162457	162456	0.999999
Highpt-Tracklet Strip	272563	272539	0.999911
SL-Tracklet Wire	884743	864019	0.976576
SL-Tracklet Strip	884743	850322	0.961094
SL-Highpt Wire (only Highpt)	151186	151070	0.976576

いずれのケースも 95 % 以上の割合で対応が取れていて、大きなバグが存在しないことを示している。では、なぜ 100 % にならないのだろうか。原因としては

- タイミングがずれて、対応するデータが Next や Previous に入ってしまっている
- ハードウェアの段階で chip に焼く bit ファイルなどにバグがある
- ハードウェアの想定している設定とソフトウェアの想定している設定が合っていない
- ソフトウェアのコンバータのミス

が考えられる。

タイミングは Hit、Tracklet に対してと Highpt、SL に関しては同じ SLB チップから出力されるので、タイミングがずれることはないので、それで対応が取れていないのはタイミングの問題ではない。

ソフトウェアのコンバータ部分本体については、バグはないだろうと考えているが、bit 化けなどのデータ自体に問題があり、BS2RDO や RDO2PRD などのコンバージョンが失敗した場合の影響があると考えられる。

また、SL の bit ファイルなどは神戸大学の担当者が、開発を進めており、これから、始まるランでデバッグが進められていくと考えられる。

6 宇宙線を用いた検出器の解析

実際にコミッショニングが行われたうえでのデータで TGC に対する解析を行った。使用した Run を表 6 に載せる。Gas は (CO_2:n-pentan=55:45) で TGC の Trigger 条件は以下の三種類である。
pt1(wire 3outof4 strip 3outof4)、pt4(wire 3outof4 & 2outof3 strip 3outof4 $\Delta R < 10$)、pt5(wire 3outof4 & 2outof3 strip 3outof4 $\Delta R > 10$)

表 6　解析に使用した RunNumber とその時のコンディション

Run	HV	Threshold(mV)		Gate(ns)		Start Time	End Time	気圧
Number	(V)	wire	strip	wire	strip	year-mon-day h:m:s	year-mon-day h:m:s	hPa
90413	2800	80	80	30	45	2008-09-30 20:19:11	2008-10-01 05:15:30	972.9
90525	2750	80	80	35	45	2008-10-01 22:27:35	2008-10-02 08:52:14	969.1
91636	2850	80	80	35	45	2008-10-14 12:37:32	2008-10-14 17:42:29	980.0
91800	2650	80	80	35	45	2008-10-15 18:44:00	2008-10-16 00:07:07	976.6

Run90413、90525、91636、91800 の TGC チェンバーの Efficiency を比較する。HV は大気圧にほぼ反比例でゲインが変わるので (気圧が高いほど落ちる)[23]、気圧が 1000hPa の場合の設定値を仮定して、かけるべき電圧を算出する。

Efficiency の求め方は図 6.1 のようにする。

調べたい層以外の Hit を要求することで、Trigger 条件は満たされるので、あとはその層が反応するかのみであり、トリガーに対するバイアスはかからないはずである。分子と分母は stationEta と stationPhi ごとで数えている。stationEta は Endcap のチェンバを 1 から始まり内側から数えたもので、stationPhi はエンドキャップで 1 から 48、Forward で 1 から 24 というチェンバをφ方向で識別する数である。

図 6.2 は Doublet の Wire の Efficiency を R 差分が width 以下を条件として求めたものである。これをみると Eta4(E2) の Efficiency がおかしいことがわかる。

図 6.3 にその部分 (Eta4、Layer6) の R 差分の分布を載せる。これを見ると-80 あたりにピークがあることが分かるが、このチェンバのチャンネル幅は約 40mm となっており、40mm のピークより 80mm のピークが高いというのは異常である。

これは Eta4 のチャンネルがずれるという AtlasGeometry のバグによって起こっており、現在修正するように求めている。

上で述べたバグが存在するため、Hit との条件を以下のようにして、Efficiency を求めた。

- $|r(Hit) - r(Coincidence)| < 2 \times width$
- $|phi(Hit) - phi(Tracklet)| < 0.009$
- $|phi(Hit) - phi(Highpt)| < 0.07$

Strip に対する数字は Endcap に対して少しだけ条件が甘くなるが、

$$\frac{2\pi}{24(Forward\ チェンバ数) \times 32(ch\ 数)} \simeq 0.009(Tracklet) \qquad \frac{2\pi \times 8}{24 \times 32} \simeq 0.07(Highpt)$$

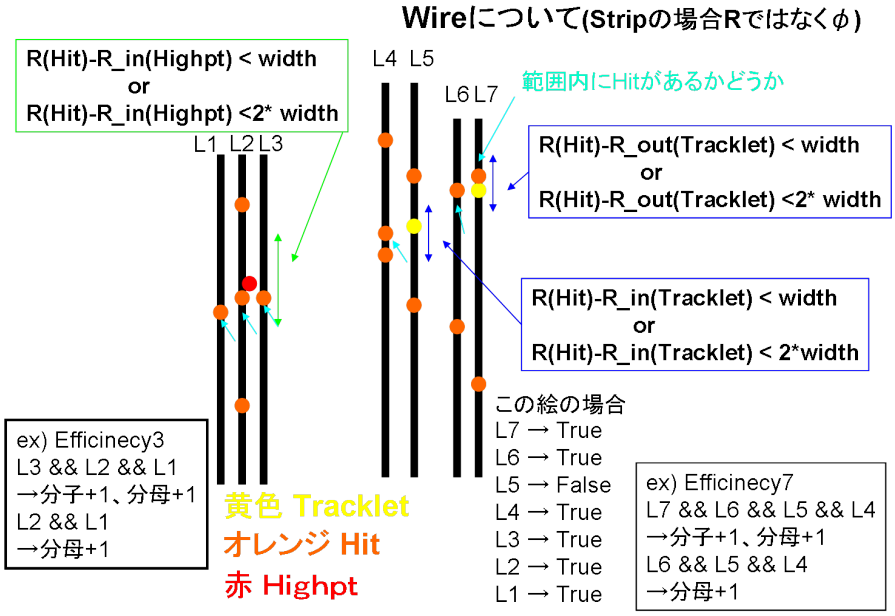


図 6.1　Efficinecy の求め方

Doublet は Tracklet を用いて、Triplet は Highpt データを用いて求める。それらのデータからレイヤーごとに width か 2×widht 以内に Hit があるか調べる。Hit があるなしによって分子、分母を計算する

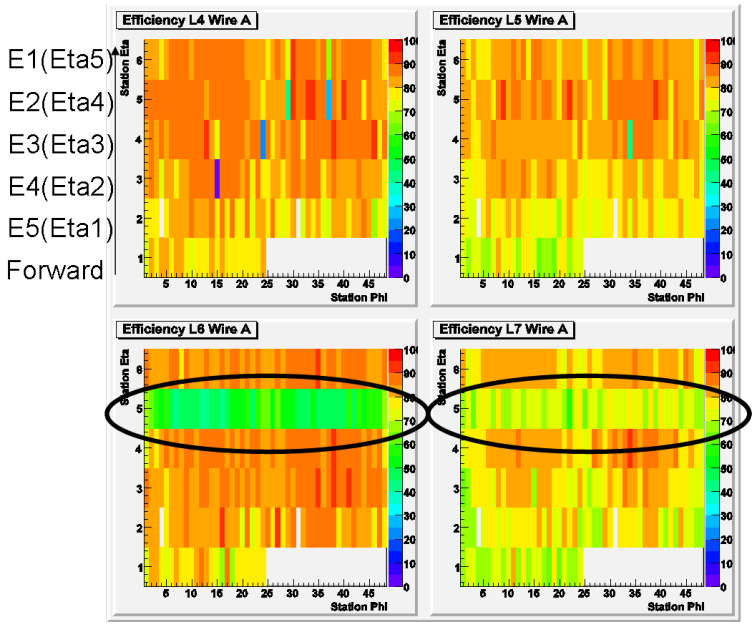


図 6.2　Efficiency Doublet Wire ラン 90413 2.8kV R(Hit)-R(Tracklet) < width の場合

黒枠で囲ったように Layer6 と Layer7 の E2(Eta4) チェンバで Efficiency が低い。これは AtlasGeometry にバグがあり、この領域で 1ch ずれているため。また、チェンバの数え方はエレキとオフラインソフトウェアで異なる。エレキは外側から数えるが、オフラインソフトでは内側から数える。

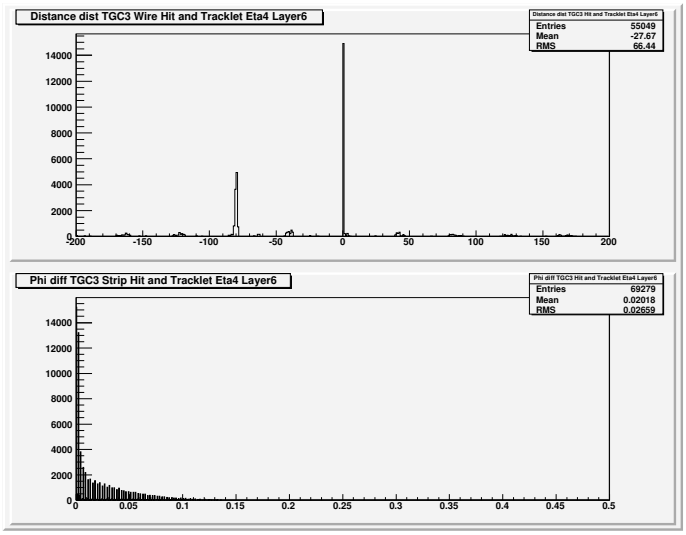


図 6.3　Layer6 での Eta4(E2) の Tracklet と Hit の R(mm) 差分分布 (上)、Phi(rad) の差分分布 (下)
-40 あたりのピークより-80 の方に立ってしまっている。1ch の幅はこのチェンバでは 40mm で、1 粒子による連続チャンネル Hit のエントリは-40 より-80 が立つことはありえない。

