ATLAS 飛跡検出器開発用
新型ビーム試験 DAQ システムの構築

京都教育大学大学院 教育学研究科
教科教育専攻 理科教育専修 基礎物理学研究室
修士課程 2 年

西村 龍太郎

January 16, 2014

概 要

欧州原子核研究機構 (CERN) の陽子/陽子衝突型加速器 Large Hadron Colider (LHC) には汎用粒子検出器である ATLAS 検出器が設置されている。2023 年に開始予定の、HL-LHC のもとでのアップグレードされたアトラス実験の物理ランに向けて、ATLAS 検出器には新型のストリップ型シリコン検出器が搭載されることが予定されている。本研究では、新型のストリップ型シリコン検出器を評価するためのビームテスト用 DAQ システムを構築し、疑似パルス信号を用いた動作試験によって本システムが実際のビームテストに供しうるものであることを示した。また、DAQ 用ソフトウェアについて、機能のモジュール化・マルチプロセス化を行い、今後の試験に合わせた機能の追加・変更が容易に行えるようにし、汎用性を高めたソフトウェアを新たに開発した。

目 次

図 目 次

表 目 次

第1章　序論

1.1　背景

1.1.1　LHC および ATLAS 検出器について

1.1.1.1　LHC

LHC(Large Hadron Collider、大型ハドロン衝突型加速器) は、スイスとフランスの国境に位置する CERN(欧州原子核研究機構) に設置された全周 27km の陽子・陽子衝突型円形加速器である。

加速器のリング上には 4 箇所のビーム衝突点が設けられており、Point1：ATLAS 検出器、Point2：ALICE 検出器、Point5：CMS 検出器、Point8：LHC-B 検出器の 4 つの検出器がそれぞれ設置されている。

2010 年 3 月より最高重心系エネルギー 7TeV での素粒子実験を開始し、2012 年 4 月には重心系エネルギー 8TeV で稼働して多くのデータを収集した。現在はシャットダウン [1] され、メンテナンス・改修が行われている。

LHC では、2023 年に向けてルミノシティを $5 \times 10^{34}\mathrm{cm}^{-2}\mathrm{s}^{-1}$ に上げるアップグレードが予定されており、その後 10 年間で積分ルミノシティ $3000\mathrm{fb}^{-1}$ を目指す。

図 1.1 に LHC の概要を、表 1.1 に LHC の主要パラメータの設計値を示す。

[1] Long Shutdown 1

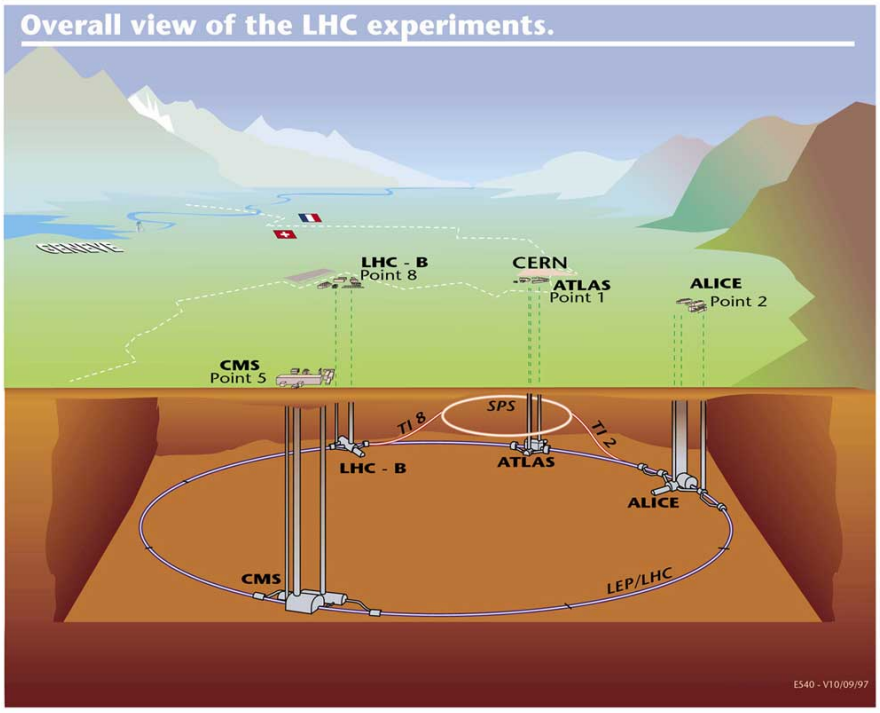

図 1.1: LHC (画像提供：CERN ATLAS Experiment)

リング長	26.7km
重心系エネルギー	7TeV+7TeV
デザインルミノシティ	$10^{34}\mathrm{cm}^{-2}\mathrm{s}^{-1}$
バンチ間隔	25nsec (40MHz)
バンチ当たり陽子数	10^{11}
超伝導双極磁石	1232 台
双極磁石長	14.2m
双極磁石の磁場	8.33T
ビーム半径	16μm
陽子衝突数	約 20

表 1.1: LHC の主要パラメータの設計値

1.1.1.2 ATLAS 検出器

ATLAS 検出器 (A Toroidal LHC ApparatuS、円環状磁場 LHC 測定器) は、LHC の 4 つのビーム衝突点のうちの 1 つに ATLAS 実験グループが設置した汎用粒子検出器であり、Higgs 粒子や標準模型を超える物理現象の探索を行なっている。ビーム衝突点から外

側に向かって順に内部飛跡検出器、超伝導ソレノイド磁石、電磁カロリーメータ、ハドロンカロリーメータ、ミューオン検出器で構成されている。内部飛跡検出器はビーム衝突点から外側に向かって順にピクセル型シリコン検出器、シリコンマイクロストリップ検出器 (SemiConducter Tracker、SCT)、遷移輻射飛跡検出器 (Transition Radiation Tracker、TRT) から構成される。

ATLAS 検出器は 2023 年の LHC アップグレードに合わせてより高いルミノシティに対応するためのアップグレードが予定されている。

図 1.2 に ATLAS 検出器の概要を示す。

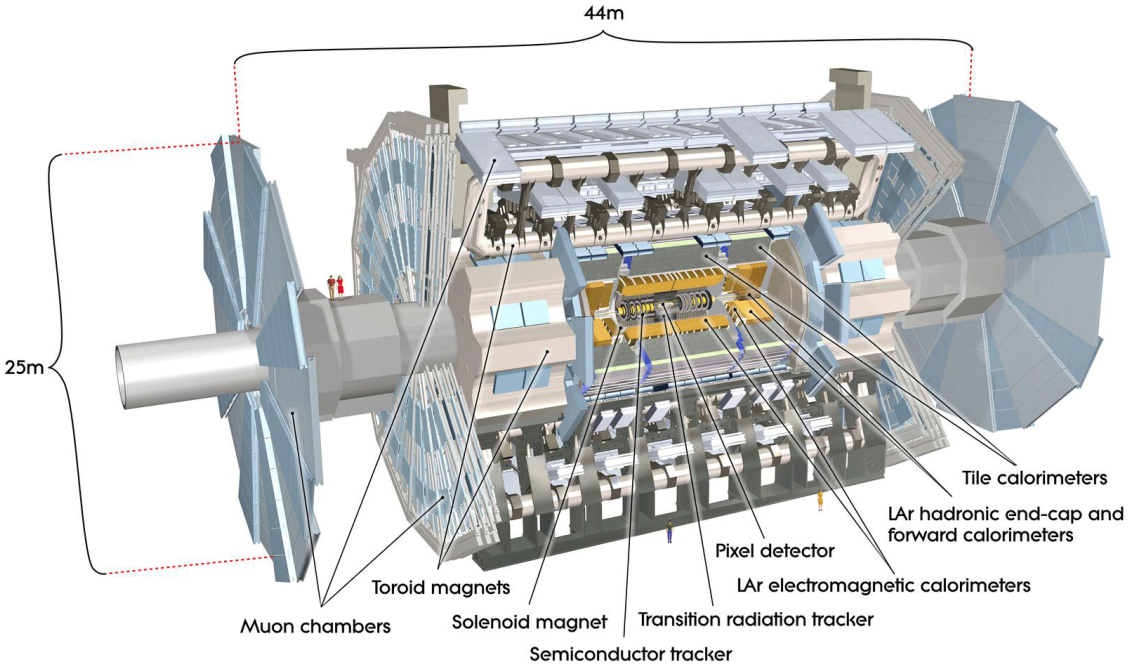

図 1.2: ATLAS 検出器 (画像提供：CERN ATLAS Experiment)

1.1.2 SCT

SCT は ATLAS 検出器のビーム衝突点に設置されている内部飛跡検出器を構成するストリップ型シリコン検出器である。ビーム衝突点からの荷電粒子は、内部飛跡検出器の外側に設置された超伝導ソレノイド磁石が発生する磁場によって曲げられ、SCT を通過する。このときに SCT で測定した粒子の飛跡情報から荷電粒子の運動量を測定することができる。

SCT は図 1.3 に示すように、ビームラインに平行な方向に 4 層 (barrel-SCT)、垂直な方向に 9 層 (EndCap-SCT) の検出器から構成されている。

2023 年のアップグレードにおいては、ルミノシティの向上に伴って入射粒子が増加し、放射線損傷、ヒット占有率[2] の増大が見込まれる。そこで、SCT においてはより放射線耐性を高め、パターン認識能力を向上した新型の検出器に交換されることが予定されている。

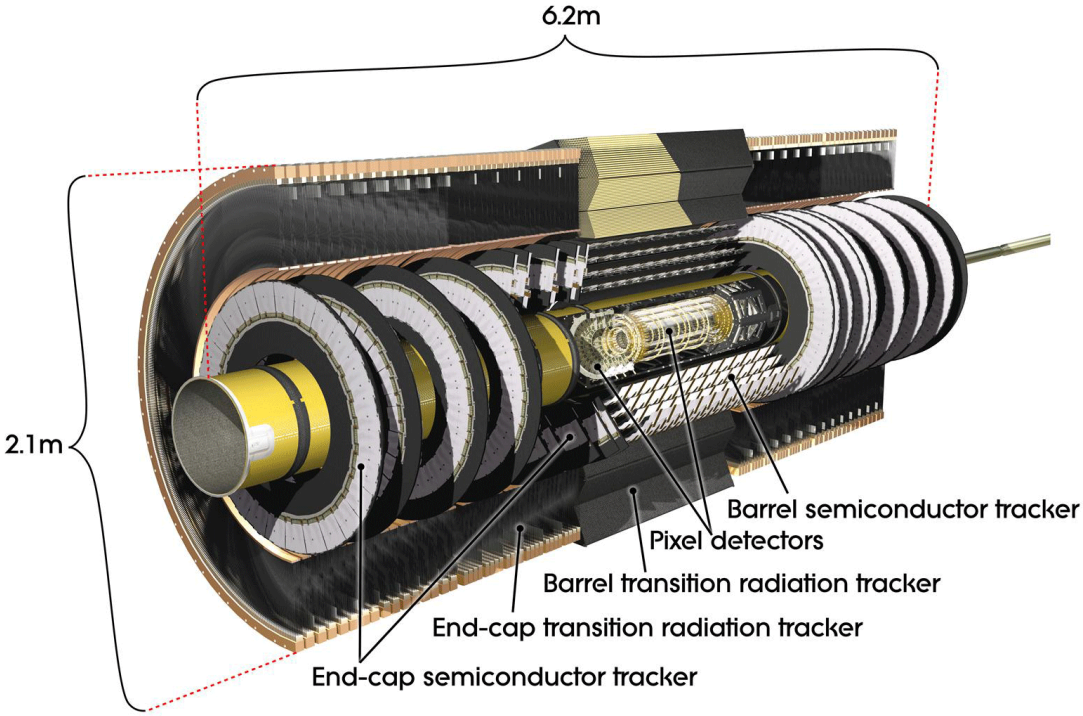

図 1.3: 内部飛跡検出器 (画像提供 : CERN ATLAS Experiment)。9 層存在する EndCap-SCT には前方後方合わせて 1976 個、4 層存在する Barrel-SCT には 2112 個の SCT モジュールがそれぞれ配置されている。

1.1.3 SCT モジュール

SCT の各層には、裏表 2 枚のストリップ型シリコン (シリコンストリップ) センサーと、読出し用 ASIC[3] である Atlas Binary Chip DMILL (ABCD) を搭載した基板によって構成されるモジュール (SCT モジュール) が搭載されている。

LHC アップグレード後の高ルミノシティ下での運転でヒット占有率を下げるため、アッ

[2] ヒット数/全イベント数。
[3] Application Specific Integrated Circuit。特定用途のための集積回路。

プグレード用に開発されている新型 SCT モジュールではシリコンストリップの長さが従来の 12.8cm から 2.4cm(内側 4 層)・4.8cm(外側 1 層) がと短くなり、ストリップの間隔は 80μm から 74.5μm となる。また、新型 SCT モジュールの読み出し ASIC である Atlas Binary Chip Next (ABCN 又は ABCn250) は 1 個当たり 128 本のシリコンストリップセンサーを読み出し、各ストリップのヒットの有無を返す。ABCn250 は hybrid と呼ばれる基板に、10 個まとめて 1 列にしたもの (column) が 2 列配置されている。hybrid は SCT モジュールの片面につき 2 つ配置されており、片面のシリコンストリップセンサーの信号は 40 個の ABCn250 チップによって読み出される。hybrid は SCT モジュールの両面に計 4 つ配置されているので、1 モジュールでは 80 個の ABCn250 チップが 10240 本のシリコンストリップを読み出すことになる。図 1.4 に ABCn250 チップを用いた SCT モジュールのプロトタイプの写真を示す。ABCn250 チップについては第 2 章にて述べる。

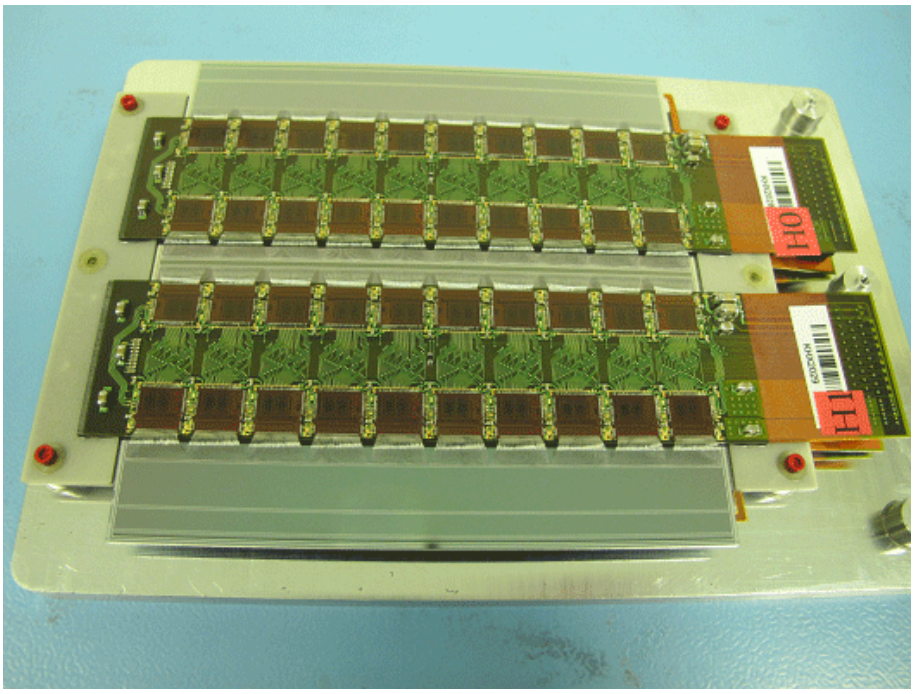

図 1.4: SCT モジュールのプロトタイプ ([Mechanical Studies towards a Silicon Microstrip Super Module for the ATLAS Inner Detector upgrade at the High Luminosity LHC] p6 [15])

1.1.4　ビームテスト

ビームテストは、開発されたシリコンストリップセンサーに対し、実際に粒子ビームライン上で正常に動作するかテストを行うものである。

SCT モジュール内を粒子が通過するタイミングで外部からトリガー信号を入力し、

- データが正しく取得できるか
- ビーム照射前と照射中・照射後でヒット特性にどのような変化が生じるか調べ、適切でないヒットが生じていないか

について確認し、開発されたシリコンストリップセンサーが ATLAS 検出器での使用に耐えうるものかを評価する。

図 1.5 にビームテストのイメージ図を示す。

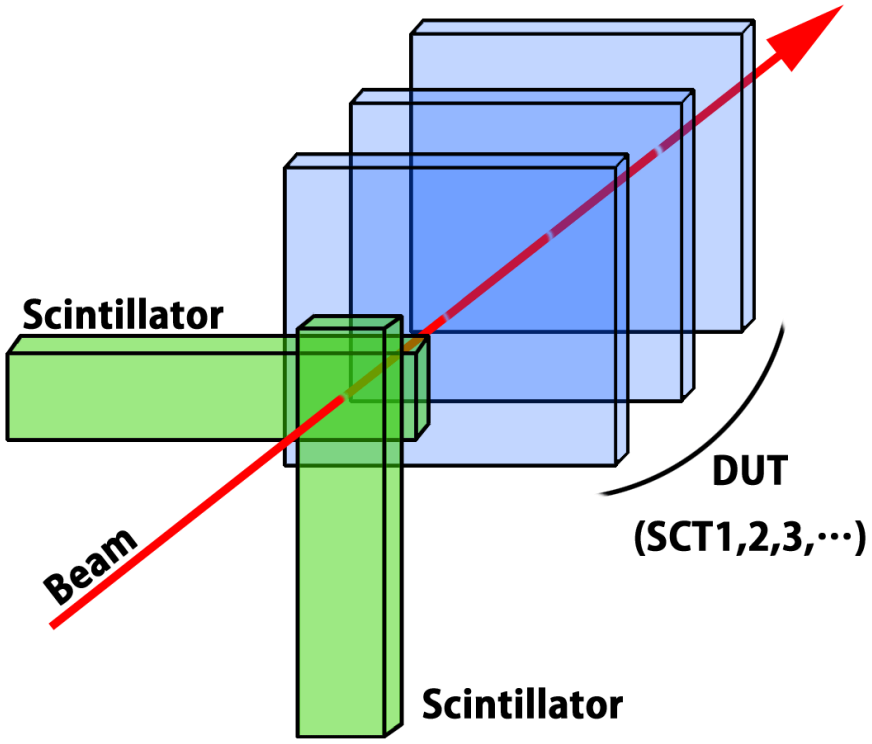

図 1.5: ビームテストのイメージ。ビームの通過する直線状に試験を行うセンサー (DUT) を含むすべての測定機器とビーム通過検出用の Scintillator を配置し、ビームがすべての Scintillator を通過した時にトリガー信号を入力してデータを取得する

1.1.5 ABCn250 1Chip モジュール

ABCn250 1Chip モジュールとは、新型の SCT モジュールと同様の ABCn250 チップが 1 個搭載されたモジュールである。通常の SCT hybrid と同様の信号線が用意されているので、hybrid を制御する場合と同様の方法で制御することができる。128 本のストリップセンサー読出し用チャネルのうち、Ch 30、32、34、36、…、96 の 34 チャネルについてはビームテスト等の性能評価に使用する小型のセンサーを接続するためのコネクタが接続されている。コネクタからチャネルへの配線が 1 つ置きであるのは、隣り合うチャネルとのクロストークによる影響を避けるためである。

1.1.6 TLU

TLU(Trigger Logic Unit) は、データ収集システム (Data Acquisition System、DAQ) において、各測定機器の状況を把握し、トリガー信号の送出を制御するものである。

基本的な機能としては、

トリガー信号送出制御
: ビームテストにおいて粒子ビームが通過したのを検知した際に、各測定機器が出力するステートに基づいて DAQ システムが次のイベント取得が可能かどうかを判断し、イベント取得が可能な時にのみトリガー信号を送出する。

同期用信号送出
: 各測定機器からの信号を一つのイベントとして処理するために、イベント識別用の情報を付加する必要がある。このイベント識別用の情報を生成する元となる共通の信号を各測定機器に供給する。

の 2 つが必要となる。

1.2 新型ビーム試験用 DAQ システムの目的

ABCn250 からの信号を読み出すための DAQ システムにはいくつかの先行研究が存在するが、ビームテスト用に汎用的に用いることができる DAQ システムはまだ存在せず、ビームテストの都度それ専用の DAQ システムを開発しているのが現状である。そこで、本