として決定した。

結果として、四つのラン全ての Efficiency を横軸に stationPhi、縦軸にチェンバーで図 6.4 - 6.19 に載せる。HV が低いラン 91800 に関して Efficiency が取れていないことが分かり、HV が 2.8kV を超えるラン 90413 と 91636 に関して高いことが分かる。

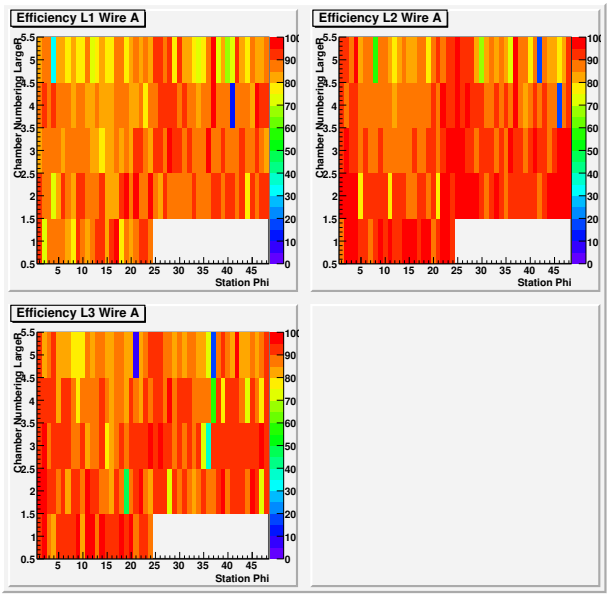


図 6.4　Efficiency Triplet Wire ラン 90413 2.8kV
横軸 stationPhi、縦軸 R 方向のチェンバの番号。Forward を 1 として外側へ向かって数えていく。90% 近くの Efficiency で、四つのランのうち最も高い傾向が見られる。

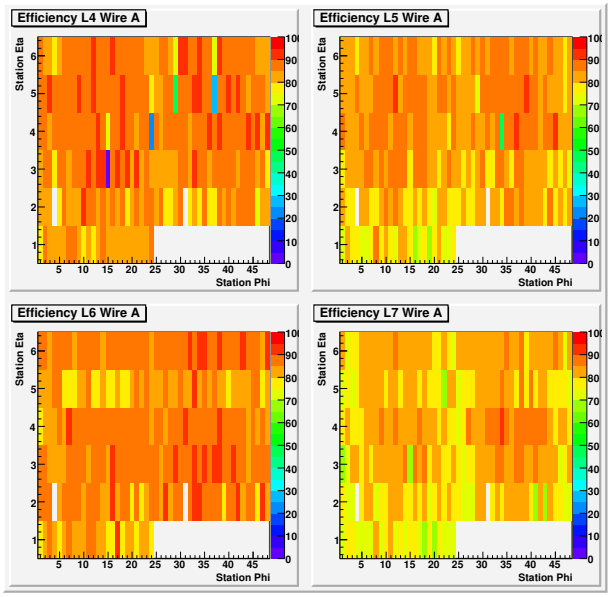


図 6.5　Efficiency Doublet Wire ラン 90413 2.8kV
横軸 stationPhi、縦軸 R 方向のチェンバの番号。Forward を 1 として外側へ向かって数えていく。80% 強の Efficiency で、四つのランのうち最も高い傾向が見られる。また、横方向の Phi で比較的 Efficinecy が落ちている。

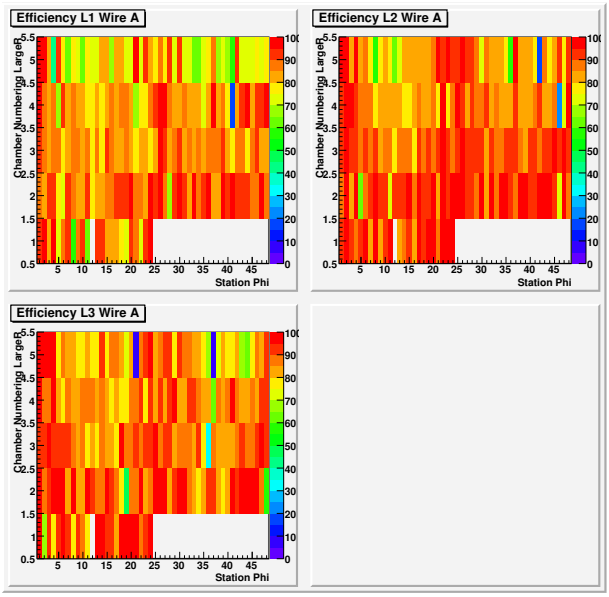


図 6.6　Efficiency Triplet Wire ラン 90525 2.75kV
横軸 stationPhi、縦軸 R 方向のチェンバの番号。Forward を 1 として外側へ向かって数えていく。2.8kV や 2.85kV と比べると若干落ちる。

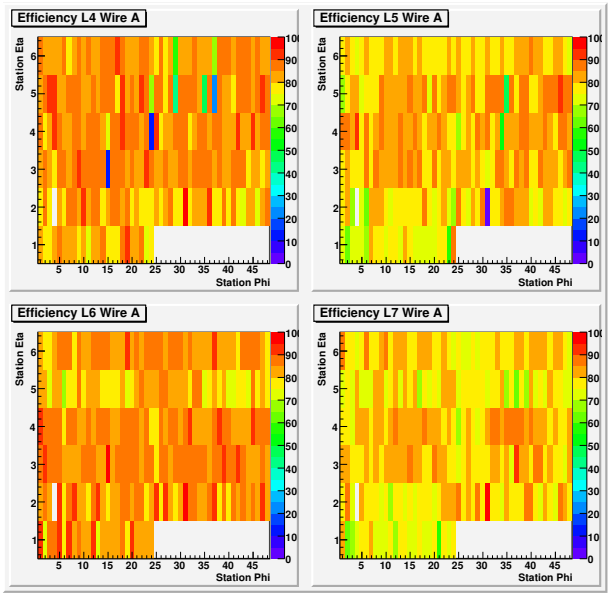


図 6.7　Efficiency Doublet Wire ラン 90525 2.75kV
横軸 stationPhi、縦軸 R 方向のチェンバの番号。Forward を 1 として外側へ向かって数えていく。2.8kV や 2.85kV と比べると若干落ちる。

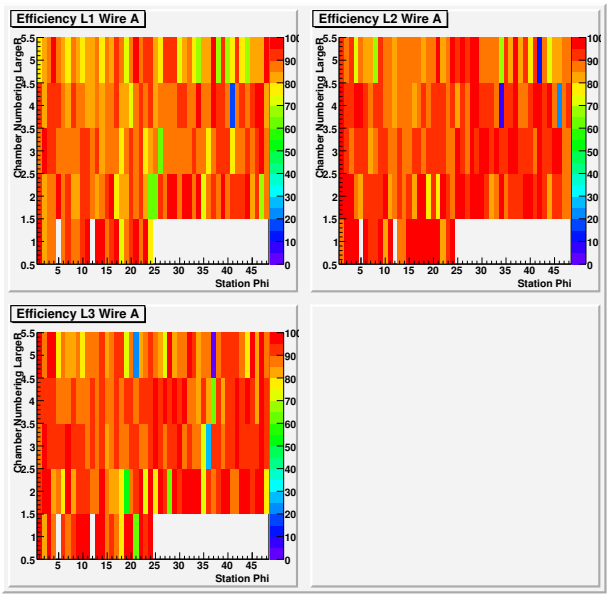


図 6.8　Efficiency Triplet Wire ラン 91636 2.85kV
横軸 stationPhi、縦軸 R 方向のチェンバの番号。Forward を 1 として外側へ向かって数えていく。2.8kV と同等の Efficiency を得た。

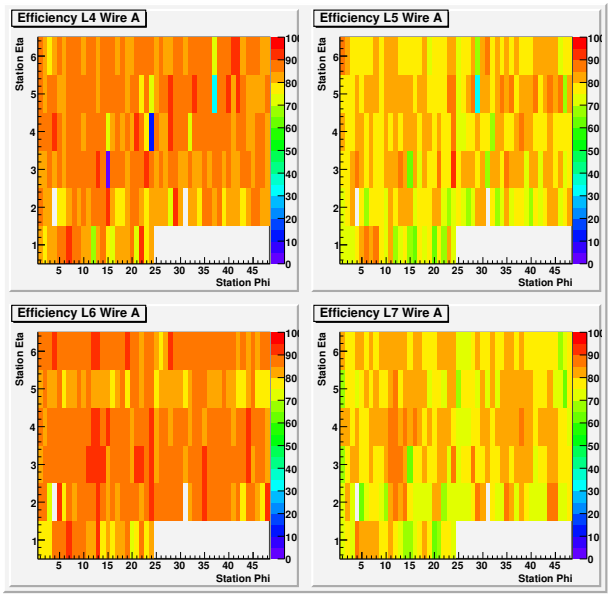


図 6.9　Efficiency Doublet Wire ラン 91636 2.85kV
横軸 stationPhi、縦軸 R 方向のチェンバの番号。Forward を 1 として外側へ向かって数えていく。2.8kV よりほんの少しだけ落ちているように見える。

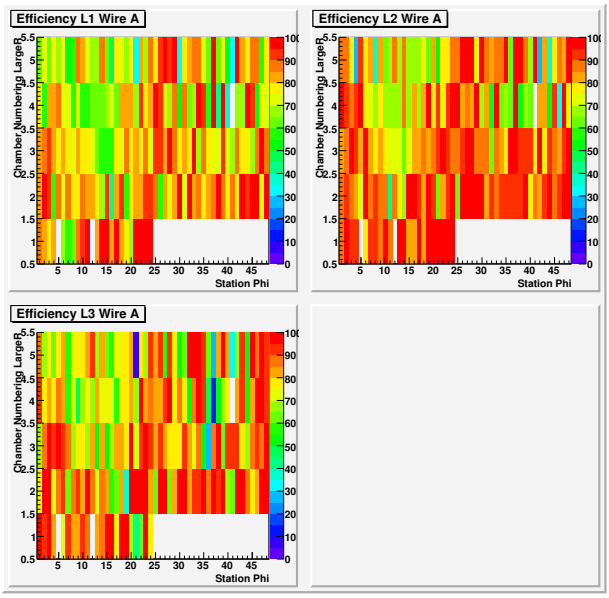


図 6.10　Efficiency Triplet Wire ラン 91800 2.65kV
横軸 stationPhi、縦軸 R 方向のチェンバの番号。Forward を 1 として外側へ向かって数えていく。目に見えて低いのが分かる。

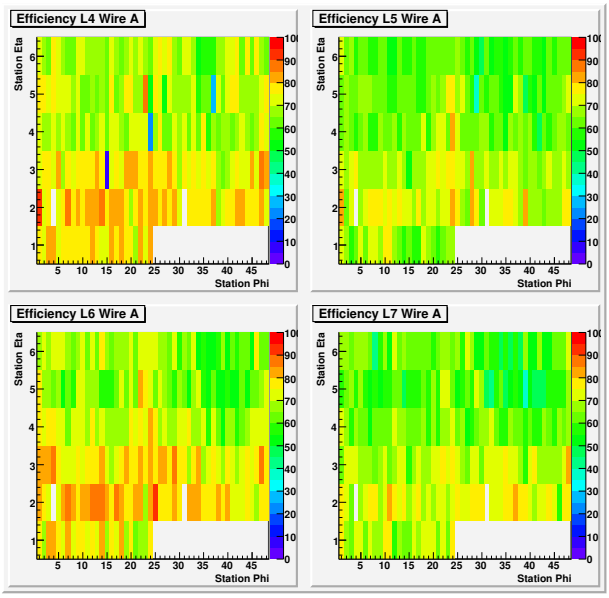


図 6.11　Efficiency Doublet Wire ラン 91800 2.65kV
横軸 stationPhi、縦軸 R 方向のチェンバの番号。Forward を 1 として外側へ向かって数えていく。目に見えて低いのが分かる。

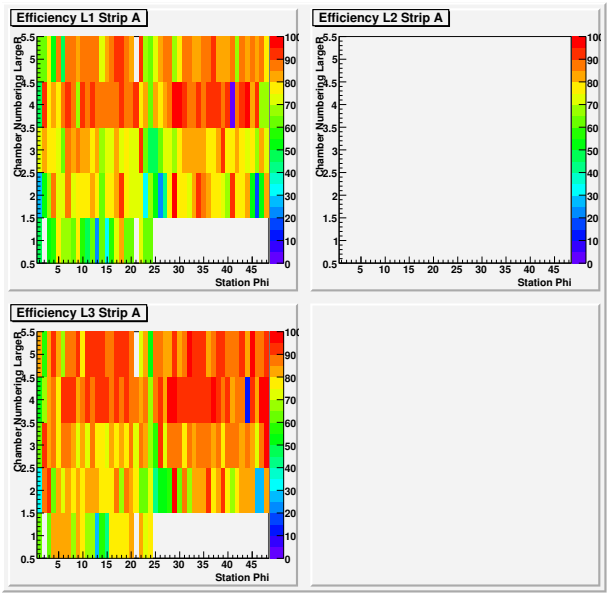


図 6.12　Efficiency Triplet Strip ラン 90413 2.8kV
横軸 stationPhi、縦軸 R 方向のチェンバの番号。Forward を 1 として外側へ向かって数えていく。Triplet の Strip は 2 層しかないので、若干信頼性がないが、2.75kV や 2.65kV よりは高い値を示している。

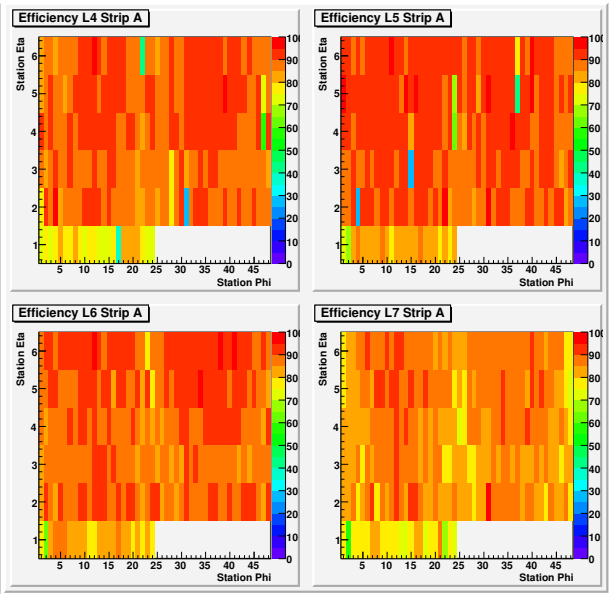


図 6.13　Efficiency Doublet Strip ラン 90413 2.8kV
横軸 stationPhi、縦軸 R 方向のチェンバの番号。Forward を 1 として外側へ向かって数えていく。2.65kV や 2.75kV より高い。また、Endcap に対して多少甘い条件でとったので、Endcap が高い。

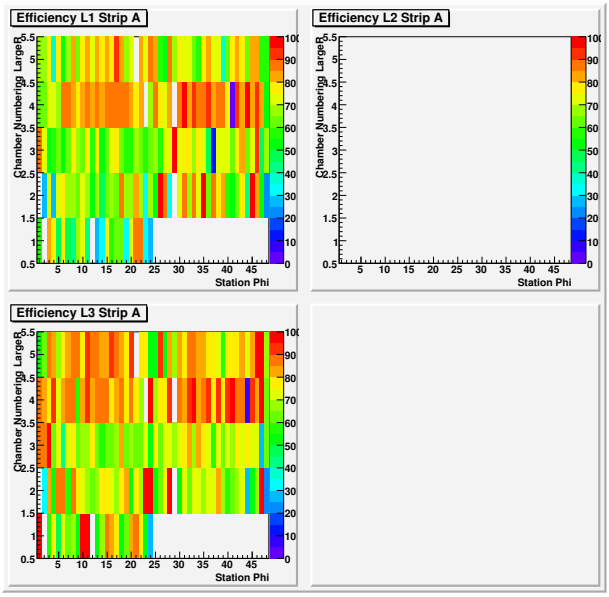


図 6.14　Efficiency Triplet Strip ラン 90525 2.75kV
横軸 stationPhi、縦軸 R 方向のチェンバの番号。Forward を 1 として外側へ向かって数えていく。Triplet の Strip は 2 層しかないので、若干信頼性がないが、2.8kV や 2.85kV よりは低い値を示している。

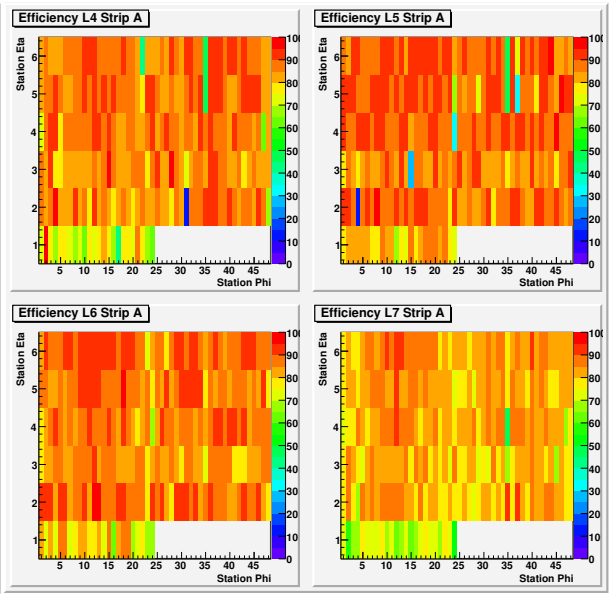


図 6.15　Efficiency Doublet Strip ラン 90525 2.75kV
横軸 stationPhi、縦軸 R 方向のチェンバの番号。Forward を 1 として外側へ向かって数えていく。2.75kV や 2.85kV より低い。また、Endcap に対して多少甘い条件でとったので、Endcap が高い。

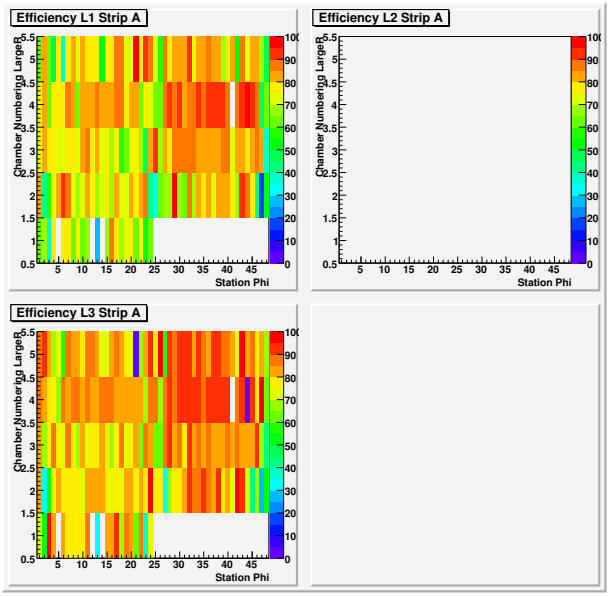


図 6.16　Efficiency Triplet Strip ラン 91636 2.85kV
横軸 stationPhi、縦軸 R 方向のチェンバの番号。Forward を 1 として外側へ向かって数えていく。Triplet の Strip は 2 層しかないので、若干信頼性がないが、2.8kV と並ぶ高い値を示している。