研究においては、既存の ABCn250 読出し用 DAQ システムをベースとしてビームテスト用に汎用的に用いることができる DAQ システムを新たに開発することを第一の目的とした。開発する DAQ システムがビームテスト試験に耐えうるか否かについては実際のビームテスト、または疑似的なテスト環境による試験によって確認する。

また、DAQ 用ソフトウェアについて、既存の ABCn250 読出し用 DAQ システムは測定機器の制御、データの記録および解析が単一のプログラムになっていたために、複雑化し、機能の追加・拡張が難しいものとなっていた。また、データの記録と解析が同時には実行できず、速度の向上にも限界があった。そこで、新たに開発する DAQ 用ソフトウェアでは各処理を独立したソフトウェアモジュールに分離し、プロセス間通信によって接続することによって構造を単純化して機能追加・拡張を容易にすること、マルチプロセス化による実行速度の向上を測ることを第二の目的とした。

第2章　信号読出し用ASIC

2.1　ABCn250(ABCN)

ABCn250[4] はシリコンストリップセンサーからの信号読出しのために設計された ASIC である。外部につながる信号線には次のものがある。

- BCO：LHC のビーム衝突頻度である 40MHz のクロックを入力する
- DCLK：ABCn250 のデータ出力を同期させるクロック (40MHz/80MHz) を入力する
- L1：L1 トリガーを入力する (L1 トリガーは COM を使用して送ることもできる)
- RESETB：チップを電源投入直後の状態に戻すための信号を入力する
- COM：レジスタ値書き込みなどのコマンドビットストリームを入力する
- DATAOUT：センサーのヒット情報などを出力する

ABCn250 チップはシリコンストリップセンサーからの信号を増幅、波形整形した後、設定された閾値を超えているか否かでヒットの判定を行う。ヒットの有無はバイナリデータとしてバッファメモリに順次保存されていく。チップが L1 トリガー [1] を受け取ると、バッファメモリに保存されたバイナリデータにチップ自身の識別番号、ヒットのあったストリップセンサーの番号の情報を付加して出力する。

チップにはヒット判定に用いる閾値などのパラメータを保存するためのレジスタが用意されており、これらには外部からビットストリームを Command 信号線 (COM) から入力して書き込む。チップは書き込まれたレジスタの値に基づいて動作する。図 2.1 に ABCn250 のブロックダイアグラムを示す。

[1] Level 1 Trigger。ATLAS 検出器では事象を検出するためのトリガーが三段階で構成されており、最初の 1 つ目を L1 トリガーと呼ぶ

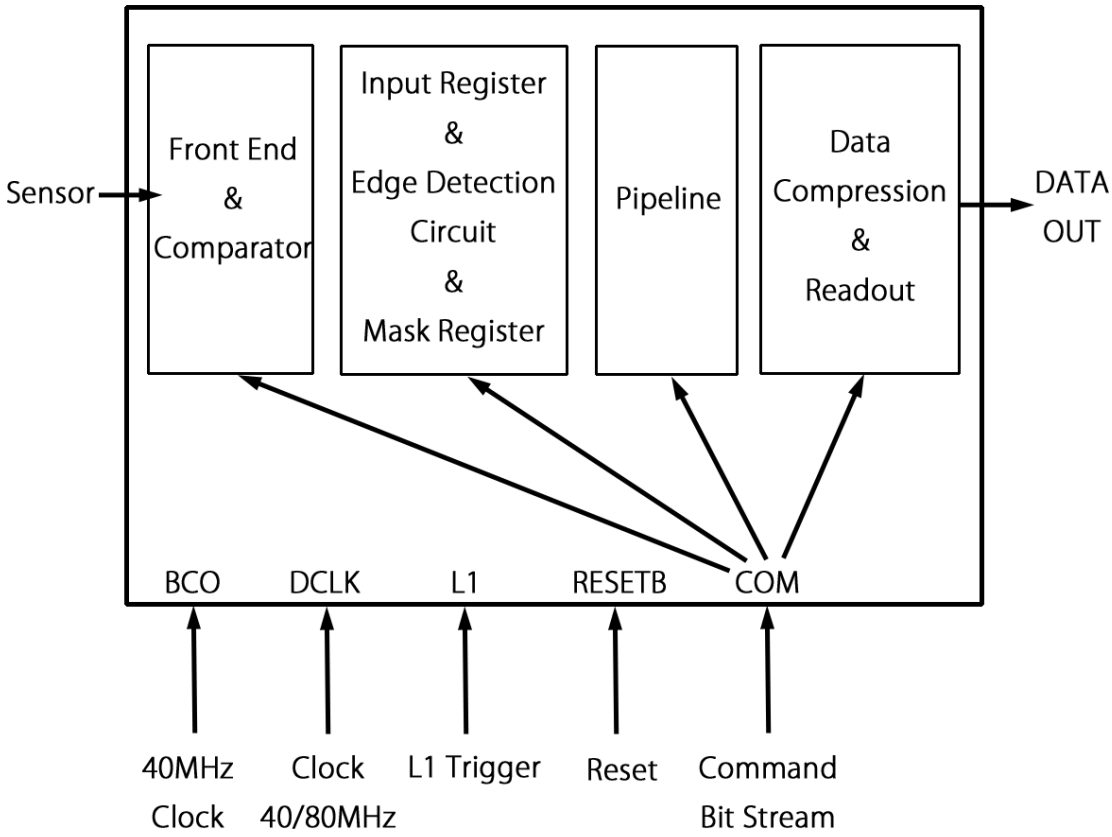

図 2.1: ABCn250 ブロックダイアグラム

以下、センサーからの信号読出しにおけるチップの重要な機能について説明する。

2.1.0.1 Front End & Comparator

ABCn250 の Front End は正負両方の信号に対応し、隣接する信号線とのクロストークは 5%以下である。Front End に入力された信号を Preamplifier が増幅し、Shaper が波形整形を行う。ゲインは 100mV/fC、ノイズは 370 + 76.7C[electrons](C は電気容量[pF])[7] である。増幅・整形された信号が設定した閾値を上回る場合に Comparator がヒット信号を出力する。

2.1.0.2 Input Register & Edge DetectionCircuit & Mask Reg-ister

Input Register は Comparator からの信号を幅 128bit、深さ 1bit のレジスタに保持し、BCO のクロックに同期して出力する。Edge Detection Circuit は設定されたヒットの検出方法に基づき、Comparator からの信号からヒットの有無を判定する。Mask Register は指定されたストリップからの信号を遮断 (Mask) する。

2.1.0.3 パイプライン

L1 トリガーは対応するバンチ衝突事象 (イベント) に対して 2 から 3μs 遅延する。ABCn250 は自身が読み出す 128 本のストリップからのヒットの有無を LHC におけるバンチ衝突事象間隔である 25ns 毎に保存されるパイプラインと呼ばれるバッファを持っている。このパイプラインは 256bit の深さを持つため、25ns $\times$ 256 $=$ 6.7μs 前までのヒット情報を保持できる。L1 トリガーを受け取った ABCn250 は予め設定された遅延時間 (L1 Delay) の分だけさかのぼってパイプラインに書き込まれたヒット情報を出力する。この時、L1 トリガーの遅延に該当するイベントと時系列的に前後のイベントのヒット情報の合わせて 3 つ分のバンチ衝突のヒット情報を出力する。

2.1.0.4 Data Compression

パイプラインには 128 本のストリップすべてのヒット情報が保持されているが、L1 トリガーを受け取ってヒット情報を出力する際にはヒットが有るストリップについてのみのアドレスとヒット情報を出力することにより、データ量を削減している。

2.1.0.5 Readout

ABCn250 は、データの出力時の振る舞いについて Master、Slave、End の 3 つのモードが存在する。各モードに設定されたチップは以下の様に振る舞う。

Master

自分の出力の先頭に受け取った L1 トリガーの回数、バンチ衝突回数を付加して出力する。

Slave

自分の出力のみを出力する。

End

自分の出力に続いてデータの終了位置であることを示すトレイラーを付加して出力する。

データが読み出されるときには、Master → Slave(1、2、3…) → End の順に読出しが行なわれる。Master、End の役割を 1 つのチップに重複して設定することもでき、その場合は column から読み出されるチップはそれ 1 つだけとなる。なお、テスト用の 1 チップモジュールはこの設定で使用することになる。

2.1.0.6 各種レジスタ

ABCn250 では、様々な回路がセンサーからの信号の読出しやデータ圧縮などの処理を行う。その各処理を外部からコントロールするために、様々な設定用パラメータがレジスタに保持される。各レジスタにはそれぞれアドレスが割り振られており、Command 用信号線に送るパラメータにこのアドレスを付加することで書き換えるレジスタを指定することができる。

2.1.0.7 Calibration Pulse

ABCn250 には内部テスト用に自分で Front End に Calibration 用の信号を入力する機能が存在する。この Calibration 用の信号の入力強度はパラメータを書き込むことによって変更することができる。また、一度に信号を入力できる Front End のチャネルはクロストークを抑えるために 3 つ置きになっており、どのチャネルに入力するかについてもパラメータの書き換えによって設定する。

2.1.0.8 動作モード

ABCn250 には Send_ID、Data_Taking、Read Register、Clock Feed Through の 3 つの動作モードが存在する。それぞれの動作モードにおいてチップは以下の様に動作する。

Send_ID

L1 トリガーを受け取るとレジスタ値 (Configuration Register 1) を応答する。

Data_Taking

Enable Data_Taking Command を受信するとこのモードになる。L1 トリガーを受け取ると各チャネルのヒット情報を応答する。

Read Register

Read Register Command を受信するとこのモードになる。L1 トリガーを受け取ると直前のコマンドで指定したレジスタ値を応答する。

Clock Feed Through

入力されたクロック信号を DATAOUT からそのまま出力する。L1 トリガーの入力に対しては応答しない。

チップの基本的なモードは Send_ID Mode であり、他のモードになっている場合もレジスタ値を書き込むと自動的にこのモードになる。電源投入時も仕様上は自動的に Send_ID モードとなるが、Clock feed through レジスタが 0 であるため、結果的には Clock Feed Through モードになる。

2.1.0.9 Reset

ABCn250 には、Power up reset、Soft Reset、BC Reset、Hard Reset の 4 種類のリセットが搭載されている。それぞれ以下のような機能になる。

Power up reset

電源投入時の自動リセット。全レジスタが初期値になる。

Soft Reset

Command 信号線 (COM) から所定のビットストリームを送ることによって実行される。

設定パラメータを除くレジスタ・バッファが初期化される。

BC Reset

Command 信号線 (COM) から所定のビットストリームを送ることによって実行さ

れる。

バンチ衝突数カウントが初期化される。

Hard Reset

Reset 信号線 (RESETB) を短時間 (25ns 以上)Low にすることによって実行される。電源投入直後と同等の状態になり、全レジスタが初期値になる。

2.2 ABCn250 の制御および信号読出しの流れ

2.2.0.10 ABCn250 の制御コマンド

ABCn250 はビットストリームによって制御を行う。送信時のコマンドビットストリームの構造を表 2.1 に、主要なレジスタの書き込み・読出し用コマンドを表 2.2、表 2.3、表 2.4 に、その他のコマンドを表 2.5 に示す。なお、使用しないコマンドや未実装のコマンドについて一部省略している。詳細については ABC-N ASIC Specifcations VersionV1.3.1[4] を参照のこと。

Field 1	Field 2	Field 3	Field 4	Field 5	Field 6
3 bit	4 bit	8 bit	7 bit	6 bit	16bit(without Mask Register)

表 2.1: ABCn250 コマンドビットストリーム構造

Type	Field 1	Field 2	Field 3	Description
Level 1	110	—	—	L1 Trigger
Fast	101	0100	—	Soft Reset
		0010	—	BC Reset
Slow	101	0111	Command	Slow Control Command

表 2.2: ABCn250 Control Protocol

2.2.0.11 ABCn250 のデータ出力フォーマット

ABCn250 が Data_Taking モードの時にデータとして出力するビットストリームには、データの始まりを示すヘッダとデータの終わりを示すトレーラーの間に、ヒットのあったストリップの Address、及びヒットパターンなどの情報を含むデータブロックが挟まる構造になっている。また、Send_ID、Read Register モードの時に出力するビットストリー

Field 3	Field 4	Field 5	Field 6	Description
0001,1101	aaaaaaa	000 000	dddd,dddd,dddd,dddd	Write Configuration Register 1
1000,1101	aaaaaaa	001 000	d—,—-,—-,—d(128bit)	Write Mask Register
0001,1101	aaaaaaa	010 000	dddd,dddd,dddd,dddd	Write Calibration Register
0001,1101	aaaaaaa	011 000	dddd,dddd,dddd,dddd	Write Threshold Registers
0001,1101	aaaaaaa	111 000	dddd,dddd,dddd,dddd	Write Bias Register 1
0001,1101	aaaaaaa	000 100	dddd,dddd,dddd,dddd	Write TrimDac
0001,1101	aaaaaaa	111 010	dddd,dddd,dddd,dddd	Write Bias Register 2
0001,1101	aaaaaaa	111 100	dddd,dddd,dddd,dddd	Write Bias Register 3
0001,1101	aaaaaaa	001 100	dddd,dddd,dddd,dddd	Write Configuration Register 2
0001,1101	aaaaaaa	010 100	dddd,dddd,dddd,dddd	Write Latency Register

表 2.3: ABCn250 Slow Control Write Register Command

Field 3	Field 4	Field 5	Field 6	Description
0000,1101	aaaaaaa	000 001	—	Read Configuration Register 1
0000,1101	aaaaaaa	011 101	—	Read Fuse Register
0000,1101	aaaaaaa	011 001	—	Read Threshold Registers
0000,1101	aaaaaaa	111 001	—	Read Bias Register 1
0000,1101	aaaaaaa	111 011	—	Read Bias Register 2
0000,1101	aaaaaaa	111 101	—	Read Bias Register 3
0000,1101	aaaaaaa	001 101	—	Read Configuration Register 2
0000,1101	aaaaaaa	010 101	—	Read Latency Register
0000,1101	aaaaaaa	010 001	—	Read Calibration Register

表 2.4: ABCn250 Slow Control Read Register Command

ムについてもデータブロックの位置にパラメータが入る構造となる。表 2.6 に ABCn250 の基本データ出力フォーマットを示す。

DT(Data Type)

Data が L1 Trigger Data か Information Data かを示す。ただし、現行の ABCn250 においては値は常に'0' となる。

L1

Data が最後のカウンタリセットから何回目の L1 トリガーによるものかを示す。

BC(Beam Crossing Number)

Data が最後のカウンタリセットから何回目のバンチ衝突によるものかを示す。

Field 3	Field 4	Field 5	Field 6	Description
0000,1101	aaaaaaa	100 000	—	Instr. Test Pulse to Input_Reg
0000,1101	aaaaaaa	101 000	—	Instr. Enable Data taking Mode
0000,1101	aaaaaaa	110 000	—	Instr. Issue Calibration Pulse

表 2.5: ABCn250 Slow Control Command (without Read/Write Register)

Preamble	DT	L1	BC	Sep	Data Block		Data Block	Trailer
11101	0	4 bit	8 bit	1	<Block_1>	……	<Block_n>	100000000 00000000

表 2.6: ABCn250 基本データ出力フォーマット

Data Block

チップから出力される Data の本体。Physics Data、No-Hit Data、Configuration Data 1、Register Readout Data、Error Data の 5 つの型に分類される。

Data Block の各型については以下で説明する。

2.2.0.11.1 Physics Data

この型はセンサから読み出したヒット情報や ABCn250 自身の Calibration Pulse によるヒット情報を出力する際のデータフォーマットであり、表 2.7 のように複数の Data Packet で構成される。

Data Block			
Data Packet	Data Packet		Data Packet
<Packet_1>	<Packet_2>	⋯	<Packet_n>

表 2.7: ABCn250 Physics Data 出力フォーマット

Data Packet には Isolated Hit Data Packet と Non Isolated Hit Data Packet の 2 種類が存在する。Physics Data はこれらを組み合わせた形で構成される。以下に各 Data Packet のフォーマットを示す。

Isolated Hit Data Packet

この Data Packet はあるヒットチャネルについて隣接するチャネルにヒットがない場合のものである。フォーマットは表 2.8 のようになる。

Non Isolated Hit Data Packet

この Data Packet はあるヒットチャネルについて隣接するチャネルにもヒットが

Isolated Hit Data Packet				
Header	Chip Address	Channel Address	Sep	Hit Pattern
01	7 bit	7 bit	1	3 bit
	MSB First	MSB First		MSB First (Previous Hit, Current Hit, Next Hit)

表 2.8: ABCn250 Isolated Hit Data Packet のフォーマット

ある場合のものである。フォーマットは表 2.9 のようになる。

Non Isolated Hit Data Packet								
Header	Chip Address	First Hit Channel Address	Sep	First Hit Pattern	Sep	Next Hit Pattern	Sep	Last Hit Pattern
01	7 bit	7 bit	1	3 bit	1	3 bit	1	3 bit
	MSB First	MSB First		MSB First		MSB First		MSB First

表 2.9: ABCn250 Non Isolated Hit Data Packet のフォーマット

2.2.0.11.2 No Hit Data

この Data Packet は、そのトリガー分のデータについて、チップ内に 1 つもヒットしたチャネルがない場合に送信されるものである。フォーマットは表 2.10 のようになる。

No Hit Data Packet
0011

表 2.10: ABCn250 No Hit Data Packet のフォーマット

2.2.0.11.3 Configuration Data 1

この Data Packet はチップが Send_ID モードにある時に L1 トリガーを受け取った場合に送信されるものである。フォーマットは表 2.11 のようになる。

2.2.0.11.4 Register Readout Data

この Data Packet はチップが Read Register モードにある時に L1 トリガーを受け取った場合に送信されるものである。フォーマットは表 2.12 のようになる。送信されるパラメータは直前に送信した Read Register コマンド (表 2.5) によって決まり、指定されたレジスタを示す Register Address(表 2.13) が Data Packet に含まれる。

Configuration Data 1 Packet						
Header	Chip Address	Sep	MSB Byte of Config Pattern	Sep	LSB Byte of Config Pattern	Sep
000	7 bit	111	8 bit	1	8 bit	1
	MSB First		MSB First		MSB First	

表 2.11: ABCn250 Configuration Data 1 Packet のフォーマット

Register Readout Data Packet							
Header	Chip Address	Sep	Register Address	MSB Byte of Config Pattern	Sep	LSB Byte of Config Pattern	Sep
000	7 bit	010	5bit	8 bit	1	8 bit	1
	MSB First		MSB First	MSB First		MSB First	

表 2.12: ABCn250 Register Readout Data Packet のフォーマット

2.2.0.11.5 Error Data

この Data Packet はチップが Data_Taking モードにあるときに有効なデータがない、またはバッファオーバーフローが生じた場合に送信される。フォーマットは表 2.14、Error Code が示す内容は表 2.15 のようになる。なお、使用頻度の低い Register について一部省略している。詳細については ABC-N ASIC Specifcations VersionV1.3.1[4] を参照のこと。

2.3 ABCn250 へのコマンド送信の流れ

ABCn250 の制御は、前述した各種コマンドを適宜送信することによって行う。ここでは、例として、ABCn250 に Calibration Pulse を発行させ、ヒットデータを取得する時に送信するコマンドとその順番を記す。

1. Soft Reset を送信してレジスタ以外のバッファをリセットする [2]。
2. 各種レジスタを設定するコマンドを送信し、ABCn250 のレジスタに値を設定する。
3. Soft Reset を送信してレジスタ以外のバッファをリセットする。

[2] 本来はここで Soft Reset を行う必要はないが、ABCn250 は電源投入直後に最初に送信されたコマンドを正しく認識しない場合がある [1] ため、正しく受信しなくとも問題のないコマンド (この場合 Soft Reset) を最初に送信している。

Register Address	Register
000 00	Configuration Register 1
001 00	Mask Register
010 00	Calibration Register
011 10	Fuse Register
011 00	Threshold Register
111 00	Bias Register 1
111 01	Bias Register 2
111 10	Bias Register 3
000 10	TrimDac
001 10	Configuration Register 2
010 10	Latency Register

表 2.13: ABCn250 主要な Register Address の一覧

Error Data Packet			
Header	Chip Address	Error Code	Sep
000	7 bit	3 bit	1
	MSB First		

表 2.14: ABCn250 Register Readout Data Packet のフォーマット

4. Data Taking モードに変更するコマンドを送信し、ヒット情報を出力するように設定する。

5. Calibration Pulse 発行用コマンドを送信し、Front End にテスト信号を入力する。

6. L1 トリガー発行コマンドを送信し、ABCn250 にヒット情報を出力させる。

Error Code	Error
001	No Data Available (The chip has not received an L1 command)
100	Buffer Overflow (Soft Reset needed)