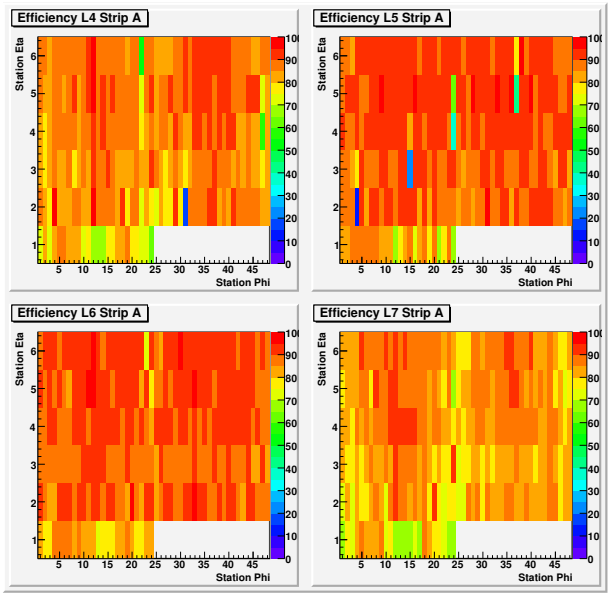


図 6.17　Efficiency Doublet Strip ラン 91636 2.85kV
横軸 stationPhi、縦軸 R 方向のチェンバの番号。Forward を 1 として外側へ向かって数えていく。2.8kV と並ぶ高さ。また、Endcap に対して多少甘い条件でとったので、Endcap が高い。

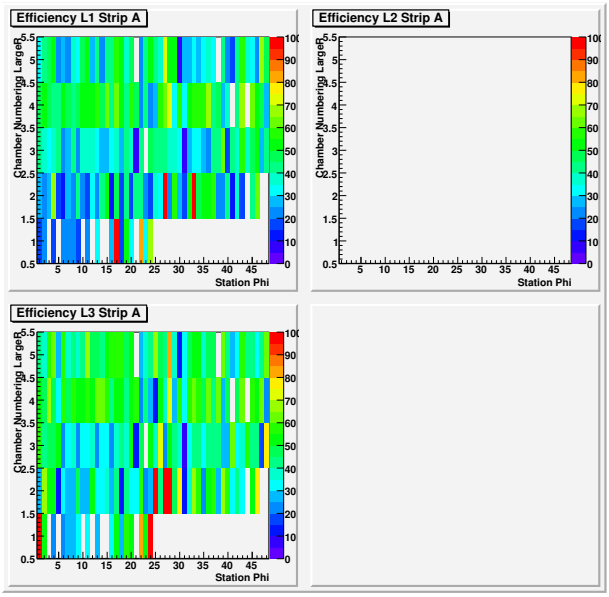


図 6.18　Efficiency Triplet Strip ラン 91800 2.65kV
横軸 stationPhi、縦軸 R 方向のチェンバの番号。Forward を 1 として外側へ向かって数えていく。目に見えて低いのが分かる。

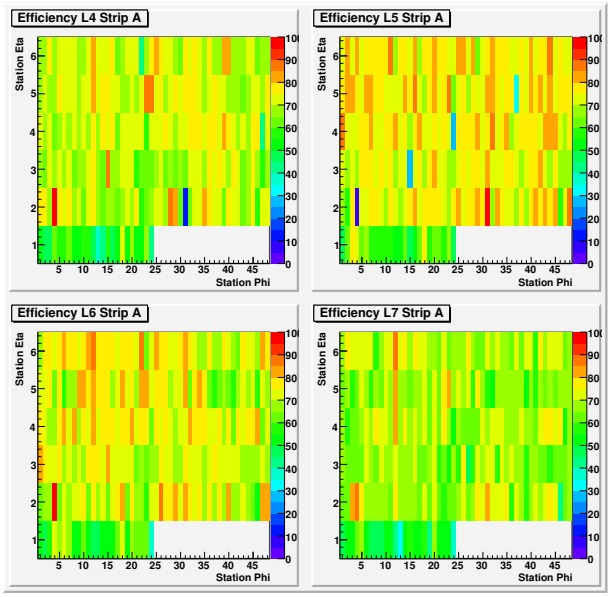


図 6.19　Efficiency Doublet Strip ラン 91800 2.65kV
横軸 stationPhi、縦軸 R 方向のチェンバの番号。Forward を 1 として外側へ向かって数えていく。目に見えて低いのが分かる。

上と同じランに関して、HV との関係を見るために、各レイヤー別に各チェンバでの Efficiency を横軸 HV、縦軸 Efficiency で作成した。その内の Layer1 と Layer4 を例として載せる (図 6.20-6.23)。この図によってチェンバの特性の違いが分かる。Wire の Doublet に関してはチェンバによる違いが小さいが、他は 10% ぐらい Efficiency が変わってしまっていることが分かる。

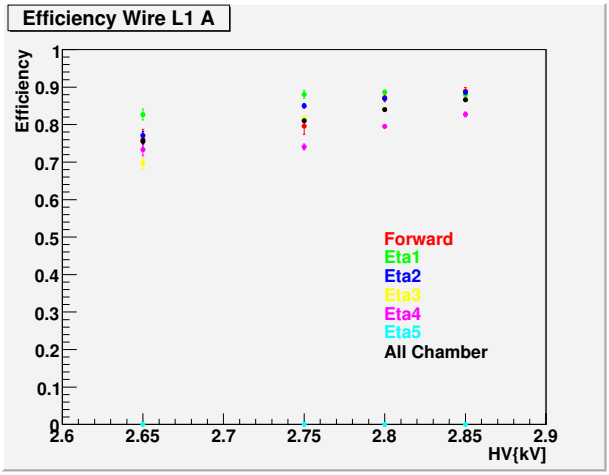


図 6.20　Efficiency vs HV Wire Layer1
HV に多少敏感なのもいれば低いのもいる。2.8kV や 2.85kV で Efficiency8% ぐらいの差がある。

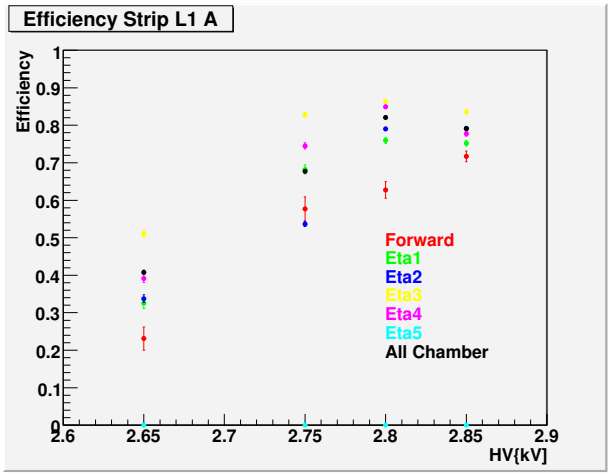


図 6.21　Efficiency vs HV Strip Layer1
Wire と違いかなり HV に影響されている。どの HV でも最も高いチェンバと低いチェンバで 10% 以上異なる。

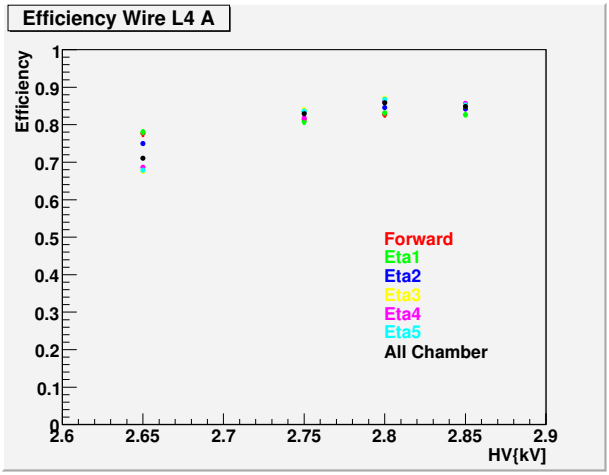


図 6.22　Efficiency vs HV Wire Layer4
比較的チェンバによる差が小さい。全てのチェンバで 2.8kV で最も Efficiency が高い。

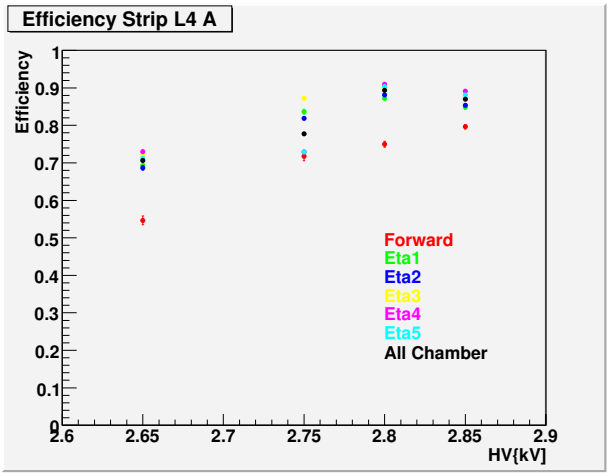


図 6.23　Efficiency vs HV Strip Layer4
L1 同様 HV に対しての影響が大きい。また、最も高いチェンバと低いチェンバでどの HV でも 10% 以上異なる。

同じランで図 6.24-6.25 はチェンバ各々について、レイヤー別に横軸 HV、縦軸 Efficiency を描写したプロットを載せる。レイヤーによっても Efficiency が変わることが多いのがわかる。

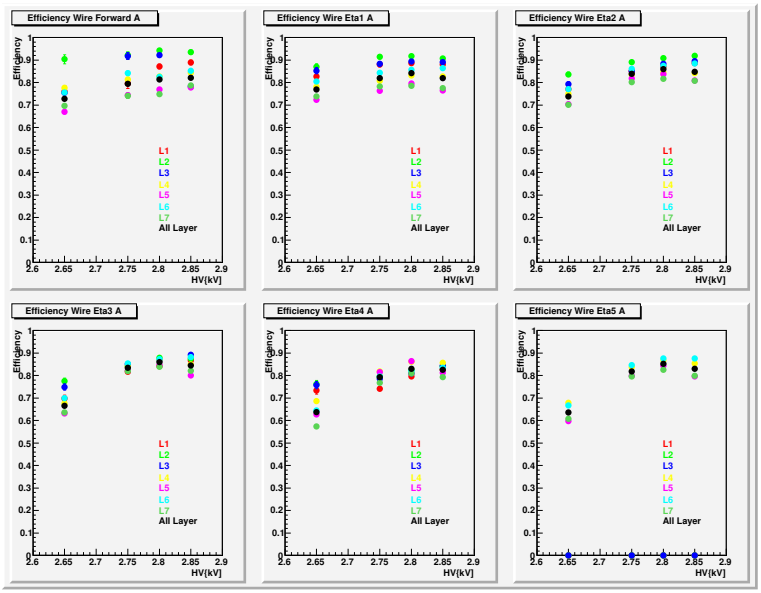


図 6.24　Efficiency vs HV チェンバー別 Wire

左上から Forward、その右が Eta1、Eta2 で下が Eta3、Eta4、Eta5 という並びになっている。レイヤーによっても特性が異なる。

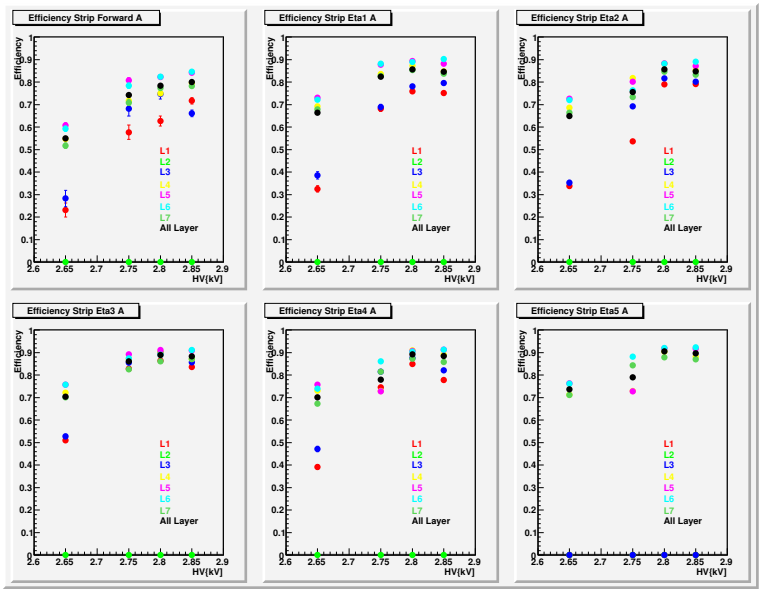


図 6.25　Efficiency vs HV チェンバー別 Strip

左上から Forward、その右が Eta1、Eta2 で下が Eta3、Eta4、Eta5 という並びになっている。レイヤーによっても特性が異なる。

チェンバにより挙動が異なるが、2.8kV が最も Efficiency が高く、それから 2.85kV にかけて安定して高い

値を示している。設定値の 2.8kV は 1000hPa で

$$HV(1000hPa) = 2.8 \times \frac{1000}{972.9} = 2.878$$

となり、1000hPa という高い気圧になる場合を考えたら 2.9kV ぐらいの HV をかけたほうがよいことが分かる。

表 7　各レイヤーでの Efficiency A-Side

Run (wire or strip)	HV (V)	L1	L2	L3	L4	L5	L6	L7	doublet 平均	triplet 平均
91636 wire	2850	0.87	0.89	0.87	0.85	0.80	0.87	0.80	0.83	0.88
90413 wire	2800	0.84	0.88	0.86	0.86	0.84	0.86	0.82	0.84	0.86
90525 wire	2750	0.81	0.85	0.84	0.83	0.81	0.84	0.79	0.82	0.83
91800 wire	2650	0.76	0.82	0.78	0.71	0.65	0.70	0.64	0.68	0.79
91636 strip	2850	0.79	-	0.81	0.87	0.90	0.90	0.85	0.88	0.80
90413 strip	2800	0.82	-	0.85	0.89	0.90	0.90	0.86	0.89	0.84
90525 strip	2750	0.68	-	0.77	0.78	0.78	0.84	0.80	0.80	0.72
91800 strip	2650	0.41	-	0.45	0.71	0.75	0.74	0.68	0.72	0.43

各レイヤーでのチェンバの区別なし、つまりレイヤー一枚を大きなチェンバと考えた場合の Efficiency は表 7 のようになっている。

TGC は Doublet のトリガーが取れれば、データを取得するので Doublet 全体の Efficiency を求めてみる。各レイヤーの Efficiency を一律で ϵ とすると、3outof4 の Efficiency は

$$(1-\epsilon) \times_4 C_1 \times \epsilon^3 + \epsilon^4$$

となる。

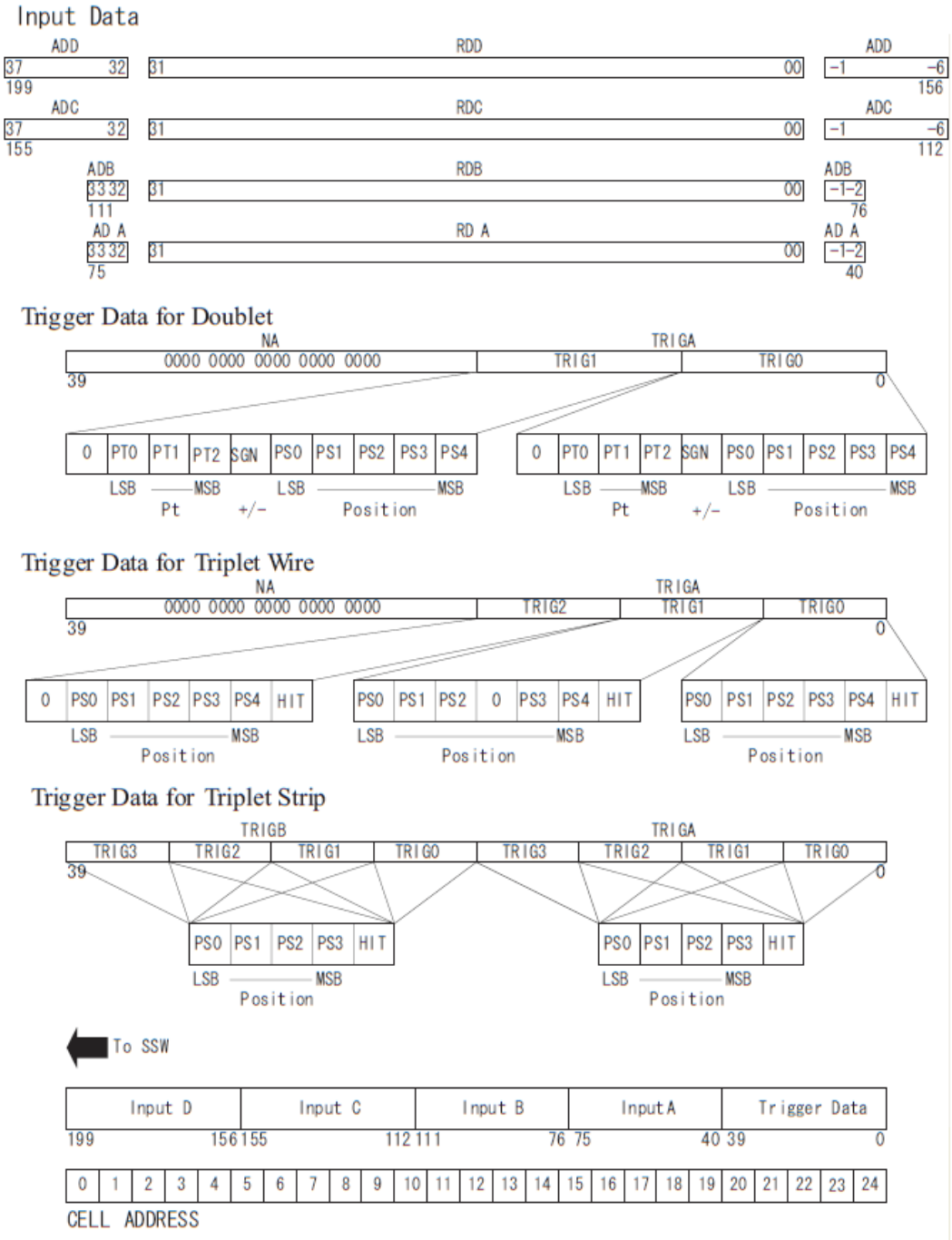
Efficinecy を 2850kV 時の Wire の平均 83% と Strip の平均 88% を代入するとそれぞれ、86% と 93% となり、要求されている 99% には達しない。しかし、このランではケーブリングミスが幾つか存在しており、現在はもう修正済みとなっているので今後の TGC のパフォーマンスはこの値より上になるはずである。

7　まとめ

ATLAS のミューオントリガーシステムは、ハードウェアの段階で Pt を判別し、ラベルをつけて送り出すという画期的なシステムとなっている。そのシステムのエンドキャップ部分を担う TGC システムは、ボードの検査から、地上動作試験、地下での宇宙線試験を通して、システムを稼動して行き、2008 年に全てのセクターが動作、シングルビームのデータも収集でき、Muon トリガーとして機能していることを証明した。

TGC のオフラインソフトウェアもケーブリングからコンバータまで、大きなバグも消え、トリガー、リードアウトの各種情報を扱うことが出来、それを利用した、モニタリング、デバッグ、解析が 2009 年の来るべき 14TeV のビーム衝突に万全の体制で望むべく行われている。

付録 A　PSB Data Format



図付録 A.1

付録 B　SSW Data Format

Version 01 of the TGC Front End link data format

Event Header 000　　Now, Record Type=01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000			RecType		SSWID				RX mask pattern (1=enabled, 0=disabled)																							

Record Type (RecType) is 01 in this format version, hard-wired in FPGA
SSWID is arbitrarily set by a dip-switch on each SSW board

SLB header 010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
010			SLBID					0	BCmap			Mod Type			0	L1ID						BCID									

BCmap shows 3BC data lines taken by RX. 3bit shows {next, current, previous} events. 1=adopted. 0=discarded
SLBID, Mod Type, L1ID and BCID are all SLB's data. See SLB documents.