表 2.15: ABCn250 Error Code の一覧

第3章　開発したDAQシステム

この章では本研究で開発したDAQシステムに関して説明する。本システムは、遠藤理樹氏 (大阪大学)、田窪洋介氏 (KEK[1])、Sergio Gonzalez Sevilla 氏 (University of Geneva) の開発した ABCn250 読出し用ファームウェア及びソフトウェアを基本とし、安芳治氏 (KEK)、久保田智徳氏 (東京工業大学)、山本賢氏 (京都教育大学) と共同で開発を行なったものである。

[1]高エネルギー加速器研究機構

3.1 基本的なシステム構成

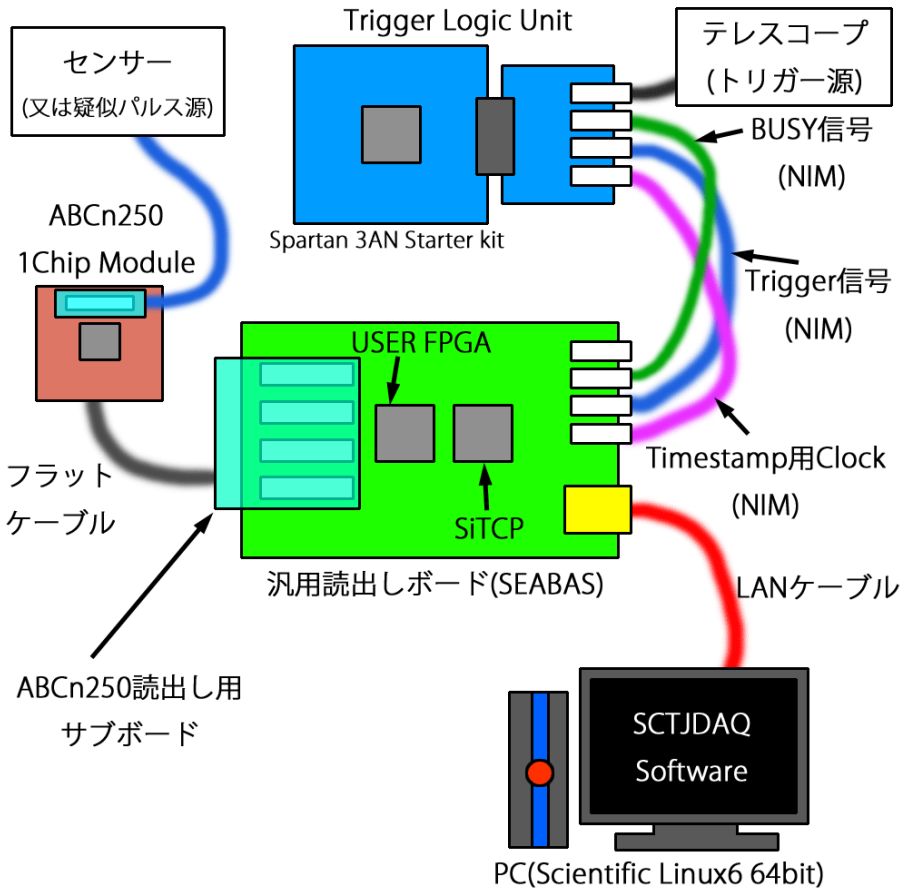

図 3.1: SCTJDAQ 概要図。ABCn250 の 1Chip Module 1 つを読み出す時の構成

図 3.1 に、本研究で開発した DAQ システム (以下 SCTJDAQ) の概要を示す。

ABCn250 1Chip Module[2] にはセンサー (またはそれに代わる信号源) が接続されており、コマンド及びデータのやり取りに用いる信号線は読出し用サブボード [3] を介して汎用読み出しボード (SEABAS) と接続されている。この SEABAS ボードは LAN ケーブルによって制御用 PC と接続されており、TCP/IP 通信によって ABCn250 1Chip Module に対して各種コマンド・レジスタ書き込み・データ読出しを行なうことが可能である。

プラスチックシンチレーター (またはそれに代わるトリガー信号源) からのトリガー信号は、Trigger Logic Unit (TLU) を介して SEABAS に入力される。TLU はトリガー信

[2]ABCn250 チップを 1 個搭載したテスト用モジュール

[3]岸田拓也氏 (東京工業大学) の修士論文執筆のための研究に使用された 8LINK サブボード。最大 8 個の ABCn250 モジュールを同時に制御できる

号を受信した場合、SEABAS から出力される BUSY 信号の状態によって SEABAS 側のトリガー信号受信の可否を判断し、可能な場合はトリガー信号を送信する。

SEABAS はトリガー信号を受信すると、USER FPGA 上のファームウェアで L1 トリガー発行処理を行い、ABCn250 1Chip Module にトリガーコマンドが送信される。

チップからの信号は SEABAS によって読み出され、TLU から供給される Time Stamp 用のクロックから生成された Time Stamp 情報を付加された上で PC に転送される。

PC 上では SCTJDAQ Software によって各測定機器の制御および受信したデータの処理が行われる。送られてくるデータは Time Stamp とトリガー発行回数の情報によって識別され、イベントビルドが行われる。

3.2 Soi EvAluation BoArd with Sitcp (SEABAS)

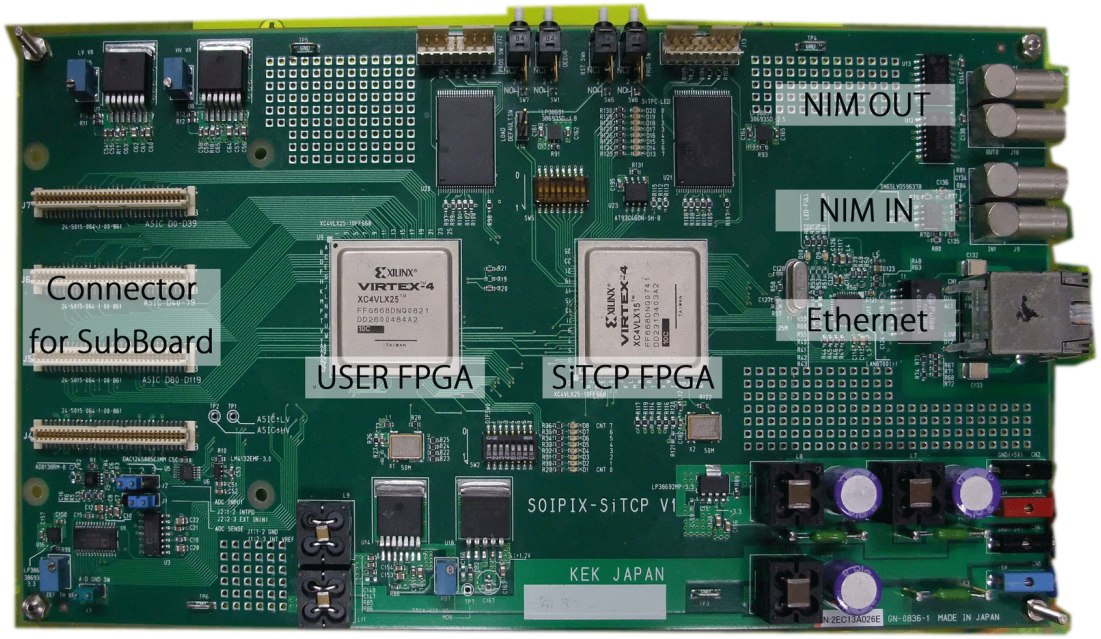

図 3.2: SEABAS ボード

Soi EvAluation BoArd with Sitcp(SEABAS) は、KEK の測定器開発室 SOI グループによって開発された汎用読み出しボードである。ネットワークプロセッサの対応速度、及び USER FPGA の仕様の違いにより SEABAS1、SEABAS2 の 2 種類が存在するが、以降特記なき限り SEABAS1 について述べることとする。

本ボードには各種測定用のサブボードを接続するための 64pin コネクター 4 個とサブボードを制御・信号処理するための USER FPGA が用意されており、使用したいデバイスに応じてファームウェアを作成すれば様々なデバイスの読み出しが可能となる。また、TCP/IP 通信でネットワークを介してデータの送受信を可能にする SiTCP[4] ファームウェアを実装した FPGA(SiTCP FPGA) を搭載し、最高 100Mbps(SEABAS1)、1Gpbs(SEABAS2) での情報伝送が可能となっている。その他の機能として、DAC、ADC、NIM 信号入出力を搭載している。図 3.2 に SEABAS の写真を示す。

[4]内田智久氏 (東京大学) によって開発されたネットワークプロセッサ

3.2.0.12 SiTCP FPGA

Xilinx Vertex4(XC4VLX15-10FF668)

100Base-T(SEABAS1)、1000Base-T(SEABAS2) に対応する SiTCP ファームウェアを搭載した FPGA である。

3.2.0.13 USER FPGA

Xilinx Vertex4(XC4VLX25-10FF668)(SEABAS1)

Xilinx Vertex5(XC5VLX50-1FF676)(SEABAS2)

120 本 (SEABAS1) の IO がサブボード用コネクタに配線されている。本 FPGA からはサブボード上のデバイスに加えて、DAC、ADC、NIM I/O の制御が可能である。

3.2.0.14 電源

本ボードは、±5V (SEABAS1)、±3.3V (SEABAS2) の直流電源を供給することにより動作する。

3.3 SEABAS 用ファームウェア

ここでは SEABAS の SiTCP FPGA、USER FPGA にそれぞれ搭載されるファームウェアについて説明する。

3.3.1 SiTCP FPGA

SiTCP には、TCP および UDP による通信機能と USER FPGA 用のインターフェースが搭載されている。

TCP 通信については TCP I/F と呼ばれる専用のインターフェースが用意されている。TCP I/F は同期 FIFO のインターフェースと同様の構造になっているため、USER FPGA からは FIFO のデータを読み書きするのと同様の方法で使用できる。SiTCP FPGA から USER FPGA 側にデータを読み出す場合は、SiTCP FPGA のデータ保持の有無を示す data valid 信号が High である時に、read enable 信号を High にしてデータを読み出す。SiTCP FPGA に USER FPGA からにデータを送る場合は、SiTCP FPGA に空き容量

があることを示す almost full 信号が Low の時に、write enable 信号を High にしてデータを SiTCP FPGA に送る。

UDP 通信については Remote Bus Control Protocol(RBCP) と呼ばれる専用のインターフェースが用意されている。PC から UDP 通信によって送信されたデータは RBCP を介して USER FPGA 上の専用のレジスタに書き込まれる。複数個用意されているレジスタにはそれぞれ固有のアドレスが割り振られており、PC からデータを送信する際にデータを書き込むレジスタに対応するアドレスを指定すると、指定されたレジスタにデータが書き込むことができる。

PC からの信号の受信、及び PC への信号の送信は、SiTCP FPGA によって自動的に処理が行われる。これらの処理を行うファームウェアは、内田智久氏 (KEK) によって開発されたもので、特別な要求がなければ変更を加える必要はない。また、SiTCP FPGA に搭載されたファームウェアのソースコードは著作権の関係上非公開であるため、改変は基本的に推奨されない。本研究でも内田氏の開発したものをそのまま用いている。

3.3.2 USER FPGA

SCTJDAQ の中で、USER FPGA が関与するのは、

1. PC から送信されたコマンドを SiTCP FPGA を介して受け取り、ABCn250 へコマンドを送信すること
2. ABCn250 から読み出された信号を SiTCP FPGA を介して PC に送信すること
3. 外部信号によるトリガー発行
4. イベントビルドに必要な情報の付加

である。これらのうち、1、2 については先行研究において同様の機能を有するファームウェアが開発されている [1][2][3]。

本研究では、遠藤理樹氏、田窪洋介氏、Sergio Gonzalez Sevilla 氏の開発した ABCn250 読出し用ファームウェアを本研究で開発する USER FPGA 用ファームウェア (以下、遠藤版ファームウェア) を基本に採用した。ただし、このファームウェアは本研究で基本として採用するにあたって以下の 3 点について対応が必要である。

ABCn250 用コマンドの送信先の切り替えについて

遠藤版ファームウェアは、BCC[5] と呼ばれるチップが ABCn250 と SEABAS の間に存在することを前提にしている。BCC は ABCn250 へのコマンド送信に関して、コマンドの宛先が自分の配下にある ABCn250 であるかどうかを判別する機能が存在しており、自分の配下にある ABCn250 が宛先である場合のみ配下にある ABCn250 にコマンドを中継する。よって、遠藤版ファームウェアではコマンドの送信先を判別せず、すべてのリンクに送信している。

本研究によって開発する DAQ システムでは、ABCn250 1Chip モジュールを BCC なしで用いるため、複数のモジュールを接続した場合には常に全モジュールにコマンドが送信されてしまう状態になる。ABCn250 自身にもコマンドの宛先が自身であるかを判別する機能が存在するが、1Chip モジュールに搭載される ABCn250 はすべて識別用のアドレスが '000000' であるため、そのままであると個別のモジュールにコマンドを送信したい場合に不都合が生じる。よって、本研究で遠藤版ファームウェアを用いることができるようにするには必要に応じてコマンドの送信先を切り替える機能を追加する必要がある。

外部からの信号によるトリガーについて

遠藤版ファームウェアは、ABCn250 の内部試験用に開発されているため、外部からの信号によってトリガーをかけることができない。よって、本研究で遠藤版ファームウェアを用いることができるようにするには外部からの信号に対応して ABCn250 に L1 トリガー信号を発行する機構を追加する必要がある。

トリガーイベントを識別するための情報付加について

遠藤版ファームウェアは、ABCn250 の内部試験用に開発されているため、読み出した ABCn250 のデータに対して個別のトリガーイベントを識別するための情報を付加していない。よって、本研究で遠藤版ファームウェアを用いることができるようにするには ABCn250 から読み出したデータにトリガーイベントを識別するための機構を追加する必要がある。

上記の 3 点を踏まえて、本研究においては

[5] Buffer Control Chip

- NIM IO を用いた外部トリガー機能
- NIM IO からのクロック供給によるタイムスタンプの生成、及び読出しデータへの付加機能
- ABCn250 の仕様に基づく Hard Reset 機能
- コマンドの送信先となる信号線を適宜切り替える機能

を追加したファームウェアを開発した。

以下、図 3.3 にファームウェアのブロックダイヤグラムを示し、各モジュールについて説明する。なお、基本としたファームウェアと同等の部分については遠藤氏の修士論文を引用している。

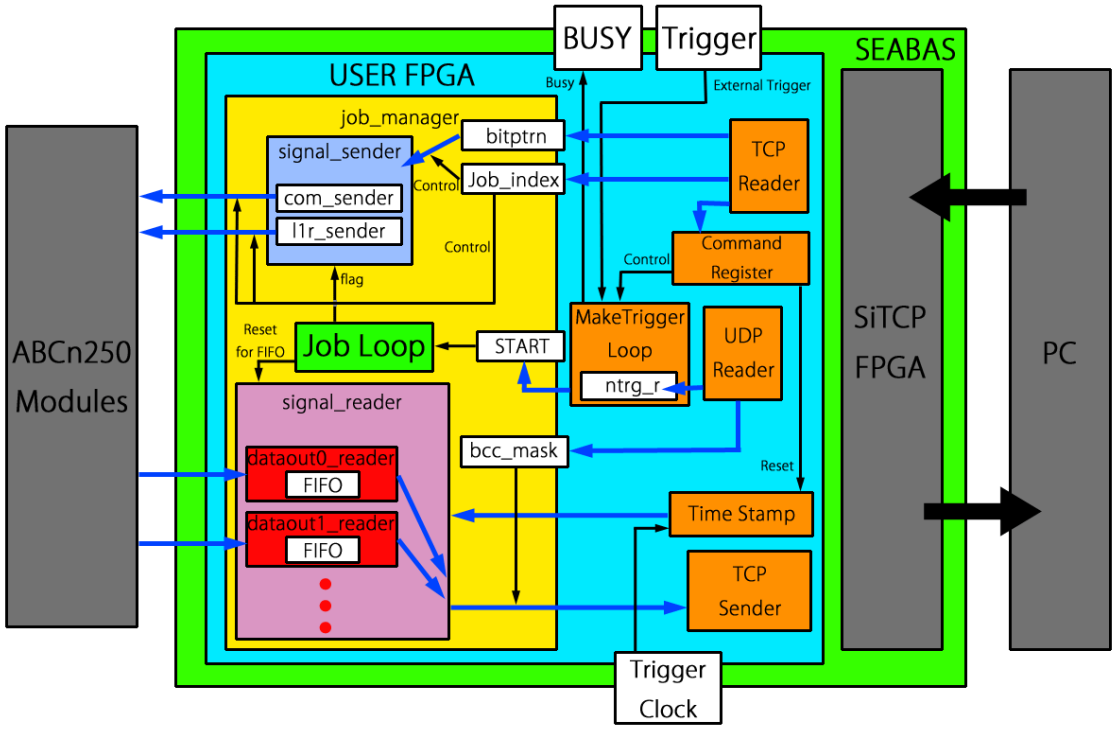

図 3.3: USER FPGA 用ファームウェア ブロックダイヤグラム

TCP reader

PC から TCP で送信されたビットストリームを SiTCP FPGA を介して受け取る。受け取ったビットストリームから、ABCn250 に送信するビットストリーム、

job_index、ファームウェア内処理用のコマンド (Command Register[6]) を抜き出す。ビットストリームは bitptrn というレジスタに保持、job_index は job_index というレジスタにそれぞれ保持する。Command Register については適宜対応するレジスタに保持される。

UDP reader

PC が送信した FIFO の読み出し制御やファームウェア上の処理で用いるパラメータを UDP 通信で受信する。bcc_mask は最大 32 個の FIFO の bitmask 用のパラメータである。bcc_mask によってマスクされた FIFO については読出しが行われない。clk_mode は ABCn250 からの出力データ読み出しを同期させるクロックの周波数を決定する。一度に自動発行できる L1 トリガーの数は ntrg_r というレジスタに保持される。

Make Trigger Loop

DAQ を開始するためのフラグ (DAQ 開始フラグ) を ntrg_r の数だけ生成するループ処理。ABCn250 に送信するビットストリームは、L1 トリガーコマンドを含めてすべてこのループにより生成されるフラグによって送信が制御される。DAQ 開始フラグは後述の job_manager に渡す。

このループは PC からのコマンドにより外部トリガーが有効化されている場合は NIM 入力からのトリガー信号によっても処理を開始させることができる。また、このループによって DAQ 処理が開始された場合、処理の終了まで NIM 出力より BUSY 信号を出力する。

signal_sender

signal_sender は、ABCn250 へビットストリームを送信する機能を持つ。l1r_sender と com_sender の二つがあり、l1r_sender は ABCn250 の L1 信号線にビットストリームを送信し、com_sender は ABCn250 の COM 信号線にビットストリームを送信する。どちらのモジュールを使うかは job_index により制御する。各 sender には ABCn250 へ送信するビットストリーム用のレジスタがあり、bitptrn の値を com_sender のレジスタと l1r_sender のレジスタのどちらに入れるかも job_index で

[6]外部トリガーの有効化、Hard Reset 機能の制御等のために用いる各種レジスタ

決定する。後述の job_manager から送信開始フラグを受け取ると信号線へのビットストリーム送信を開始する。

signal_reader

signal_reader には ABCn250 から読み出したデータ保存用の FIFO があり、1 個の signal_reader が 1 個の ABCn250 マスターチップの DATAOUT より一対一対応でデータを受け取る。

signal_reader は、dataoutX_reader(X = 0,1,2,,,,31) の 32 個あり、最大 32 個の ABCn250 マスターチップからデータを受信できる。ABCn250 から読み出されたデータは FIFO に順次送られるが、このデータの先頭には後述する Time Stamp から受け取った 24 bit のイベント情報が付加される。FIFO に保持されたデータは順次 TCP sender へ送られる。ABCn250 からの出力データの読み出しは clk_mode によって決定した周波数のクロックに同期する。

Time Stamp

Time Stamp は NIM 入力により外部から受け取ったクロック信号に同期して 24 bit のカウンター用レジスタを加算する。外部からトリガーコマンドを受け取るとその時点のカウンタ値を保持し、これを signal_reader に送る。値の Reset などは PC 側からのコマンドによって行われる。詳細な機構については後述する。

job_manager

job_manager は、signal_sender や signal_reader を制御する USER FPGA の動作の要となるモジュールである。TCP reader から受け取った job_index の値に基づき com_sender か l1r_sender のどちらを使用するか選び、なおかつ、選んだ方の送信ビットストリーム用のレジスタに bitptrn に保持されているビットストリームを送る。job_manager は 40MHz のクロックに同期するループ (Job Loop) を持っている。Job Loop は DAQ 開始フラグを受け取ると、dataoutX reader の FIFO リセット用フラグの作成や、com_sender や l1r_sender のビットストリーム送信開始フラグの作成を制御する。また、bcc mask の値に対応した signal_reader の FIFO に入っているデータを順番に TCP sender へ送信する。

TCP sender

TCP sender は、最大 32 個の dataoutX_reader の FIFO に入っているデータを SiTCP FPGA を介して TCP 通信で PC へ送信する。dataoutX_reader の FIFO が読み出し可能、かつ SiTCP FPGA と PC が通信確立、かつ SiTCP FPGA に空き容量がある場合にデータを送信する。

なお、基本としたファームウェアは最大 32 個の ABCn250 マスターチップを同時に読み出すことができるが、本研究における DAQ システムにおいては使用する読出し用サブボードが最大 8 個までの ABCn250 モジュールが接続できる仕様であることから、signal_reader の FIFO についても 8 個分のみを使用している。

3.4 Trigger Logic Unit (TLU)

Trigger Logic Unit (TLU) は第 1 章で述べた通り、各測定機器の状況を把握し、トリガー信号の送出を制御するものである。本研究においては Spartan 3AN Starterkit を用いて次の機能を備えた TLU を製作することとした。

トリガー信号送出制御

TLU がトリガー信号を受け取った際に、DAQ システムが次のイベント取得が可能かどうかを判断し、イベント取得が可能な時にのみ各機器にトリガー信号を送出する。イベント取得の可否については各測定機器が出力する BUSY ステートに基づいて行なう。各測定機器のステートは NIM 信号によって受け取る。

同期用クロック信号送出

各測定機器からの信号を一つのイベントとして処理するために、イベント識別用の情報を付加する必要がある。このイベント識別用の情報を生成する元となる共通のクロック信号を各測定機器に NIM 信号として供給する。

3.4.1 Spartan 3AN Starterkit

Spartan 3AN Starterkit は Xilinx 社から発売されている FPGA 評価用ボードである。VGA ディスプレイやシリアルポートなどの多様な I/O デバイスを搭載し、FPGA アプリケーションの開発用ボードとして用いられる。

Spartan-3AN FPGA を搭載し、Ethernet コントローラを備えていることから、SiTCP を書き込んで用いることが可能である。また、FX2 拡張コネクタを使用して機能拡張が可能である。

図 3.4 に Spartan 3AN Starterkit の写真を示す。

3.4.2 拡張ボード

本研究で作成する TLU には NIM 入出力が必要であるが、Spartan 3AN Starterkit には NIM 入出力が存在せず、また Spartan 3AN FPGA は NIM 信号の規格に対応しない。そこで、NIM 信号を LVCMOS 信号に変換する機能を備えた FX2 接続の拡張ボードを作成し、Spartan 3AN Starterkit に接続して用いることとした。

図 3.4: Spartan 3AN Starterkit

拡張ボードの回路構成は SEABAS1 の NIM 入出力部の回路 [9] を参考にし、図 3.5 のような回路を作成した。NIM 入出力の必要数については、DAQ システムに接続される測定機器の数によって変化するが、今回作成したものは測定機器 1 台の時の必要最小限の構成に合わせて回路を構成している。必要な NIM 入出力が増える場合には合わせて回路を増設することで容易に対応が可能である。

なお、図中からは省略しているが、本ボードの動作のための電源については、+5V、+3.3V については Spartan 3AN Starterkit からの供給または外部電源からの供給を選択、-5V については外部電源からの供給によって賄う。

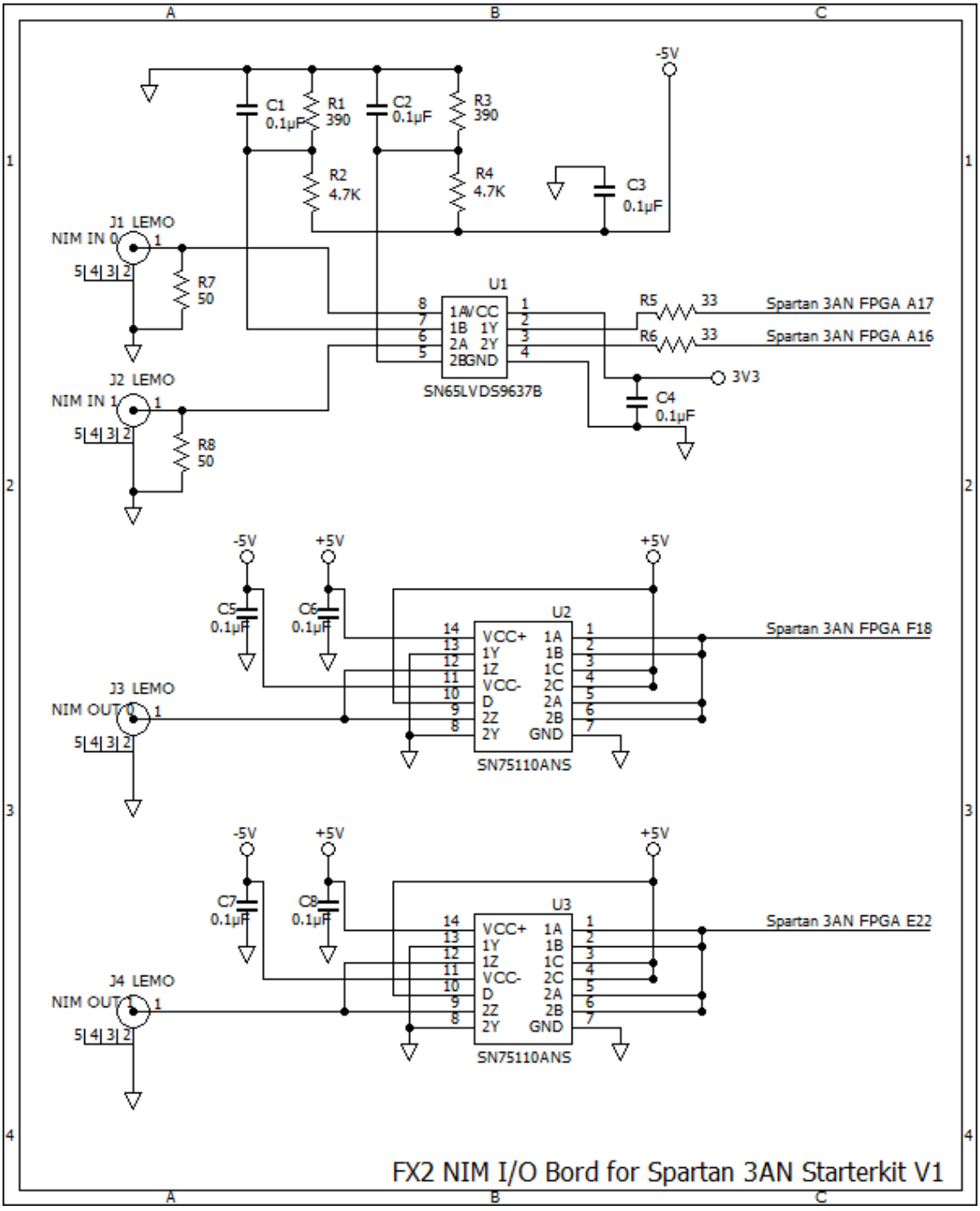

図 3.5: Spartan 3AN Starterkit 用 NIM 入出力拡張ボード 回路図 (初期版)

初期版の回路構成に基づいて実際に拡張ボードを作成した。作成した拡張ボードの写真を図 3.6 に示す。

Spartan 3AN Starterkit に接続する前のテストにおいては当初は期待される動作をし

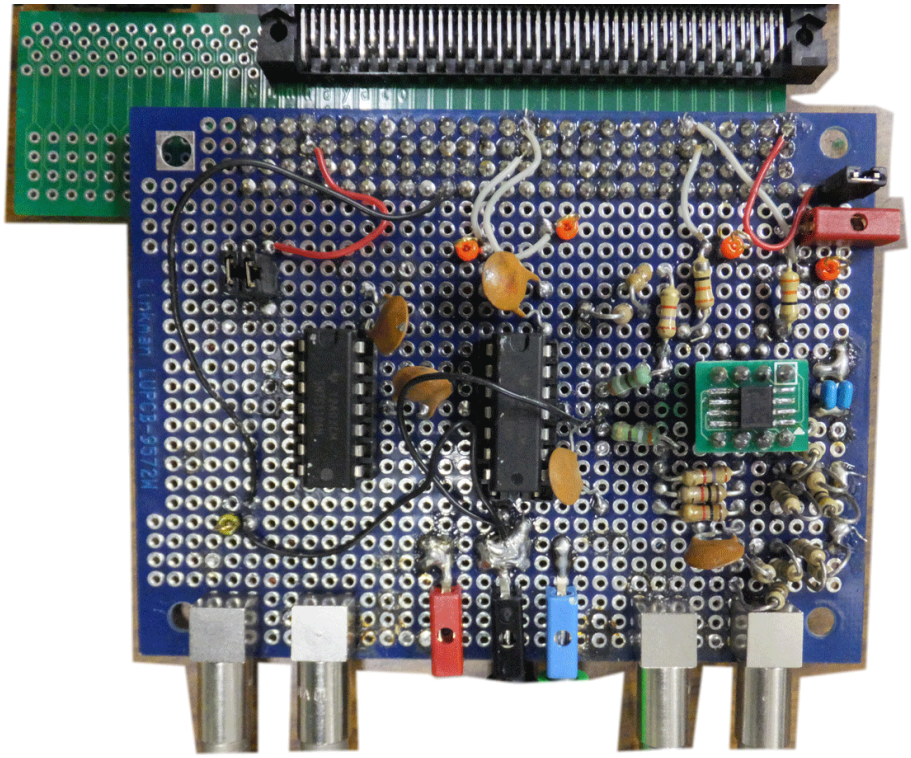

図 3.6: Spartan 3AN Starterkit 用 NIM 入出力拡張ボード

ているように思われたが、NIM 入力について、連続して信号を入力すると FPGA 側に出力される信号が不定値になる問題が確認された。IC に供給される電源の出力には問題はなく、また NIM 信号の IC よりコネクタ寄りまでは正常に信号が届いていることから、チップ自身の動作に何か問題が生じていることが疑われた。

本ボードにおいて、NIM 信号から LVCMOS 信号に変換を行う SN65LVDS9637B は、入力される NIM 信号 (NIM IN 0 は 8 ピン、NIM IN 1 は 6 ピン) と基準電位 (7、5 ピン) の差を取り、差 (8 ピン-7 ピン、6 ピン-5 ピン) が-32mV 以上である時に Hi、-100mV 以下である時に Low を Output 側に出力する。いずれの条件にも当てはまらない場合は不定値となるため、発生した問題は入力される NIM 信号か基準電位を供給する回路 (7、5 ピンに接続されている回路) のいずれかの電位が想定されている範囲を外れていると考えられた。NIM 信号の IC よりコネクタ寄りまでは正常に信号が届いていることを確認しているため、基準電位を供給する回路に問題が有ることが疑われた。基準電位を供給する回路は GND - -5V 電源間の電位を分割し、$-5\mathrm{V} \times 390\Omega/(390\Omega + 4700\Omega) = -38\mathrm{mV}$ を供給している。390Ω の抵抗には電位を安定させるために並列にコンデンサが接続している [7]

[7] 参考にした SEABAS1 の NIM 入出力も同様の仕様である

が、電圧供給先の負荷が大きくない場合は無くても動作する。そこで、基準電位を供給する回路からコンデンサ (図 3.5 中の C1、C2) を除去し、改めて試験を行った。コンデンサ除去後の回路を図 3.7 に示す。

コンデンサ除去後は正常に動作するようになった。詳細な原因は不明であるが、除去したコンデンサが悪影響を与えていたと考えられる。

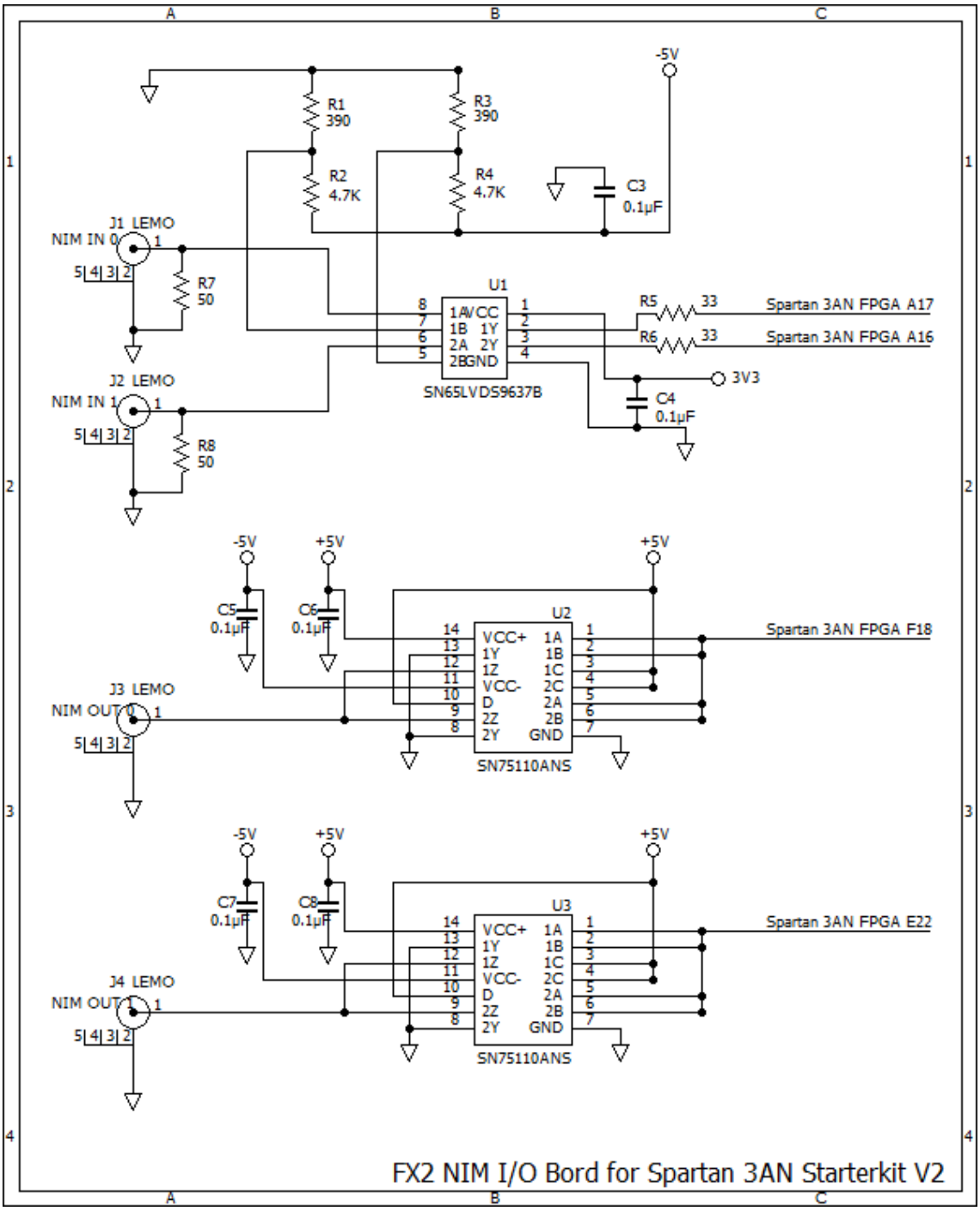

図 3.7: Spartan 3AN Starterkit 用 NIM 入出力拡張ボード 回路図 (改良版)

3.4.3 TLU 用ファームウェア

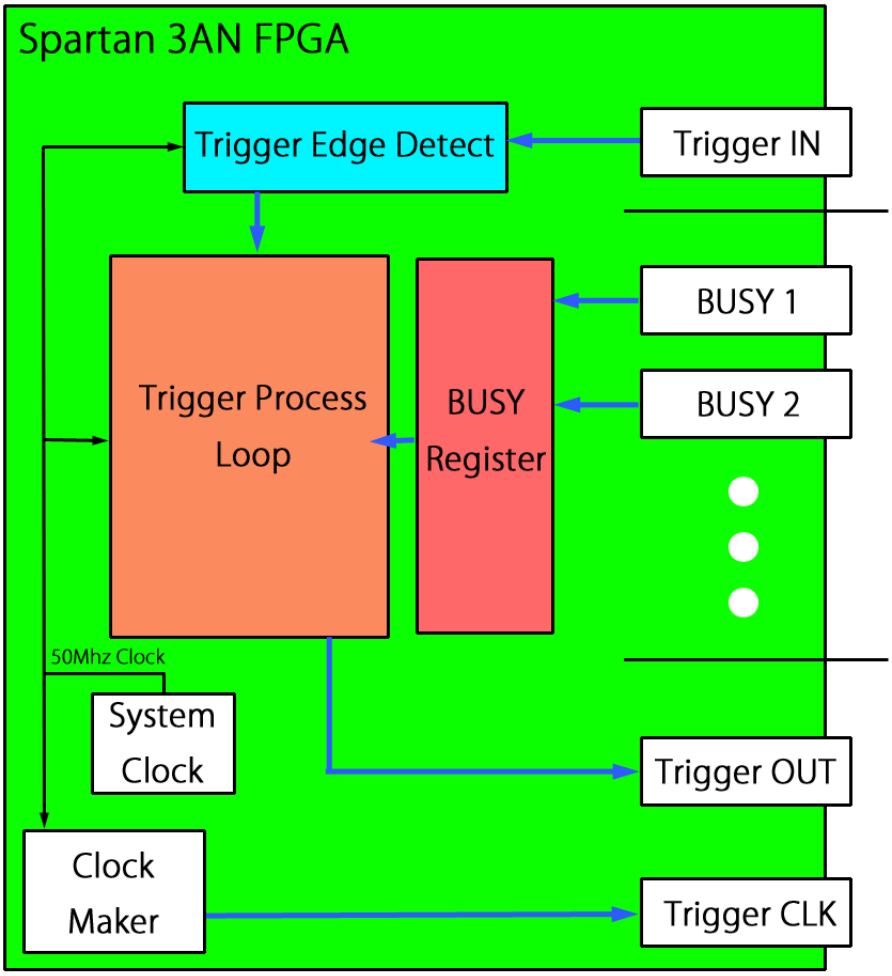

図 3.8: TLU 用ファームウェア ブロックダイヤグラム

図 3.8 は TLU 用に作成したファームウェアのブロックダイヤグラムである。以下、図中の各モジュールについて説明する。

Trigger Edge Detect

NIM 入力 (Trigger IN) を介して受信されたトリガー信号は、このモジュールによって処理が行われる。このモジュールは 2 bit のレジスタを持ち、50Mhz のシステムクロックの立ち上がり時に下位ビットの値を上位ビットにシフトした後 Trigger IN の Hi/Low をレジスタの下位ビットに記録する。トリガー信号の立ち上がり時には必ず 2 bit のレジスタは '01' になるので、これを用いてトリガー信号の入力を検知

する。トリガー信号の入力を検知した際には後述する Trigger Process Loop にトリガー検知フラグを渡す。

BUSY Register

各測定機器からの BUSY 信号はこのレジスタに記録される。BUSY 信号が Hi であるとき、対応するレジスタの値が'1' となる Trigger Process Loop がトリガー検知フラグを受け取った際には、このレジスタの値がすべて '0' であることを確認することによって各測定機器が次のトリガーを受信可能であることを確認する。

Trigger Process Loop

各測定機器にトリガー信号を送る処理はこのモジュールによって行われる。トリガー検知フラグを受信した際に、BUSY Register がすべて '0' になっていれば、トリガー出力用 NIM 出力 (Trigger OUT) を 100 ns 間 Hi にする。続いて 300 ns 間 Trigger OUT を Low にした後、次のトリガー受信を待ち受ける。

Clock Maker

本 TLU は各測定機器に同期用クロック信号を送出する。このモジュールではシステムクロックから 100kHz のクロック信号を生成し、クロック信号用 NIM 出力 (Trigger CLK) から出力する。

3.5 Time Stamp

Time Stamp はビームテストにおいてトリガーイベントを識別するための情報の一つである。複数の測定機器からのデータを 1 つのトリガーイベントとして統合する際には、この Time Stamp およびトリガーの発行回数が同一であることによって同一のトリガーイベントによるデータであることを確認する。