SLB header 011 –0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
011			0	0	RXID					0	RX FIFO status					SLB-OVF								RX-OVF							

RXID is RX identified number from 0 to 22.
RX FIFO status tells what amount of data are stored in RX-FIFO then.
SLB-OVF is SLB's data. See SLB documents.
RX-OVF is RX-FIFO overflow counter. This tells the snapshot value when this word is sent from RX to TX.

SLB trailer 011 –1　　This word appears after SLB data words only when there is an error

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
011			1	SEU	OVF	LVDSlnk		RX error state							

LVDSlnk=LVDS links status. 2bits are {now,old}. 1=Not linked. 0=Linked.
SEU = SLB SEU flag. See SLB documents.
OVF = RX-FIFO overflow flag. If OVF=1, some overflows have happened in this RX data.

SLB data 100, 101, 110

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
100			cell address					cell bitmap							
101			cell address					cell bitmap							
110			cell address					cell bitmap							

In any order:
Cell data for Current BC data
Cell data for Previous BC data
Cell data for Next BC data

PAD word 110

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
110			11111					0							

i.e. 0xDF00

Event Trailer 111

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111			0x1CA									Glnk	T1C	NRC	T2C	XOR check sum															

Glnk = Glink TX status. "Locked" signal of Glink Tx. 1=Not locked. 0=Locked
T1C = Timeout1_count_frag. Time-out to collect the event fragment from all the enabled input ports.
NRC = Nores_count_flag. No response from RX FIFO of "enabled" (not masked) input port.
T2C = Timeout2_count_flag. Time-out to collect the event fragment from each enabled RX FIFO.
These three flags are reset at every event.

The XOR operation includes the first word(16bits) of the event header through the first word of the event trailer.
When the result is XOR'ed with the XOR checksum word, the result becomes zero. (the XOR does not include the 0x0B0F an 0x0E0F framing words)

Framing

Each event is preceeded by the 32-bit word 0x0000'0B0F and followed by the 32-bit word 0x0E0F'0000, both words are sent in Glink control mode.

図付録 B.1

付録 C　ROD Output Data Format

	Data word				Comments
	31..24	23..16	15..8	7..0	
Frame	x'B0F0xxxx'				event frame word (control mode word)
Hdr 0	x'EE1234EE'				start of header marker for ROD data
Hdr 1	***reserved***	***reserved***	header size = 9		words (excluding x'B0F0xxxx' word)
Hdr 2	ATLAS format version=3.0		TGC format version=3.0		i.e.: ATLAS=0x03'00, TGC=0x03'00
Hdr 3	0	x'67' or x'68'	0	octant[7..0]	source id: x'67'/x'68' = A/C endcap;
Hdr 4	Run type	Run number			
Hdr 5	Level-1 ID				High byte is Extended Level-1 ID
Hdr 6	***reserved***	***reserved***	Bunch crossing ID[11..0]		
Hdr 7	***reserved***	***reserved***	***reserved***	Trigger type	
Hdr 8	Detector event type				not used yet
Status	First status word: specific | generic				≠0: event is ***not*** OK.
Status	TGC ROD event status				
Status	ROD VME filter bits		Star Switch timeout status		one bit per SSW; 1 means accepted or timeout
Status	Local status word		presence		Presence indicates which of the following fragments are present[a].
Status	orbit count				orbit count; zero for first L1AID.[b]
Data	Fragment ID	raw data word count			fragment ID = 1, length in words
Data	Fragment ID	"readout format" hit data word count			fragment ID = 2, length in words
Data	Fragment ID	"readout format" tracklet data word count ("tracklet" = 3/4 or 2/3 coincidence)			fragment ID = 3, length in words
Data	Fragment ID	"chamber format" hit data word count			fragment ID = 4, length in words
Data	Fragment ID	"chamber format" tracklet data word count			fragment ID = 5, length in words
Data	Fragment ID	HipT output word count			fragment ID = 8, length in words
Data	Fragment ID	Sector Logic word count			fragment ID = 9, length in words
Data	raw data, hit, tracklet, sector logic, etc. fragments, in the order of the word counts.				
Data	. . .				
Data	last raw data, hit or tracklet word				
Trail 0	number of status elements = 5				
Trail 1	number of data elements				
Trail 2	Status block position = 0, i.e. data follows status				
Frame	x'E0F0xxxx'				event frame word (control mode word)

a. The number of fragment ID | WC words and fragments is equal to the number of **Hi** bits in this pattern.
b. 32 bits give >100 hrs

図付録 C.1

付録 D　ROI Numbering

Sub Sector Numbering (ROI numbering)

Forward
all sector

η \ φ	3	2	1	0
	7	6	5	4
	11	10	9	8
	15	14	13	12
	19	18	17	16
	23	22	21	20
	27	26	25	24
	31	30	29	28
	35	34	33	32
	39	38	37	36
	43	42	41	40
	47	46	45	44
	51	50	49	48
	55	54	53	52
	59	58	57	56
16	63	62	61	60

Endcap
odd number sector (sector 1,3,...)

η \ φ	3	2	1	0
	7	6	5	4
	11	10	9	8
	15	14	13	12
	19	18	17	16
	23	22	21	20
	27	26	25	24
	31	30	29	28
	35	34	33	32
	39	38	37	36
	43	42	41	40
	47	46	45	44
	51	50	49	48
	55	54	53	52
	59	58	57	56
	63	62	61	60
	67	66	65	64
	71	70	69	68
	75	74	73	72
	79	78	77	76
	83	82	81	80
	87	86	85	84
	91	90	89	88
	95	94	93	92
	99	98	97	96
	103	102	101	100
	107	106	105	104
	111	110	109	108
	115	114	113	112
	119	118	117	116
	123	122	121	120
	127	126	125	124
	131	130	129	128
	135	134	133	132
	139	138	137	136
	143	142	141	140
37	147	146	145	144

Endcap
even number sector (sector 0,2,...)

η \ φ	0	1	2	3
	4	5	6	7
	8	9	10	11
	12	13	14	15
	16	17	18	19
	20	21	22	23
	24	25	26	27
	28	29	30	31
	32	33	34	35
	36	37	38	39
	40	41	42	43
	44	45	46	47
	48	49	50	51
	52	53	54	55
	56	57	58	59
	60	61	62	63
	64	65	66	67
	68	69	70	71
	72	73	74	75
	76	77	78	79
	80	81	82	83
	84	85	86	87
	88	89	90	91
	92	93	94	95
	96	97	98	99
	100	101	102	103
	104	105	106	107
	108	109	110	111
	112	113	114	115
	116	117	118	119
	120	121	122	123
	124	125	126	127
	128	129	130	131
	132	133	134	135
	136	137	138	139
	140	141	142	143
37	144	145	146	147

図付録 D.1

付録 E　TGC データ

E.1　RDO

表 8　RDO Hit variables

Name	型	範囲	意味
bcTag	int	0 から 3	0:Prev 1:Current 2:Next 3:Unknown
subDetectorId	int	103 104	103:A-side 104:C-side
rodId	int	1 から 12	ROD のアドレス。1/12 セクタの番号に対応
sswId	int	0 から 9	SSW のアドレス。
slbId	int	0 から 25	SLB チップの ID をソフトウェア上で変換した sbLoc という ID
l1Id	int	-	エレクトロニクスが判別した Level1ID
bcId	int	-	エレクトロニクスが判別した Bunch Corssing ID
slbType	slbType(enum)	0 から 6	0:DW 1:DS 2:TW 3:TS 4:IW 5:IS 6:UNKNOWN
adj	bool	-	adjancent 領域の Hit かどうか
tracklet	int	-	使用されていないはず
channel(bitpos)	int	40 から 199	Hit が SLB の Readout160bit の中のどの bit に対応しているか。

表 9　RDO Tracklet variables

Name	型	範囲	意味
bcTag	int	0 から 3	0:Prev 1:Current 2:Next 3:Unknown
subDetectorId	int	103 104	103:A-side 104:C-side
rodId	int	1 から 12	ROD のアドレス。1/12 セクタの番号に対応
sswId	int	0 から 9	SSW のアドレス。see**??**
slbId	int	0 から 25	SLB チップの ID をソフトウェア上で変換した sbLoc という ID
l1Id	int	-	エレクトロニクスが判別した Level1ID
bcId	int	-	エレクトロニクスが判別した Bunch Corssing ID
slbType	enum	0 から 6	0:DW 1:DS 2:TW 3:TS 4:IW 5:IS 6:UNKNOWN
delta	int	wire:-7 から 7 strip:-3 から 3	サジッタの大きさ
seg	int	0 から 1	Triplet Strip の TRIGA か TRIGB かを判別。他は 0 が入る
sub	int	0 から 3	TRIG1 TRIG2 などの候補の判別
rphi	int	0 から 31	Doublet 1/2 Triplet 1/3 の粒度の位置情報

表 10　RDO Highpt variables

Name	型	範囲	意味
bcTag	int	0 から 3	0:Prev 1:Current 2:Next 3:Unknown
subDetectorId	int	103 104	103:A-side 104:C-side
rodId	int	1 から 12	ROD のアドレス。1/12 セクタの番号に対応
l1Id	int	-	エレクトロニクスが判別した Level1ID
bcId	int	-	エレクトロニクスが判別した Bunch Corssing ID
strip	bool	-	true:Strip false:Wire
forward	bool	-	true:Forward false:Endcap
sector	int	Forward:0 と 1 Endcap:0 から 3	1 セクタの内のチェンバの識別。
chip	int	0 から 3	TRIG1 Highpt ボードのチップの識別
index	int	0 から 1	トリガ候補の識別
hipt	bool	-	H/L bit Highpt と Lowpt の判別
hitId	int	1 から 6	16 チャンネル粒度の場所の判別
sub	int	0 から 1	HitId を二つの領域に分ける (8 チャンネル粒度)
delta	int	wire:-15 から 15 strip:-7 から 7	サジッタの大きさ

表 11　RDO SL variables

Name	型	範囲	意味
bcTag	int	0 から 3	0:Prev 1:Current 2:Next 3:Unknown
subDetectorId	int	103 104	103:A-side 104:C-side
l1Id	int	-	エレクトロニクスが判別した Level1ID
bcId	int	-	エレクトロニクスが判別した Bunch Corssing ID
cand3plus	int	0 から 1	-
forward	bool	-	true:Forward false:Endcap
sector	int	Forward:0 と 1 Endcap:0 から 3	1 セクタの内のチェンバの識別
index	int	0 から 1	トリガ候補の識別
muplus	int	0 か 1	粒子の符号
threshold	int	1 から 6	トリガメニューのラベル。どのメニューでのトリガーか判別
overlap	int	-	-
roi	int	1 から 147	ROI の番号　巻末参照

E.2 PRD

表 12　IdHelper による variables

Name	型	範囲	意味
statioName	int	41 から 48	41:TGC1F 42:TGC1E 43:TGC2F 44:TGC2E 45:TGC3F 46:TGC3E 47:TGCFI 48:TGCEI
stationPhi	int	Endcap:1 から 48 Forward:1 から 24	φ方向の 1 チェンバ分の番号
stationEta	int	-5 から-1 1 から 5	R 方向のチェンバの番号 正は A-side 負は C-side
gasGap	int	1 から 3	各ステーションでのレイヤーの識別
isStrip	int	0 から 1	0:Wire 1:Strip
channel	int	0 から 1	チェンバ単位でのチャンネル数

表 13　TgcPrepData コンストラクタの持つ変数

Name	型	意味
RDOId	Identifier	OfflineID と呼んでいるもの。unsigned int
idDE	IdentifierHash	オブジェクトが入っていた Collection の Identifier のハッシュ
locpos	LocalPosition	検出器のローカルな座標位置 1 チェンバの中心を 0 として R、φ方向の座標を取る
rdoList	vector≪Identifier≫	その Event での Identifier のリスト
locErrMat	ErrorMatrix	チャンネルの Width から計算されたエラー ($\frac{Width}{\sqrt{12}}$)
detEl	TgcReadoutElement	その Hit 位置での TgcReadoutEelemt のポインタ

表 14　TgcCoinData Tracklet タイプの持つ変数

Name	型	意味
channelIdIn	Identifier	TGC2 のトリガ領域内の OfflineID
channelIdOut	Identifier	TGC3 のトリガ領域内の OfflineID
collectionIdHash	IdentifierHash	オブジェクトが入っていた Collection の Identifier のハッシュ
detElIn	TgcReadoutElement	TGC2 のトリガ領域内の TgcReadoutEelemt のポインタ
detElOut	TgcReadoutElement	TGC3 のトリガ領域内の TgcReadoutEelemt のポインタ
type	CoinDataType(enum)	0:TRACKLET 1:HIPT 2:SL 3:UNKNOWN
isAside	bool	true:A-side false:C-side
phi	int	TGC3 位置の stationPhi
isForward	bool	true:Forward false:Endcap
isStrip	bool	true:Strip false:Wire
trackletId	int	トリガ候補が一意に決まる領域に番号付けしたもの (2×sbLoc + sub)
posIn	LocalPosition	TGC2 上の位置のチェンバに対するローカルな位置座標
posOut	LocalPosition	TGC3 上の位置のチェンバに対するローカルな位置座標
widthIn	double	TGC2 上の位置でのチャンネルの Width
widthOut	double	TGC3 上の位置でのチャンネルの Width
delta	int	RDO の delta と同様

表 15　TgcCoinData Highpt タイプの持つ変数

Name	型	意味
channelIdIn	Identifier	TGC1 のトリガ領域内の OfflineID
channelIdOut	Identifier	TGC3 のトリガ領域内の OfflineID
collectionIdHash	IdentifierHash	オブジェクトが入っていた Collection の Identifier のハッシュ
detElIn	TgcReadoutElement	TGC1 のトリガ領域内の TgcReadoutEelemt のポインタ
detElOut	TgcReadoutElement	TGC3 のトリガ領域内の TgcReadoutEelemt のポインタ
type	CoinDataType(enum)	0:TRACKLET 1:HIPT 2:SL 3:UNKNOWN
isAside	bool	true:A-side false:C-side
phi	int	TGC3 位置の stationPhi
isForward	bool	true:Forward false:Endcap
isStrip	bool	true:Strip false:Wire
trackletId	int	トリガ候補が一意に決まる領域に番号付けしたもの (2×sbLoc + sub)
posIn	LocalPosition	TGC1 上の位置のチェンバに対するローカルな位置座標
posOut	LocalPosition	TGC3 上の位置のチェンバに対するローカルな位置座標
widthIn	double	TGC1 上の位置でのトリガ領域の Width
widthOut	double	TGC3 上の位置でのトリガ領域の Width
delta	int	RDO の delta と同様

表 16　TgcCoinData SL タイプの持つ変数

Name	型	意味
channelIdOut	Identifier	TGC3 のトリガ領域内の OfflineID
collectionIdHash	IdentifierHash	オブジェクトが入っていた Collection の Identifier のハッシュ
detElOut	TgcReadoutElement	TGC3 のトリガ領域内の TgcReadoutEelemt のポインタ
type	CoinDataType(enum)	0:TRACKLET 1:HIPT 2:SL 3:UNKNOWN
isAside	bool	true:A-side false:C-side
phi	int	TGC3 位置の stationPhi
isForward	bool	true:Forward false:Endcap
isStrip	bool	true:Strip false:Wire
trackletId	int	Wire のトリガ候補が一意に決まる領域に番号付けしたもの
trackletIdstrip	int	Strip のトリガ候補が一意に決まる領域に番号付けしたもの
posOut	LocalPosition	TGC3 上の位置のチェンバに対するローカルな位置座標
errMat	ErrorMatrix	TGC3 上の R、φ方向の Width(エラーではない)
roi	int	ROI の番号。RDO と同様
pt	int	Pt 閾値の判別ラベル。RDO の threshold と同様

付録 F　Athena インストラクション

ここで実際に MuonSpectrometer のみのデコードを例に Athena のインストラクションを行う。CERN にある lxplus というコンピュータ上を想定して記述する。これは CMT や Athana、それに関連するデータベースなどの外部ソフトウェアの全てがインストールされている。もし、別のコンピュータを使用する場合はソフトウェアがインストールされているディレクトリを調べる必要がある。

それでは、実際の手順を追って説明する。まず、最初に作業を行うディレクトリの作成と CMT コマンドを使用できるように環境を設定する必要がある。その為に以下のようにディレクトリの作成とセットアップファイルを読み込む (14.4.0 の部分は任意)。

```
[fkanega@lxplus226]~% mkdir 14.4.0
[fkanega@lxplus226]~% source /afs/cern.ch/sw/contrib/CMT/v1r20p20080222/mgr/setup.sh
[fkanega@lxplus226]~% which cmt
cmt='${CMTROOT}/${CMTBIN}/cmt.exe'
```

これにより、cmt コマンドが使用できるようになる。今度は Athena を使用するための requirements ファイルを用意し、CMT によりコンフィグレーション、セットアップを行う。以下が requirements ファイルの例である。

```
set   CMTSITE  CERN
set   SITEROOT /afs/cern.ch
macro ATLAS_DIST_AREA ${SITEROOT}/atlas/software/dist
macro ATLAS_TEST_AREA /afs/cern.ch/user/f/fkanega/14.4.0

apply_tag setup
apply_tag setupCMT

use AtlasLogin AtlasLogin-* $(ATLAS_DIST_AREA)
```

ATLAS_ TEST_ AREA は作業場所として作成したディレクトリである。apply_ tag の記述はこの後のセットアップの際に自動的に指定される tag で、setup は athena コマンドをこの段階で使用できるようにするのと、setupCMT は使用する CMT のバージョンを適切なものに設定する意味を持つ。適当なディレクトリ上で、この記述の requirements という名前のファイルに書き込み、cmt コマンドによってスクリプトを生成する。

```
[fkanega@lxplus226]~% cd 14.4.0
[fkanega@lxplus226]~/14.4.0% vi requirements
上のように記述、保存する。
[fkanega@lxplus226]~/14.4.0% cmt config
...
[fkanega@lxplus226]~/14.4.0% ls
cleanup.csh cleanup.sh requirements setup.csh setup.sh
```