本研究によって開発した DAQ システムにおいては、SEABAS 上の USER FPGA 用ファームウェア内の Time Stamp モジュールで、TLU から供給される 100 kHz のクロック信号によって幅 24 bit のカウンタを加算し、このカウンタの値を Time Stamp として使用している。トリガー信号を受信した際にはその時点のカウンタ値を保持し、ABCn250 から読み出したイベントデータの先頭にビットストリームとして付加することによってイベントデータを識別できるようにしている。

本 DAQ システムに新たな測定機器を追加する場合には、同様の機構を測定機器に組み込むことによって Time Stamp の機能を持たせることとする。

3.5.1 Time Stamp の測定機器間の同期

SCTJDAQ においては、TLU から各機器に配信されるのは Time Stamp を動かすためのクロック信号であり、Time Stamp の値そのものは各機器が独自に生成することになる。よって、Time Stamp の値そのものを各機器が共有しないため、ビームテストの Run 中にエラーが生じると、Time Stamp 情報に不整合が生じる可能性がある。そのため、複数の測定機器の Time Stamp の値を何らかの方法で同期する機構が必要になる。本来は TLU から各測定機器にリセット信号を配信し、同じタイミングでカウンタ値をリセットするようにする方法が最も簡易であるが、本研究において使用する SEABAS1 は NIM 入力が 2 系統のみであり、トリガー信号の入力と Time Stamp 用のクロック信号で既にすべて使用してしまっているため、別のリセット機構を設ける必要があった。そこで、以下の 2 種類のリセット機構を設け、Time Stamp の同期が行えるようにした。

Command Reset

このリセットは PC 側から Time Stamp のリセットを指示するコマンドを受け取ることによって任意のタイミングで実行されるリセットである。このリセットが実行されると、Time Stamp 用カウンタの値は初期化され、カウンタの加算処理が停止する。この状態で各測定機器が L1 トリガーコマンドを受信した場合、そのトリガーの時点を ‘0’ としてカウンタの加算処理を開始する。

このリセットを DAQ においてデータ取得を開始する直前 (Start 時、Run の切り替わり時) に実行することで、各測定機器が最初の L1 トリガーコマンド時を ‘0’ として同時にカウンタの加算処理を開始するため、Time Stamp のカウンタ値を同期させることができる。

Auto Reset

このリセットは Time Stamp 用カウンタの値が最大値に達する毎に実行される自動的なリセットである。カウンタは幅 24 bit の 2 進数のレジスタであるので、すべてのビットが ‘1’ になるとき、すなわち最大値は 10 進数換算で 16,777,215 カウントとなる。

TLU から供給されるクロック信号が 100 kHz である場合、$16,777,215[\mathrm{Count}]/100[\mathrm{kHz}] \simeq 168[\mathrm{s}]$ でカウンタ値は最大値に達することになる。

カウンタの値が最大値に達した場合はカウンタをリセットして加算を再開することになるが、この時に、直ちに加算を再開するのではなく、Command Reset と同様に一度カウンタの加算処理を停止し、最初の L1 トリガーコマンド時を '0' としてカウンタの加算処理を再開させるようにすれば、トリガーレート以下のズレが生じている場合でもこのリセット後には修正されるようにすることができる。トリガーレートを 1kHz とすると、$100[\mathrm{kHz}]/1[\mathrm{kHz}] = 100[\mathrm{Count}]$ 程度のズレまでは自動的に修正できることになる。

3.5.2 データ構造

Time Stamp の情報は、ABCn250 の読出しシステムにおいてはビットストリームから 16 bit さかのぼった位置に付加される。このとき、Time Stamp 情報は表 3.1 に示す、ヘッダとトレーラーを含む 40 bit のビットストリームとなる。

Time Stamp Bitstream (40 bit)		
Header	Time Stamp	Trailer
10011100	24 bit Data	10000000
8 bit	MSB First	8 bit

表 3.1: Time Stamp 情報ビットストリーム構造

3.5.3 ABCn250 読出し用ファームウェアにおける実装結果

Time Stamp を付加する機能を ABCn250 読出し用ファームウェアに実装し、SCTJDAQ Software で読み出せることを確認した。図 3.9 は SCTJDAQ Software の ABCNReader モジュールで Time Stamp を取得したときのログをキャプチャしたものである。

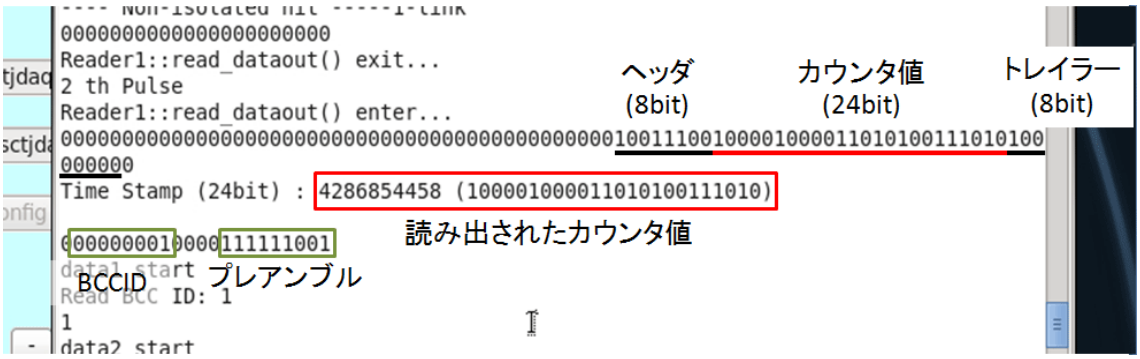

図 3.9: Time Stamp の読出し結果のキャプチャ

3.6 DAQ ソフトウェア (SCTJDAQ Software)

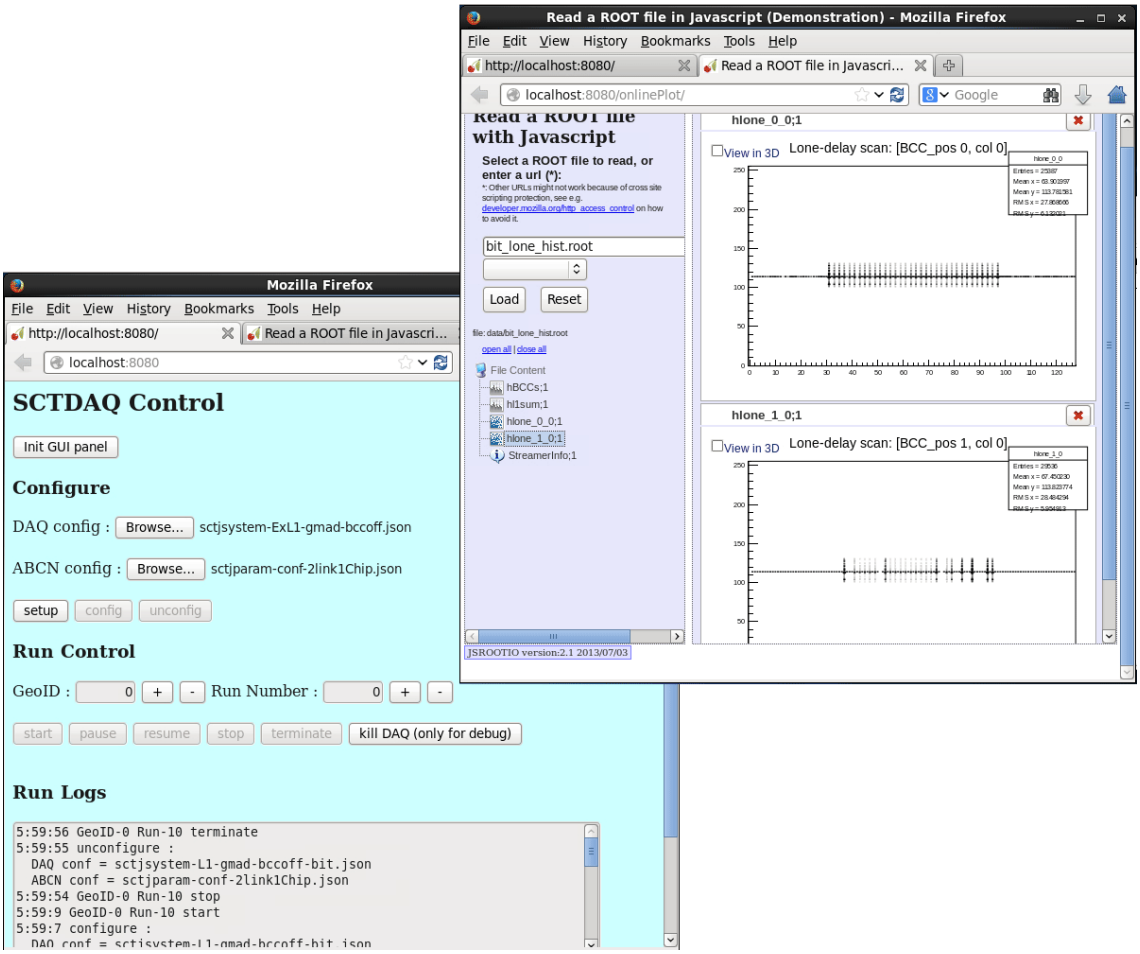

図 3.10: SCTJDAQ Software 動作イメージ

本研究において、各機器を制御し、データを処理するための DAQ 用ソフトウェア (SCTJ-DAQ Software) を新たに開発した。本 DAQ 用ソフトウェアの開発に当たって、ABCn250 読出し用 DAQ システムの制御には、制御用ソフトウェアについても遠藤氏他の開発したソフトウェアのソースコードを一部流用している [8]。DAQ ソフトウェアのフレームワークについては安氏に協力を頂いた。GUI については久保田氏、山本氏と共同で開発を行なったものである。図 3.10 に SCTJDAQ Software の動作イメージを示す。

[8] ファームウェアについて遠藤氏他の開発したファームウェアを元に機能追加したものを採用したため、コマンド処理体系がほぼ同一であることから

3.6.1 SCTJDAQ Software 概要

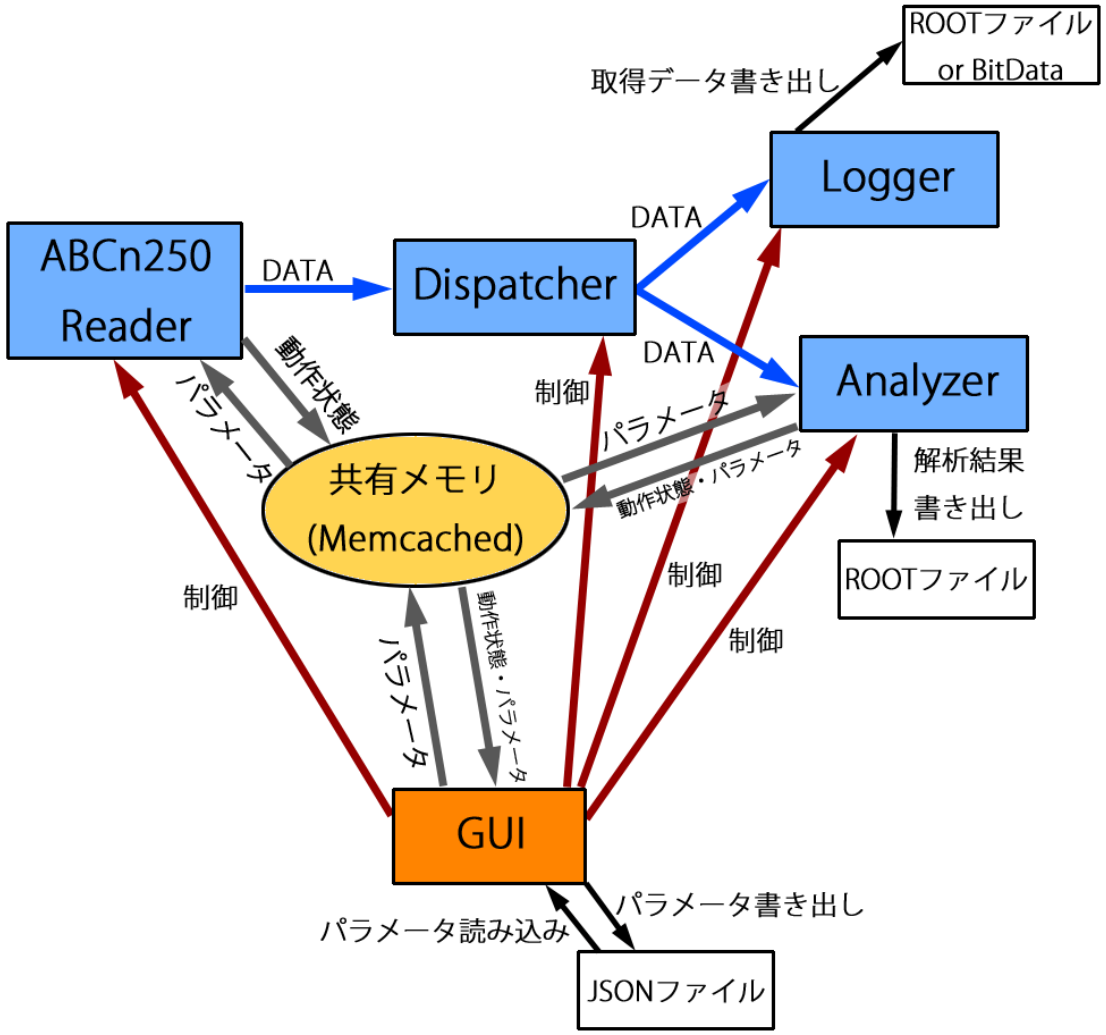

図 3.11: SCTJDAQ Software モジュール間データ・コマンドパス概要 (疑似パルステスト時)

図 3.11 に SCTJDAQ Software の概要を示す。図に示す ABCn250 Reader、Dispatcher、Logger、Analyzer はそれぞれ独立して動作するソフトウェアモジュールである。これらのソフトウェアモジュールは C++によって記述され、一部のモジュールは ROOT[9] を処理に使用している。また、GUI はエンドユーザーが SCTJDAQ を制御するための窓口にあたるモジュールである。各モジュールのプロセス間は Message Queue によって結ばれ

[9]CERN によって開発が行われている、データ解析環境および関連ライブラリ群

ており、GUI から各モジュールを制御するためのコマンド用、データ送受信用の 2 系統の経路が用意されている。各測定機器を制御するためのパラメータは JSON ファイルに格納されており、GUI が DAQ 開始前に Memcached(後述) による共有メモリ上に展開し、各モジュールが適宜取得する。また、共有メモリ上には、一部モジュールの動作状態についての情報も配置される。

以下に、各ソフトウェアモジュールについて説明する。

GUI

GUI はエンドユーザーが SCTJDAQ を制御するための窓口である。このモジュールは

- フレームワークを介して各ソフトウェアモジュールを制御
- 各測定機器を制御するためのパラメータのような各モジュールが必要とするデータを共有メモリ上に展開する
- テストの結果共有メモリ上の制御パラメータが変更になった場合は変更後の値を JSON ファイルに出力する

の機能を担う。本モジュールは Python、JQuery、HTML によってコーディングされており、CherryPy[10] によって提供される Web サーバを介して Web ブラウザ上から操作を行うことができるようになっている。

ABCn250 Reader

本モジュールは、ABCn250 が接続された SEABAS ボードを制御し、ABCn250 からのデータを受信するモジュールである。GUI が共有メモリ上に展開したデータを用いて ABCn250 へのパラメータ書き込みを行い、取得した ABCn250 からのデータビットストリームを処理し、1 トリガー毎のデータに構成した後 Message Queue を介して Dispatcher にデータを出力する。

他の測定機器が追加された場合には、それに対応する同様のソフトウェアモジュールを追加することになる。

[10]Python プログラミング言語を用いたオブジェクト指向の Web アプリケーションフレームワーク

ABCn250 用の制御・読出しモジュール独自の機能として、ABCn250 へのパラメータ書き込みが正常に行われたかを確認するため、ABCn250 からパラメータを読出し、書き込んだパラメータと照合する機能がある。

Dispatcher

本モジュールは、ABCn250 Reader モジュールから受け取ったデータを後段の Logger、Analyzer モジュールに送信するモジュールである。

Logger

本モジュールは、Dispatcher を介して ABCn250 Reader モジュールから受け取ったデータを ROOT Tree(又は BitData) としてファイルに出力する。

Analyzer

本モジュールは、Dispatcher を介して ABCn250 Reader モジュールから受け取ったデータを実行している試験の内容に応じて解析し、その結果をヒストグラムデータとして ROOT ファイルに出力する。

図 3.11 は第 4 章で述べる疑似パルスを用いた ABCn250 単独での動作テスト時の構成である。他の測定機器が加わる際には、Dispatcher の前にイベントビルドを行うモジュールを追加するなど、一部モジュールに組み替えが生じる。

以下に SCTJDAQ Software の開発に当たって核となった要素について説明する。

3.6.2 マルチプロセス・モジュール化

先行研究で開発された DAQ システム [1][2][3] においては、DAQ ソフトウェアは OS 上の単一プロセスによって動作し、試験毎に個別のソフトウェアが新たに開発されていた。そのため、

1. DAQ 制御・データ取得・データ解析等の各種処理が同時に実行できないことによる実行速度の低下
2. ソフトウェアが巨大化・複雑化したために機能追加や修正が難しくなる
3. ソフトウェアが個々の試験に特化したものになっていることに伴う汎用性の低下

といった問題が生じている。

そこで、本研究においては、DAQ 制御・データ取得・データ解析等の各種処理について、独立したソフトウェアモジュールに分割し、

1. 並列処理による実行速度の向上
2. モジュール化によるソフトウェア構造の単純化
3. 必要に応じてモジュールを組み替え可能とすることによる汎用性の向上

を目指すこととした。

3.6.3 Configuration File

SCTJDAQ Software においては、使用するソフトウェアモジュールの指定、ABCn250 の設定パラメータ等の設定情報はすべて JSON [11] で記述する。従来の DAQ ソフトウェアにおいては、設定情報については XML またはプレーンテキストで記述することが多かったが、SCTJDAQ Software では GUI やフレームワークに Python を使用しており、JSON で記述された情報であれば Python からは読み込むだけでそのまま Python 内のオブジェクトとして使用でき、利便性が高いことから JSON を採用した。

3.6.4 Message Queue

SCTJDAQ Software の個々のソフトウェアモジュールは独立したプロセスとして動作している。Message Queue は各プロセスが通信するためのソフトウェアコンポーネントであり、送信側と受信側が同時にやり取りをする必要がない非同期型通信プロトコルを提供する。この特徴から、SCTJDAQ Software においては、フレームワークを介したモジュールのステート変更、測定機器から読み出したデータのプロセス間受け渡しに使用している。

3.6.5 Memcached

SCTJDAQ Software において各プロセスでやり取りされるデータには、ABCn250 の設定パラメータのように、複数のプロセスで共有するもの、繰り返し利用する物が存在する。

[11] JavaScript Object Notation。軽量なデータ記述言語の 1 つ

前述した Message Queue は送信側と受信側が 1:1 で対応しており、受信側が一度データを受け取ってしまうとキャッシュからは削除されてしまうため、設定パラメータのような用途には使用できない。そこで、これらのデータのやり取りには汎用の分散型メモリキャッシュシステムである Memcached を利用する。この Memcached は元々データベースのキャッシュに用いるために開発されたものであり、キャッシュされる値に対して値を呼び出すためのキーが 1:1 で対応する。また、一度キャッシュされた値は同じキーに対して新たな値が書き込まれない限り繰り返し呼び出すことができる。この特徴から、SCTJDAQ Software においては、ABCn250 をはじめとする測定機器の設定パラメータの受け渡し、各モジュールの動作状況を他のモジュールへ通知するのに用いる共有メモリに使用している。

第4章　試験方法とその結果について

構築したSCTJDAQについて、以下のような試験により動作を確認した。

基本とした ABCn250 用 DAQ システムとの結果比較

ABCn250には内部試験用に自分でFront EndにCalibration用の信号を入力する機能が存在する。この機能を用いて、内部試験におけるパラメータの適切な設定を調整する機能が従来のABCn250用DAQ用ソフトウェアには用意されている。SCTJDAQ Softwareにも同等の機能を搭載し、L1 Delay Test、Strobe Delay Test、Threshold Scan Testの3つの試験について本研究の開発の基本としたABCn250用DAQシステムによる試験結果と比較して正常に読出しおよびデータ処理ができていることを確認する。

以下にそれぞれの試験内容について説明する。

L1 Delay Test

L1トリガー信号を受け取ったタイミングからパイプラインをどれだけさかのぼるかを決める値をL1 Delayと呼ぶ。Calibration用の信号入力→L1トリガーコマンド送信をL1 Delayの値を変更しながら繰り返し、検出効率が最も高くなるL1 Delay値を探す試験がL1 Delay Testである。