上のインストラクションは一度実行すれば、設定を変えたいとき意外は行う必要はない。

今度はこのスクリプトを実行すればよいのだが、その時、tag を指定することによって、使用する Athna のバージョンを決めることになる。また、tag に noTest を指定しない場合、そのバージョンに合わせたディレクトリを ATLAS_ TEST_ AREA の下に作成する必要がある。下の例は AtlasTier0 の 14.4.0.2 を使用する場合である。

```
[fkanega@lxplus226]~/14.4.0% mkdir AtlasTier0-14.4.0.2
[fkanega@lxplus226]~/14.4.0% source setup.sh -tag=32,AtlasTier0,14.4.0.2,opt
...
[fkanega@lxplus226]~/14.4.0% echo $CMTPATH
/afs/cern.ch/user/f/fkanega/14.4.0/AtlasTier0-14.4.0.2:/afs/cern.ch/atlas/software
/buildsAtlasTier0/14.4.0.2
[fkanega@lxplus226]~/14.4.0% which athena
athena=athena.py
[fkanega@lxplus226]~/14.4.0% which athena.py
/afs/cern.ch/atlas/software/builds/AtlasCore/14.4.0/InstallArea/share/bin/athena.py
```

これで、AtlasTier0 の 14.4.0.2 での athena コマンドが実行できるようになる[*11]。tag はカンマで区切り、32 は 32bit コンパイルの設定で、AtlasTier0 と 14.4.0.2 はバージョンを、opt は最適化を行うモードで使用することを示す。AtlasTier0 という tag をつけない場合は、AtlasOffline が使用されることになる。

あとは、必要に応じて Package を持ってきて、コンフィグレーション、インストールを行い、jobOption を実行する。デフォルトだと athena がインストールされているディレクトリを参照するが、自分のディレクトリに Pakcage を持ってきてインストールすることによって、その Package のみ、自分のものが適用されるようになるので、修正が必要だったり、デフォルトとバージョンが違う Package を使いたい場合、そのパッケージを持ってくる必要がある。例として jobOption ファイルの集まりである MuonRecExample パッケージを持ってきて、jobOption を実行してみる。

[*11] tag に setup が指定されていない場合は、Package の setup をしなければ athena コマンドを使用することは出来ない。

```
[fkanega@lxplus226]~/14.4.0% cd AtlasTier0-14.4.0.2
[fkanega@lxplus226]~/14.4.0/AtlasTier0-14.4.0.2% cmt co -r MuonRecExample-01-01-06
MuonRecMuonSpectromter/MuonReconstruction/MuonRecExample
...
[fkanega@lxplus226]~/14.4.0% cd MuonSpectromter/MuonReconstruction/MuonRecExample/cmt
[fkanega@lxplus226]cmt% cmt config
...
[fkanega@lxplus226]cmt% source setup.sh
...
[fkanega@lxplus226]cmt% gmake clean
...
[fkanega@lxplus226]cmt% gmake
...
[fkanega@lxplus226]cmt% ls ~/14.4.0/AtlasTier0-14.4.0.2/
InstallArea MuonSpectrometer
[fkanega@lxplus226]cmt% cd ../run/
[fkanega@lxplus226]run% get_files -jo MuonDataRec_myTopOptions.py
[fkanega@lxplus226]run% vi MuonDataRec_myTopOptions.py
jobOption を修正
[fkanega@lxplus226]run% athena MuonDataRec_myTopOptions.py 2>&1 | tee log.txt
.....
```

パッケージを持ってくるときは cmt コマンドに co をつけて持って来る。r オプションはバージョンを指定するときに使い、指定しない場合は Head を持ってくる。上のように InstallArea というディレクトリができ、そこにリンクファイルなどが入っている。あとは適当なディレクトリで jobOption ファイルを実行するだけである。athena 実行後、そのディレクトリに Ntuple などが生成される。

デフォルトで使用されるパッケージのバージョンは以下のように調べることが出来る。

```
[fkanega@lxplus226]~% get_tag MuonSpectrometer/MuonReconstruction/MuonRecExample
 -r 14.4.0
14.4.0;AtlasReconstruction;/MuoonSpectrometer/MuonReconstruction
/MuonRecExample;MuonRecExample-01-01-06
```

謝辞

この二年間、学ぶ機会と指導、助言を頂いた指導教官の坂本宏教授 [a] に心より感謝いたします。

また、研究上で様々なご指摘と助言を頂いた川本辰夫氏 [a]、石野雅也氏 [a]、磯部忠昭氏 [a]、佐々木修氏 [b]、池野正弘氏 [b]、田中秀治氏 [b]、藏重久弥氏 [c]、松下崇氏 [c]、越智敦彦氏 [c]、石川明正氏 [c] 戸本誠氏 [f]、杉本拓也氏 [f]、福永力氏 [d]、菅谷頼仁氏 [e] に深く感謝いたします。

様々な機会でお話や、サポートを頂いた小林富雄氏 [a]、浅井祥仁氏 [g]、田中純一氏 [a]、、上田郁夫氏 [a]、真下哲郎氏 [a]、近藤敬比古氏 [b]、岩崎博行氏 [b] 他 ATLAS 日本グループの方々にも深く感謝いたします。

TGC エレクトロニクスグループで共に研究に励み、助言を頂いた織田勧氏 [a]、久保田隆至氏 [a]、野本裕史氏 [a] 、平山翔氏 [a]、結束晃平氏 [a]、越前谷陽佑氏 [a]、門坂拓哉氏 [c]、丹羽正氏 [c]、早川俊 [c]、中塚洋輝 [c]、奥村恭幸氏 [f]、高橋悠太氏 [f]、長谷川慧氏 [f] に感謝いたします。

また、同期の学生としてお世話になった東裕也氏 [g]、秋元銀河氏 [g]、鈴木拓也氏 [g]、山崎高幸氏 [a]、生出秀行氏 [a]、末廣徹氏 [a]、金子大輔氏 [a]、白雪さん [a] にも感謝します。

他、出張や、事務手続きの際に大変お世話になった秘書の安蒜律子さん [a]、片岡直子さん [a]、塩田雅子さん [a]、湯野栄子さん [a]、伊藤千代さん、塚本郁絵さんに感謝いたします。

上記の方々のおかげで、充実した研究生活を送ることが出来ました。心より感謝いたします。

所属:
東京大学素粒子物理国際研究センター (ICEPP)[a]
高エネルギー加速器研究機構 (KEK)[b]
神戸大学自然科学研究科 [c]
東京都立大学理学研究科 [d]
大阪大学理学部 [e]
名古屋大学理学研究科 [f]
東京大学理学系研究科 [g]

参考文献

[1] *ATLAS Level-1 Trigger Technical Design Report*, 1998.

[2] *ATLAS MuonSpectrometer Technical Design Report*, 1999.

[3] *The ATLAS Computing Workbook*, 2007.

[4] *Updates of the ATLAS Tracking Event Data Model(Release 13)*, 2007.

[5] Christian Arnault. *CMT Configuration Management Tool Version vlr18p20051101*, 2005.

[6] Athena. *AthenaDeveloperGuide-8.0.0*, 2004.

[7] BrunoLenzi. Muoncommsoftwareframework atlas twiki. `https://twiki.cern.ch/twiki/bin/view/Atlas/MuonCommSoftwareFramework#Data_samples`, 2009.

[8] ATLAS Collaboration. Atlas detector, 2008.

[9] Geant4. Geant4: A toolkit for the simulation of the passage of particles thorugh matter. `http://geant4.web.cern.ch/geant4/`, 2009.

[10] LHC images. Lhc images. `http://lhc-machine-outreach.web.cern.ch/lhc-machine-outreach/lhc_in_pictures.htm`.

[11] T. Kondo. Atlas-japan photos. `http://atlas.kek.jp/sub/photos/Physics/PhotoPhysicsSM.html`, 2008.

[12] Main.fredrik. Eventdatamodel atlas twiki. `https://twiki.cern.ch/twiki/bin/view/Atlas/EventDataModel`, 2009.

[13] Joao Pequenao. Cern document server: レコード 1095924: Computer generated image of the whole atlas detector. `http://cdsweb.cern.ch/record/1095924`, 2009.

[14] P.Mato. *GAUDI LHCb Data Processing Applications Framework Architecture Design Document*, 1998.

[15] Doxygen Global Search. Doxygen global search. `http://atlas-computing.web.cern.ch/atlas-computing/links/nightlyDocDirectory/globalDoxySearch.php`, 2009.

[16] TgcDocument. Tgcdocument. `https://twiki.cern.ch/twiki/bin/view/Main/TgcDocument`, 2008.

[17] The ATLAS TWiki. Webhome alas twiki. `https://twiki.cern.ch/twiki/bin/view/Atlas/`, 2008.

[18] 石野雅也. Atlas-μ system の準備状況, 2008.

[19] 田中秀治. アトラス実験ミューオントリガーチェンバーの開発, 2006.

[20] 丹羽正. Atlas 前後方ミューオントリガーシステムコミッショニングにおける sectorlogic による宇宙線トリガーの研究, 2008.

[21] 門坂拓哉. Atlas 前後方ミューオントリガーシステム sector logic 及びオンラインソフトウェアの開発, 2008.

[22] 杉本拓也. Atlas 前後方ミュー粒子レベル 1 トリガーシステム－全システム稼動に向けて, 2008.

[23] 高橋悠太. 大型ハドロン加速器実験用 μ 粒子検出器の動作検証, 2008.

[24] 桑原隆志. Atlas 前後方ミューオントリガーシステムの構築, 2007.