Strobe Delay Test

Calibration用の信号を、ABCn250の動作クロックに対してどれだけ遅延させるかを決める値をStrobe Delayと呼ぶ。Calibration用の信号の入射時刻とABCn250の動作クロックの立ち上がりの相対的な時間差が検出効率に与える影響を調べ、Calibration用の信号入力のタイミングを調整するのがStrobe Delay Testである。

Threshold Scan Test

ABCn250は、Front Endに入力された信号を増幅・整形した信号について、

ヒットであるか否かを判定する閾値 (Threshold) をレジスタ書き込みにより変更することができる。この閾値を変更しながら Calibration 用の信号入力→ L1 トリガーコマンド送信の作業を繰り返し、検出効率が 50%になる閾値を探すのが Threshold Scan Test である。

複数のモジュールからの読出しについての試験

本研究において、開発した ABCn250 読出し用ファームウェアは基本とした遠藤版ファームウェアに存在しなかったコマンドの送信先切り替え機能を追加している。本機能が正しく動作しているかを確認するため、複数の ABCn250 1Chip モジュールを接続して同時に制御・読出しを行う試験を行う。

試験方法は ABCn250 の 1Chip モジュールのキャリブレーション試験を 2 台のモジュールについて同時に行うもので、取得した結果が 1 台の試験の結果 (前項の試験の結果) と比較することによって開発した ABCn250 用 DAQ システムのファームウェアが複数のチップを同時に正しく制御・データ取得できることを確認する。

疑似パルスによる信号の読出し試験

ABCn250 の 1Chip モジュールに実際のセンサー信号の代用として疑似パルスを入力し、SEABAS に外部からトリガー信号を入力して L1 Delay の値を変更しながらデータ取得を繰り返す。トリガー信号に対する疑似パルス信号の遅延時間を $-1 \sim 4\mu$s 変更し、取得されたヒットデータが遅延時間に対応して変化するかを確認することによって正しくデータが取得できていることを確認する。トリガー信号、疑似パルス信号の信号源はいずれも周波数 1 kHz、信号幅 100 ns の NIM 信号を使用する。試験の概略図を図 4.1 に示す。

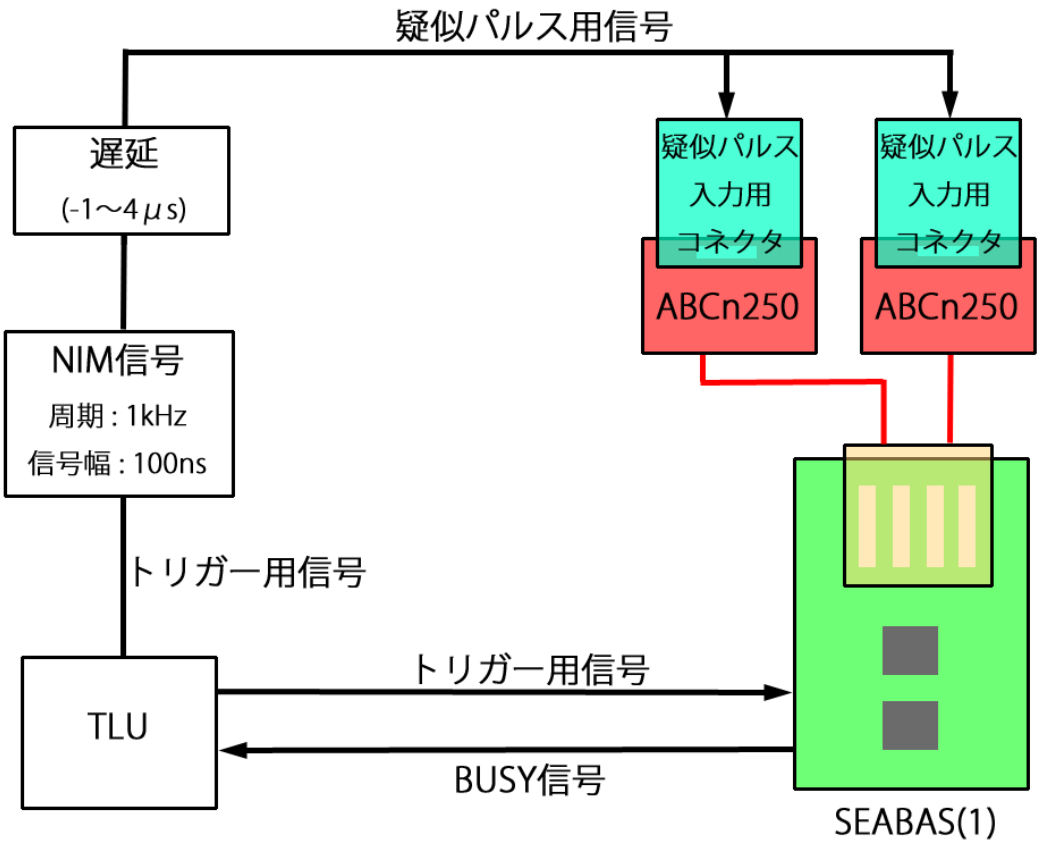

図 4.1: 疑似パルスによる信号の読出し試験の概略

疑似パルスの入力には図 4.2 の疑似パルス入力用コネクタを ABCn250 1Chip モジュールのセンサー用コネクタに接続し、疑似パルス入力用コネクタに NIM 信号を入力することによって行う。コネクタ部上面の銅テープが下面のコネクタピン (図中の ‘Connector’) 部分の直上に位置する。

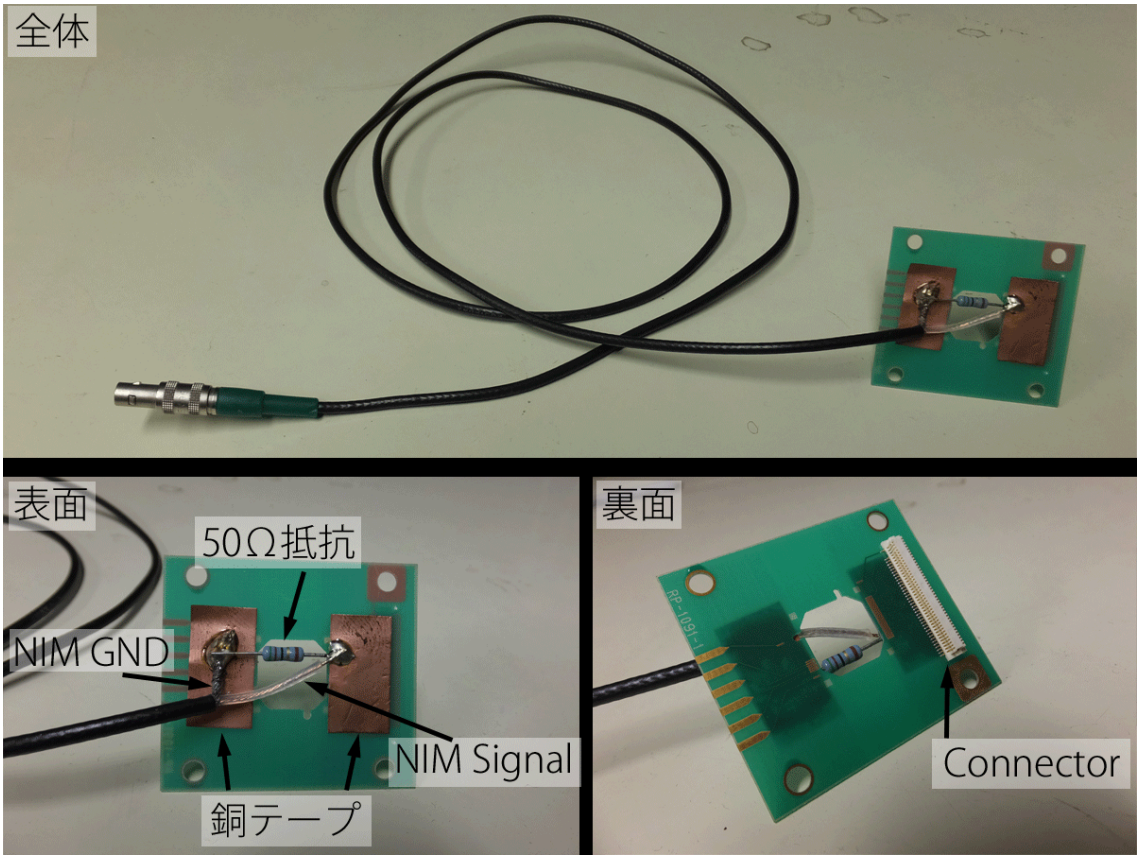

図 4.2: 疑似パルス入力用コネクタ。図上が全体図、左下がコネクタ部上面、右下がコネクタ部下面。

このコネクタは図 4.3 のような配線となっている。この回路は一種の積分回路となっており、ABCn250 1Chip モジュールに接続するためのコネクタピンがコンデンサーの一端を担う形になる。コンデンサーの構造上容量が不明であるため NIM 信号を入力した際に生じるパルス信号の大きさは不明であるが、ABCn250 によって検出できる大きさの信号であることは従来の DAQ システム [2] によって確認済みである。

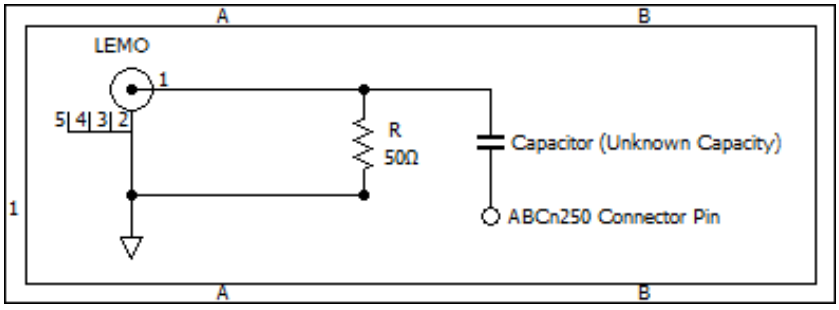

図 4.3: 疑似パルス入力用コネクタの配線図

なお、各試験において試験によって変更するパラメータ以外については固定値 (表 4.1) を使用している。

パラメータ	値 (換算値)	備考
L1 Delay	113 (2.825 μs)	
Strobe Delay	10 (10 ns)	
Threshold(Vthn)	50 (-160 mV)	新型 SCT の信号は負電位を取るため、閾値も負の値となる。

表 4.1: DAQ 動作試験時の ABCn250 パラメータ

以下、いずれの DAQ システムにおいても、SEABAS1 および ABCn250 接続用のサブカードについては同一のものを使用している。

試験には 2 台の ABCn250 1Chip モジュールを使用した。以下、1 台をモジュール A、他方をモジュール B と述べる。モジュール A、B は同一の仕様であるが、モジュール B については Ch 50 と Ch 74 について何らかの原因により応答がないため、以下に示す結果においてもすべて当該ストリップ分の情報が欠けている。

4.1 基本としたABCn250用DAQシステムとの結果比較

モジュール A を使用して、SCTJDAQ・基本とした ABCn250 用 DAQ システム双方で試験を行なった。

4.1.1 試験の結果

以下、図 4.4〜図 4.9 に各試験の結果を示す。本研究において開発した DAQ システムによる結果には SCTJDAQ、基本となった DAQ システムによる結果には Base DAQ を記す。

L1 Delay Test

本試験における L1 Delay の Scan Range は 100 (2.500μs)〜130 (3.250μs) である。
また、ヒストグラムの縦軸は L1 Delay、横軸はストリップのチャネルである。

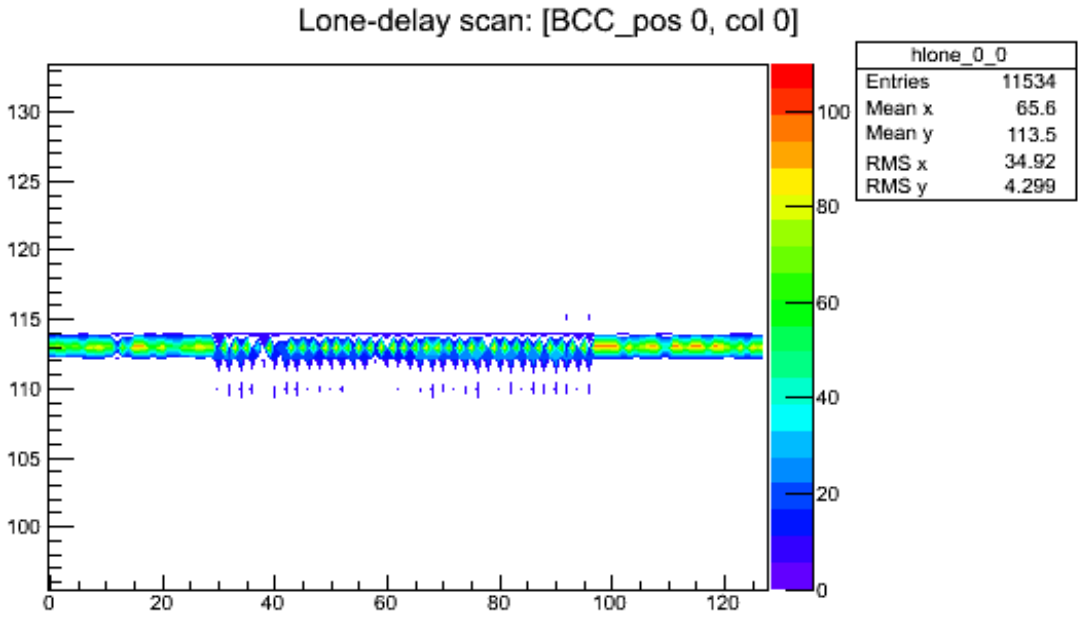

図 4.4: L1 Delay Test (SCTJDAQ)

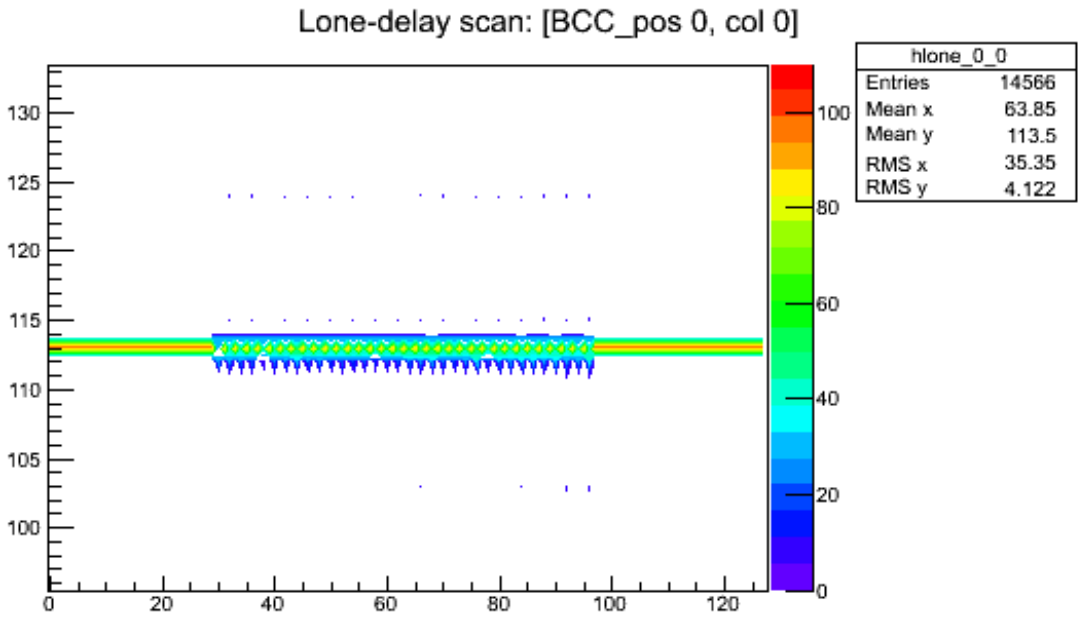

図 4.5: L1 Delay Test (Base DAQ)

Strobe Delay Test

本試験における Strobe Delay の Scan Range は 0 (0 ns)〜63 (63 ns) である。また、ヒストグラムの縦軸は Strobe Delay、横軸はストリップのチャネルである。

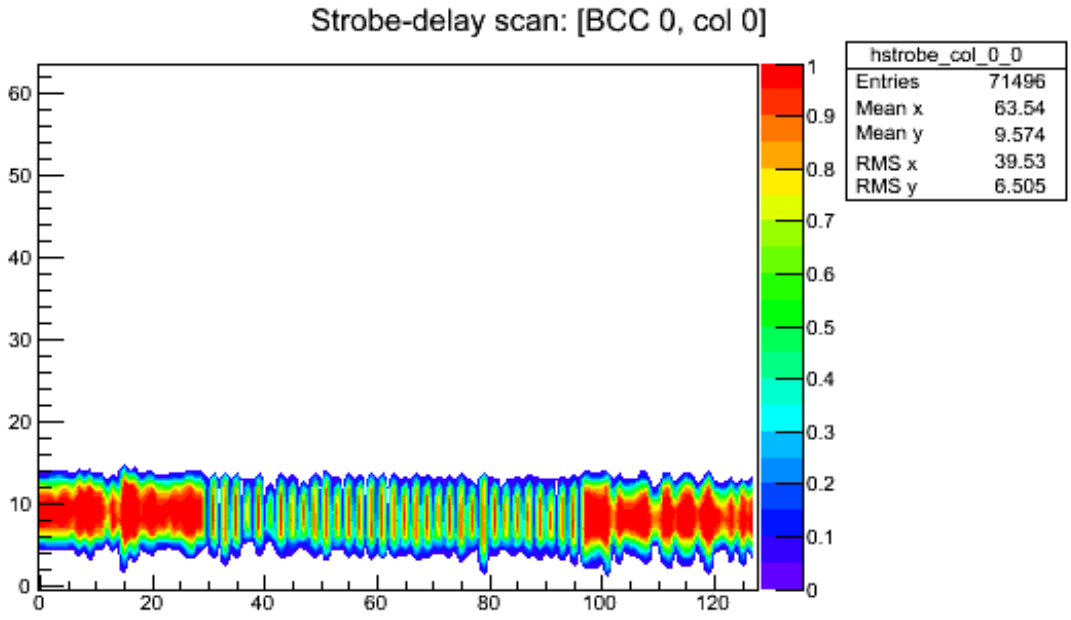

図 4.6: Strobe Delay Test (SCTJDAQ)

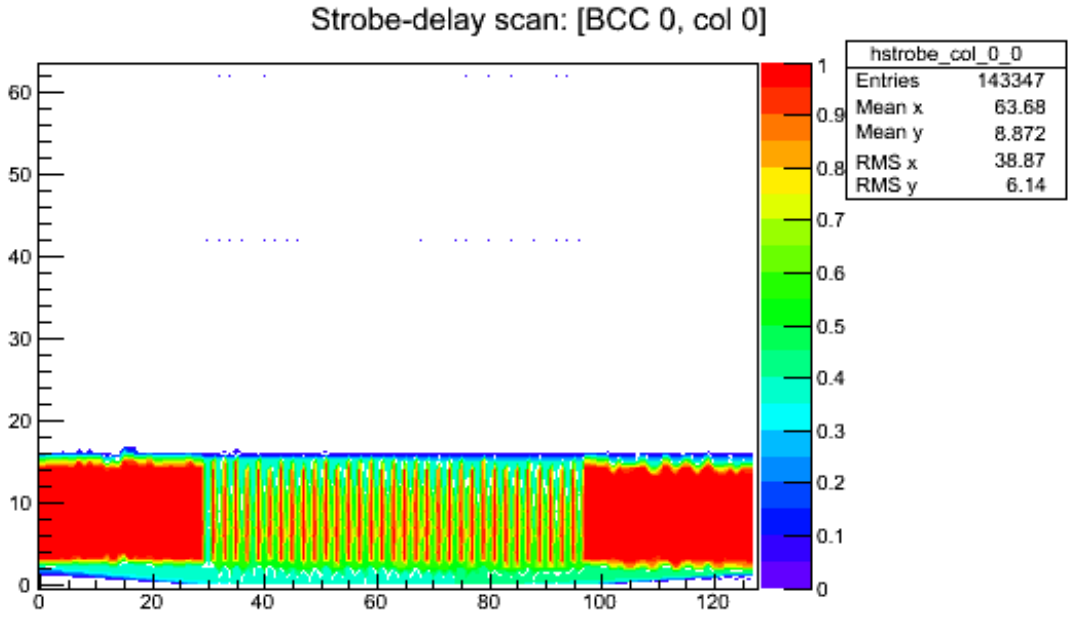

図 4.7: Strobe Delay Test (Base DAQ)

Threshold Scan Test

本試験における Threshold の Scan Range は 30 (-96 mV)〜120 (-384 mV) である。また、ヒストグラムの縦軸は Threshold、横軸はストリップのチャネルである。

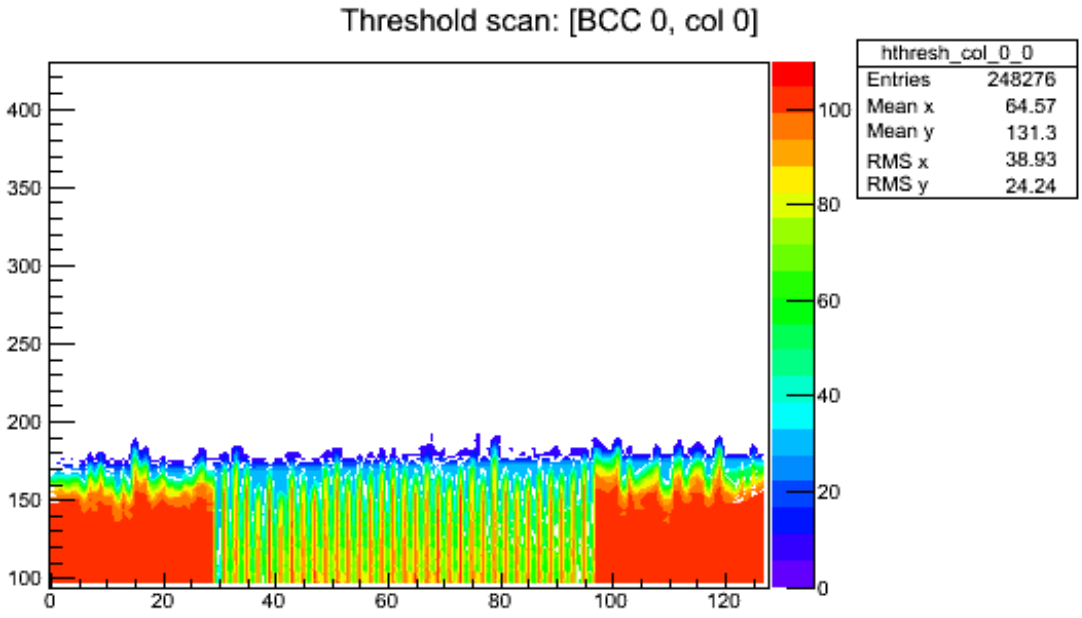

図 4.8: Threshold Scan Test (SCTJDAQ)

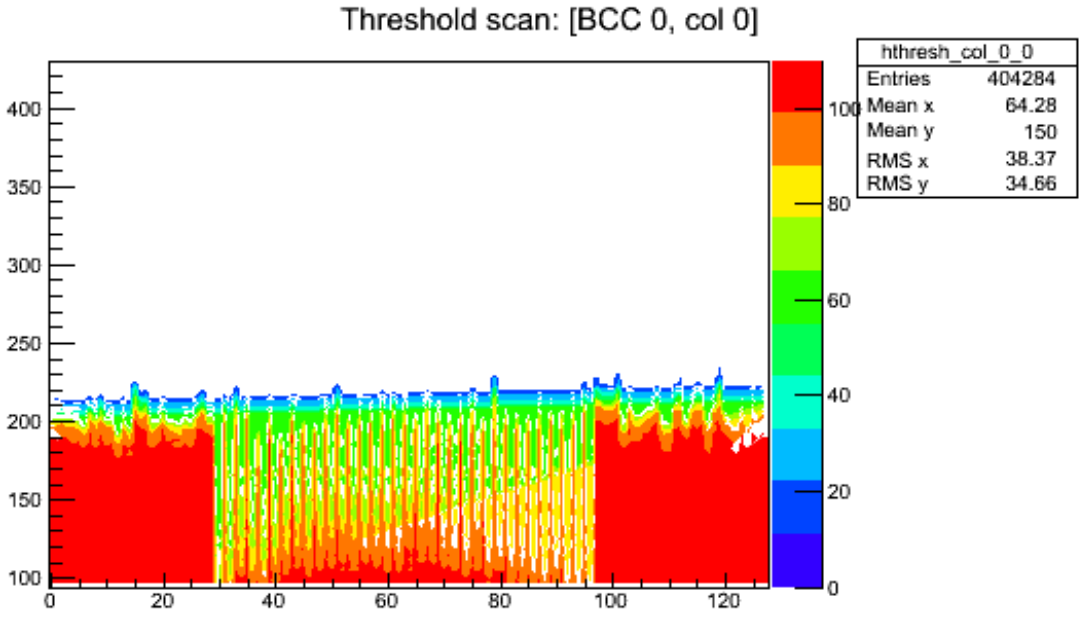

図 4.9: Threshold Scan Test (Base DAQ)

4.1.2 試験の結果について

SCTJDAQ、Base DAQ 双方とも同一ハードウェア、同一設定によってモジュール A のテストを行なっており、各テスト項目についてほぼ等しい結果が得られていることから、SCTJDAQ は正常に ABCn250 1Chip モジュールからのデータを読み出せていることが確認できた。

4.2　複数のモジュールからの読出しについての試験

この試験においては、モジュール A、B の 2 台の ABCn250 1Chip モジュールを用いて同時読出しを行なった。

4.2.1　試験の結果

以下、図 4.10〜図 4.15 に各試験の結果を示す。

L1 Delay Test

本試験における L1 Delay の Scan Range は 100 (2.500μs)〜130 (3.250μs) である。

また、ヒストグラムの縦軸は L1 Delay、横軸はストリップのチャネルである。

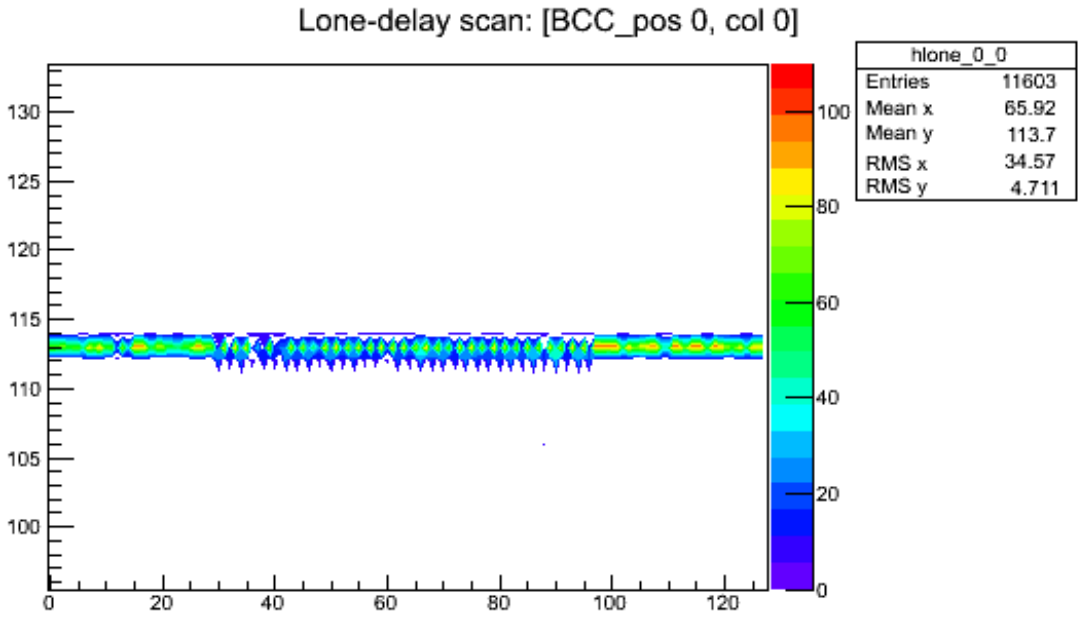

図 4.10: L1 Delay Test (モジュール A)

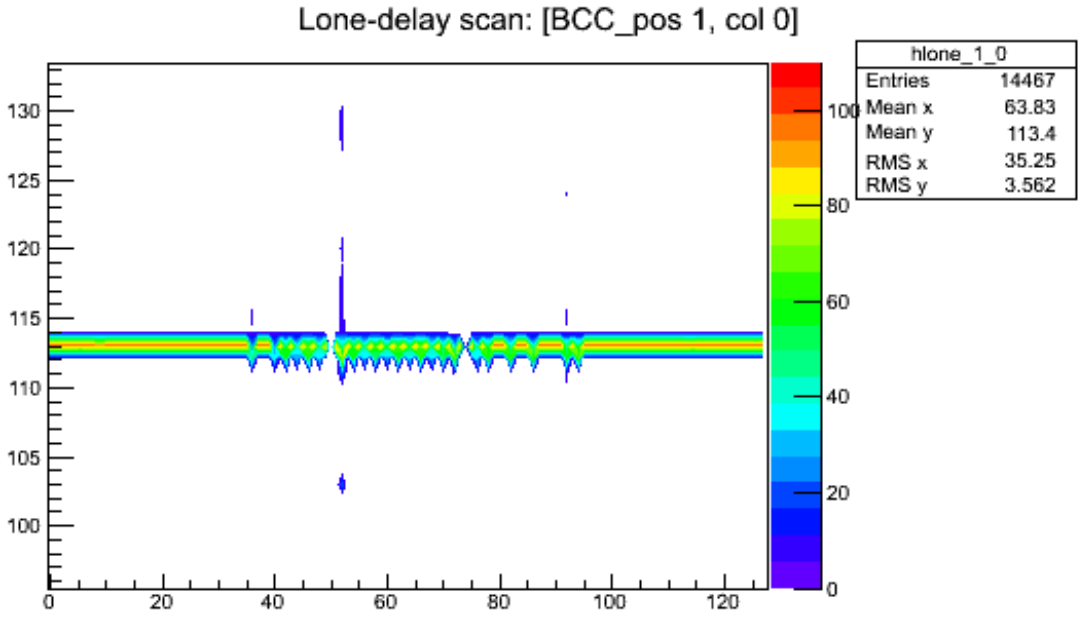

図 4.11: L1 Delay Test (モジュール B)

Strobe Delay Test

本試験における Strobe Delay の Scan Range は 0 (0 ns)〜63 (63 ns) である。また、ヒストグラムの縦軸は Strobe Delay、横軸はストリップのチャネルである。

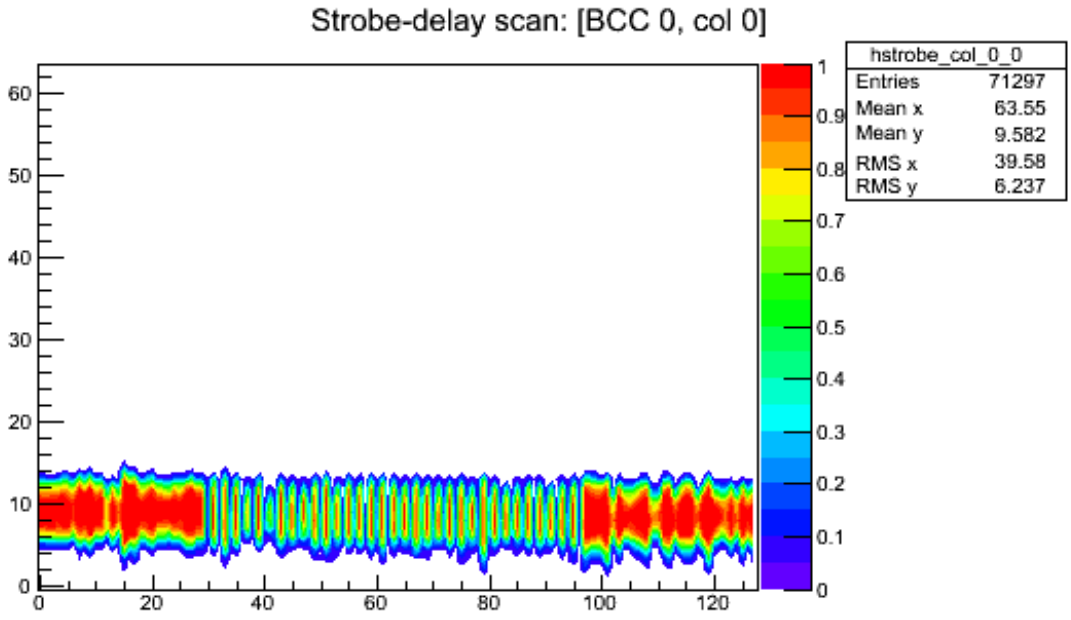

図 4.12: Strobe Delay Test (モジュール A)

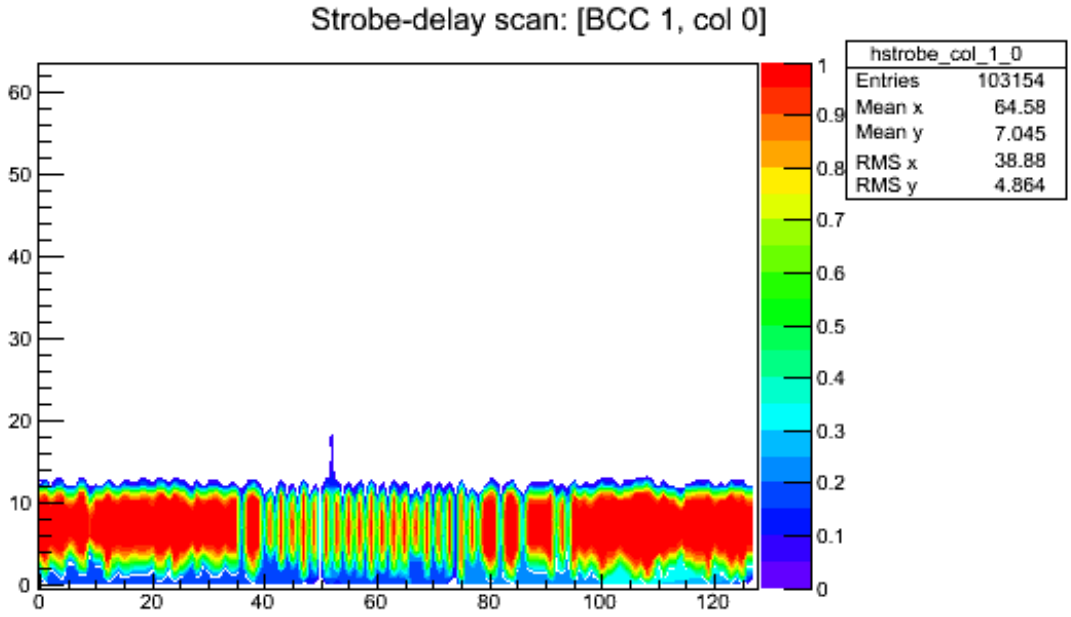

図 4.13: Strobe Delay Test (モジュール B)

Threshold Scan Test

本試験における Threshold の Scan Range は 30 (-96 mV)〜120 (-384 mV) である。また、ヒストグラムの縦軸は Threshold、横軸はストリップのチャネルである。

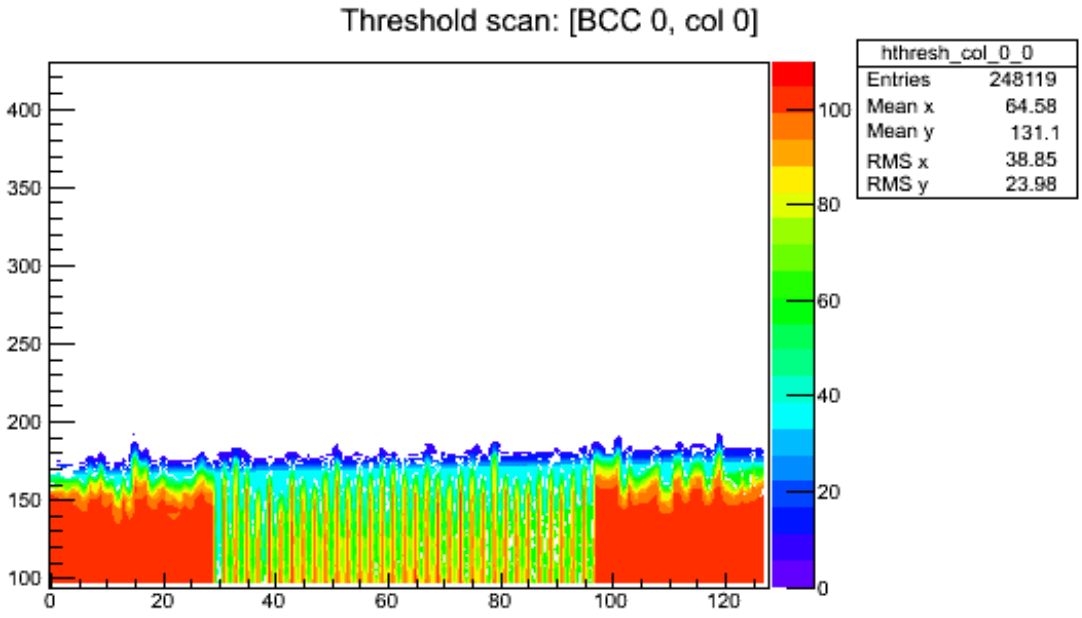

図 4.14: Threshold Scan Test (モジュール A)

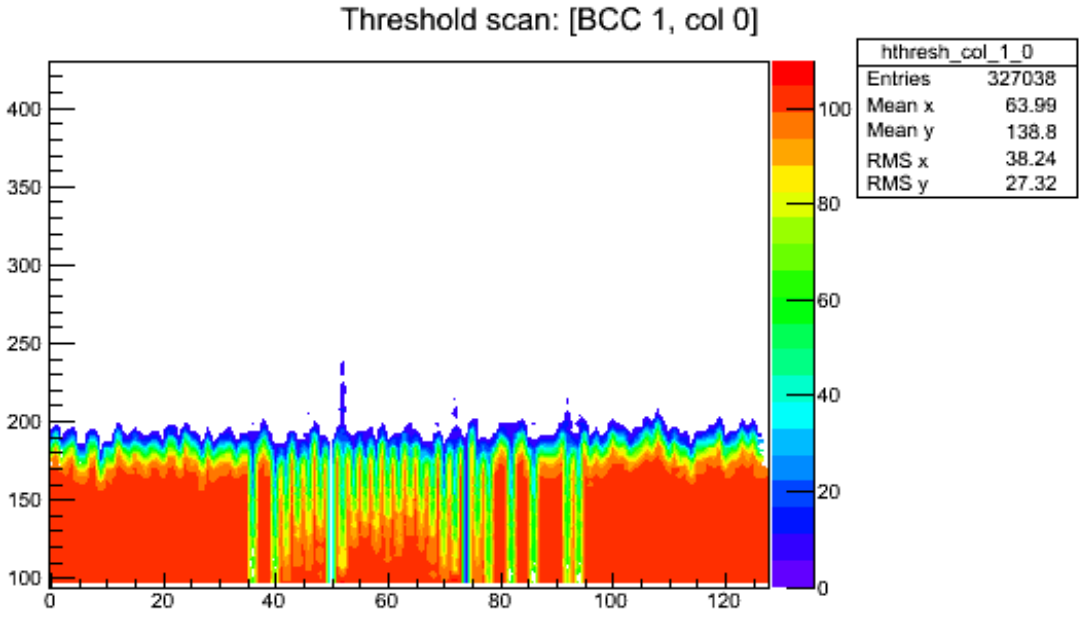

図 4.15: Threshold Scan Test (モジュール B)

4.2.2 試験の結果について

モジュール A、B は基本的には同一仕様であるものの、個体差があることからわずかに結果が異なる部分が存在する。しかし、モジュール A については 1 台のみで読み出した結果ともよく一致し、またモジュール B についても設計上の仕様から想定される範囲の結果であることから、SCTJDAQ は正常に 2 台の ABCn250 1Chip モジュールからのデータを読み出せていると判断できる。

4.3　疑似パルスによる信号の読出し試験

この試験においては、モジュール A、B の 2 台の ABCn250 1Chip モジュールを用いて同時読出しを行なった。

4.3.1　試験の結果

以下、図 4.16〜図 4.27 に各試験の結果を示す。本試験における L1 Delay の Scan Range は 0 (0.000μs)〜255 (6.375μs) である。遅延時間はトリガー用 NIM 信号の立下り時刻を 0μs として、疑似パルスの信号が遅くなる時を正、早くなる時を負で表す。また、ヒストグラムの縦軸は L1 Delay、横軸はストリップのチャネルである。

遅延時間 -1μs

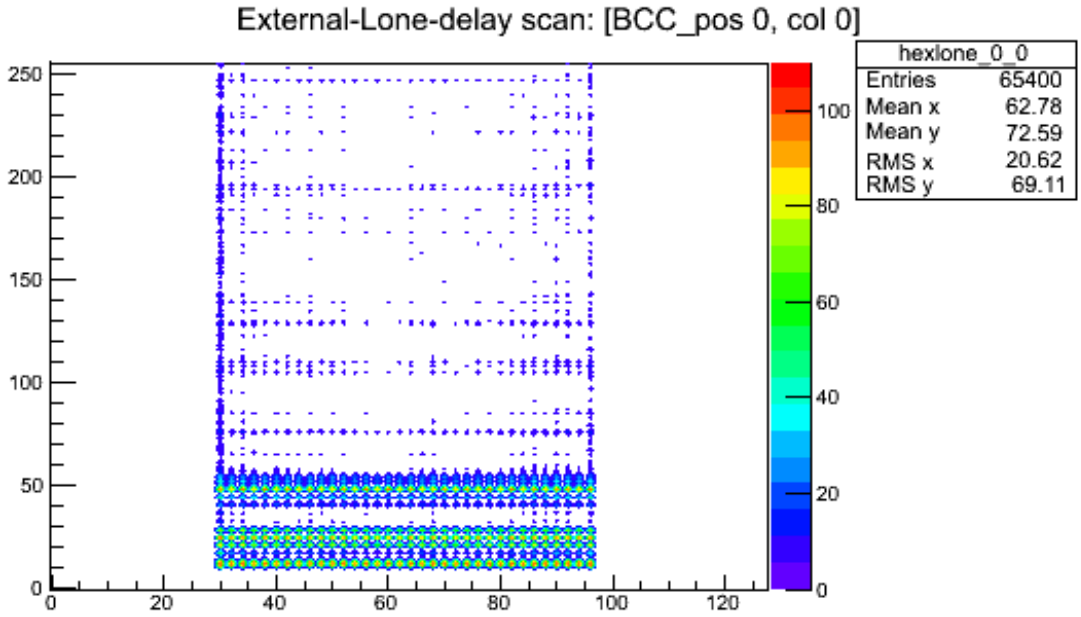

図 4.16: 疑似パルスの遅延時間 -1μs の読出し結果 (モジュール A)

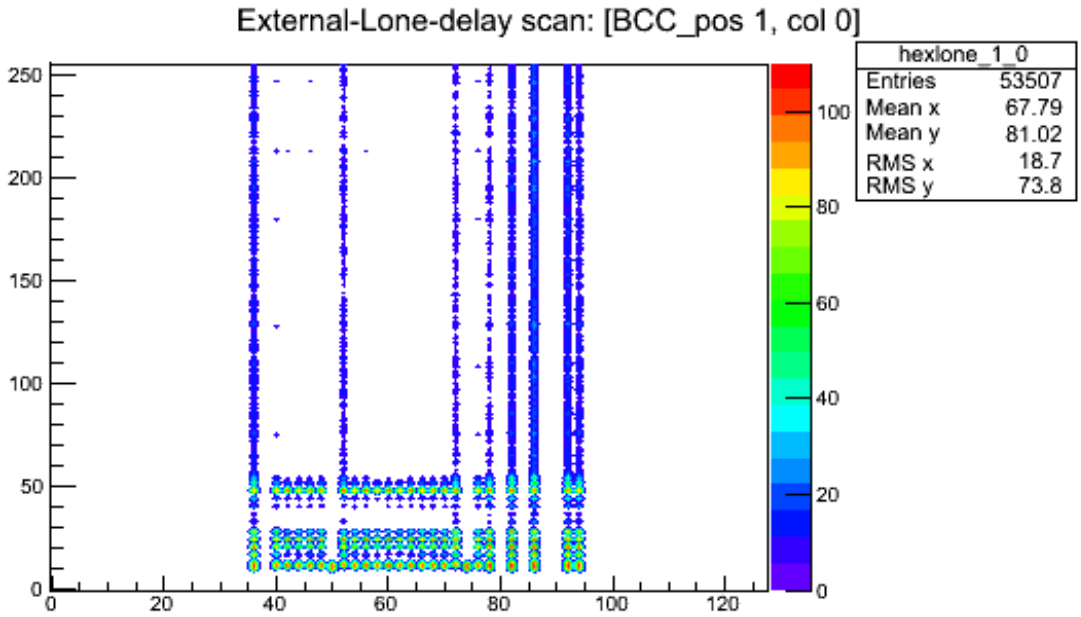

図 4.17: 疑似パルスの遅延時間 -1μs の読出し結果 (モジュール B)

遅延時間 $0\mu s$

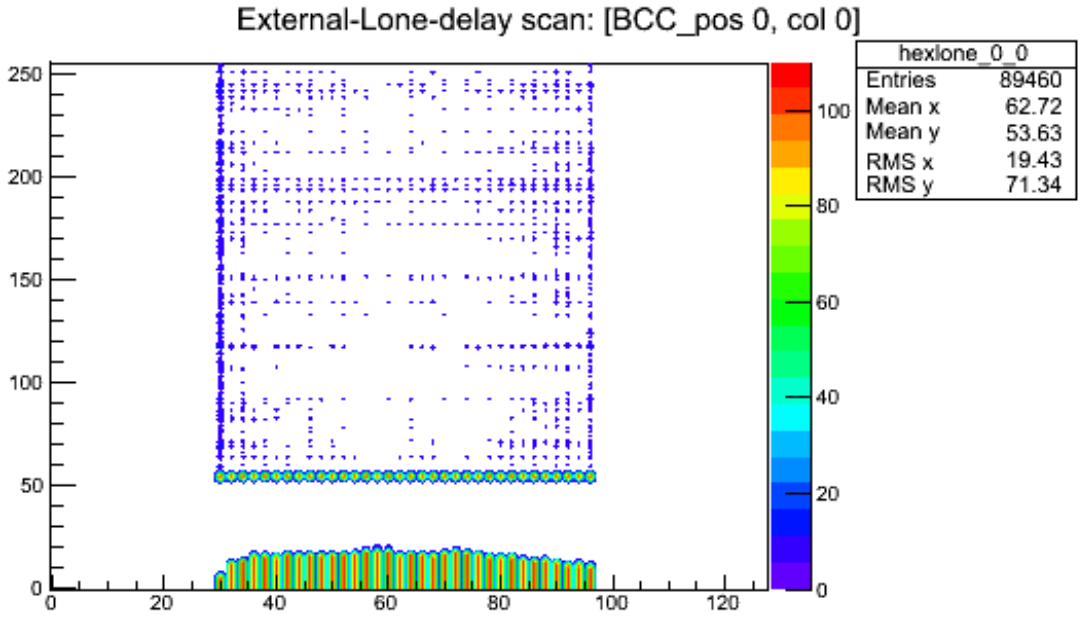

図 4.18: 疑似パルスの遅延時間 $0\mu s$ の読出し結果 (モジュール A)

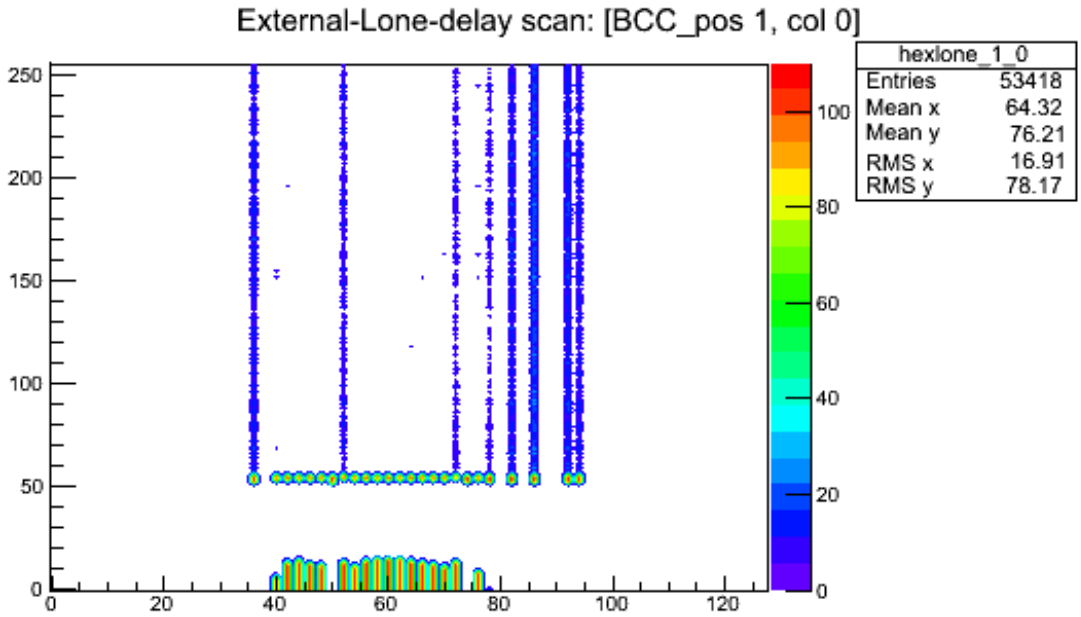

図 4.19: 疑似パルスの遅延時間 $0\mu s$ の読出し結果 (モジュール B)

遅延時間 $+1\mu$s

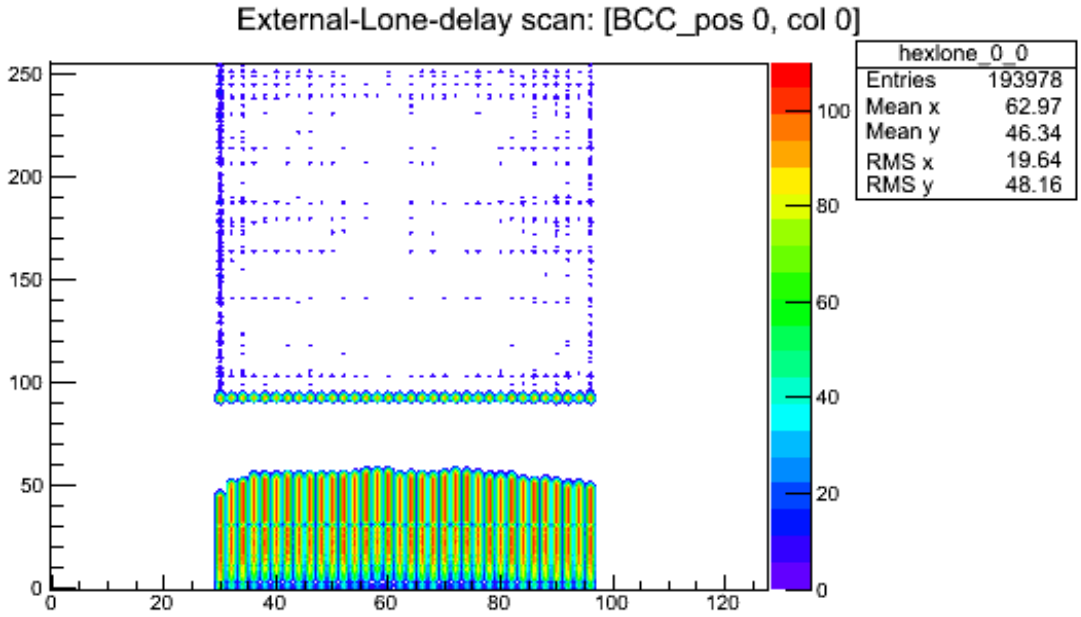

図 4.20: 疑似パルスの遅延時間 $+1\mu$s の読出し結果 (モジュール A)

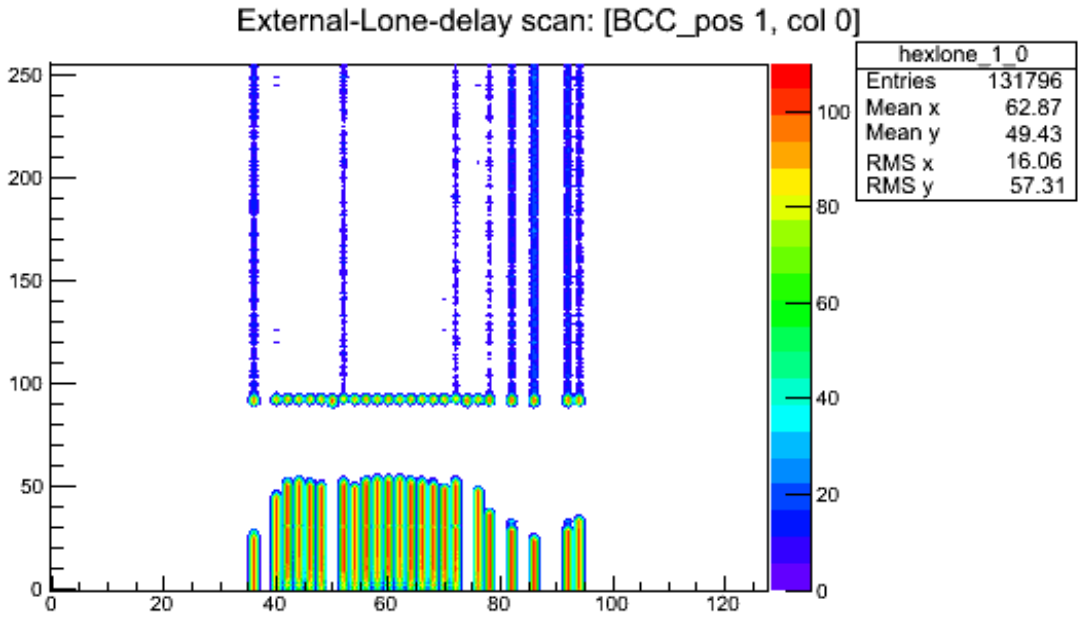

図 4.21: 疑似パルスの遅延時間 $+1\mu$s の読出し結果 (モジュール B)

遅延時間 $+2\mu$s

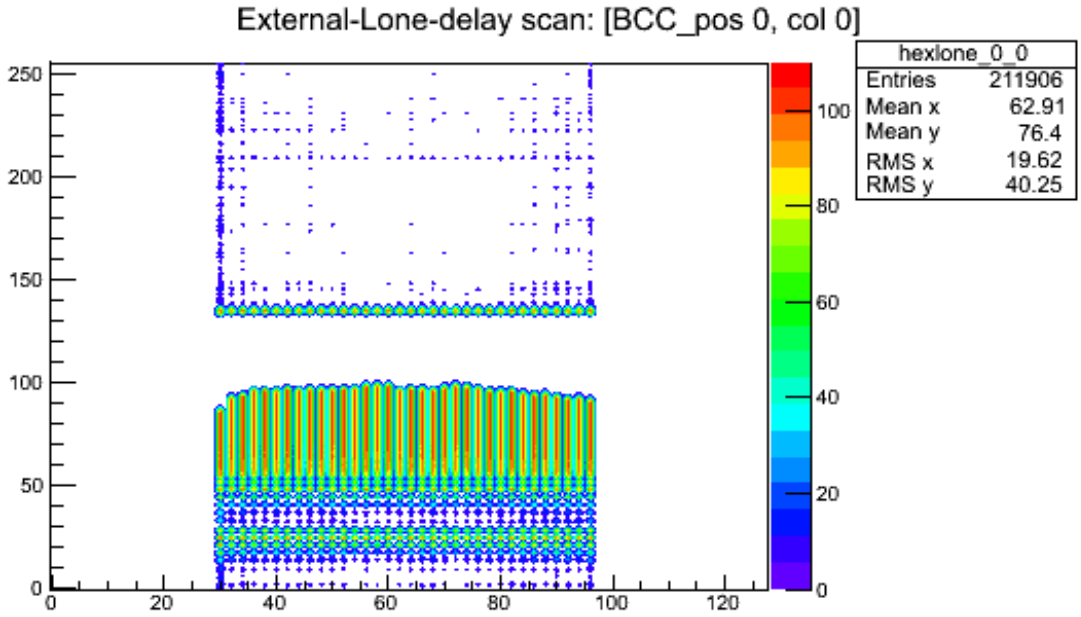

図 4.22: 疑似パルスの遅延時間 $+2\mu$s の読出し結果 (モジュール A)

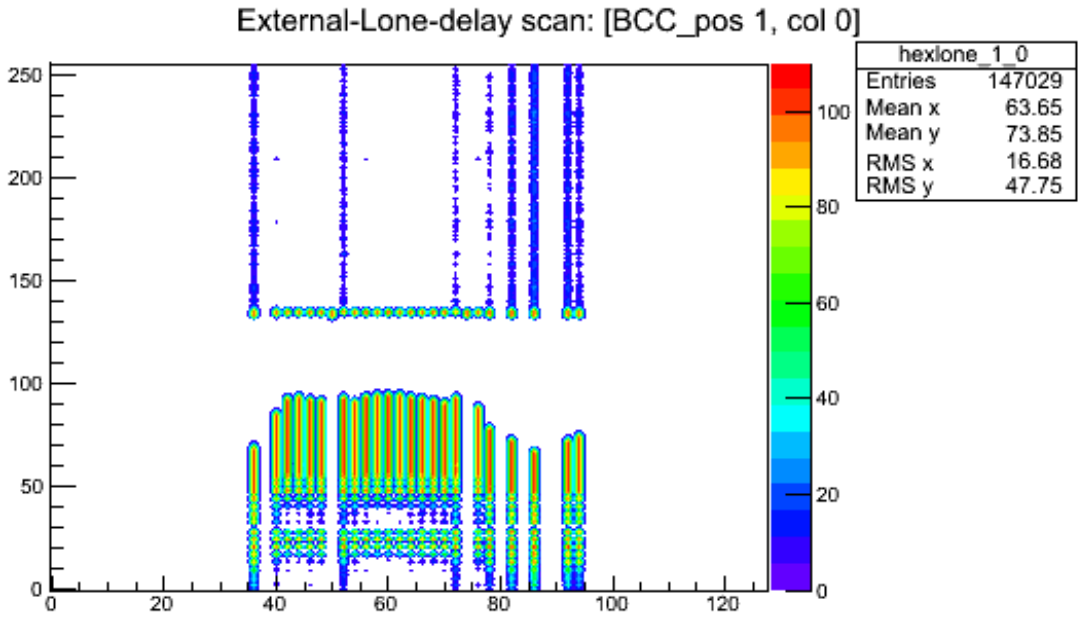

図 4.23: 疑似パルスの遅延時間 $+2\mu$s の読出し結果 (モジュール B)

遅延時間 $+3\mu$s

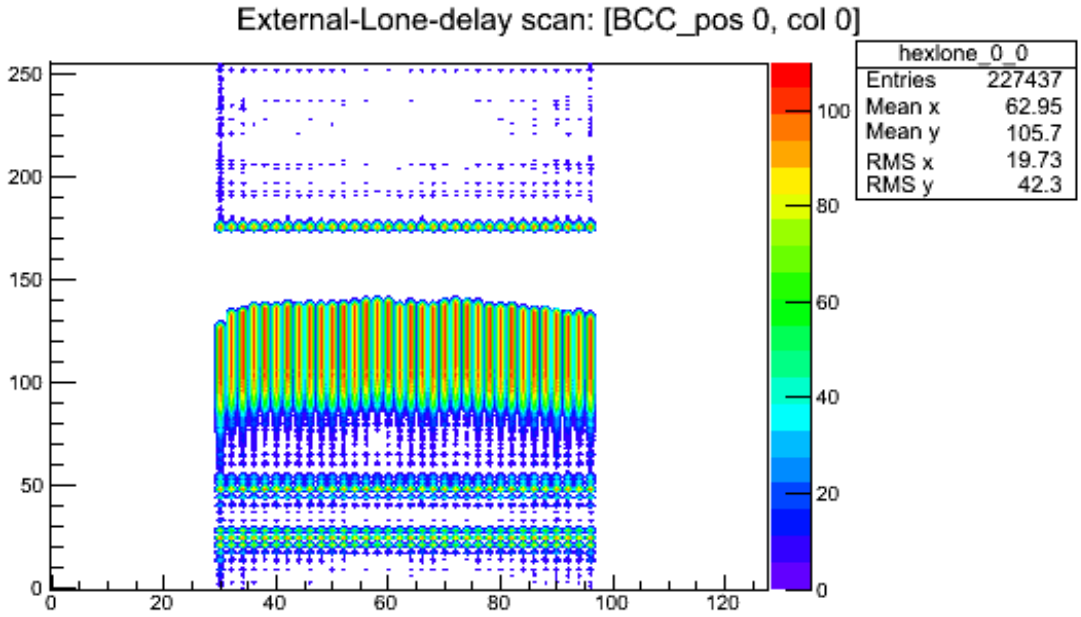

図 4.24: 疑似パルスの遅延時間 $+3\mu$s の読出し結果 (モジュール A)

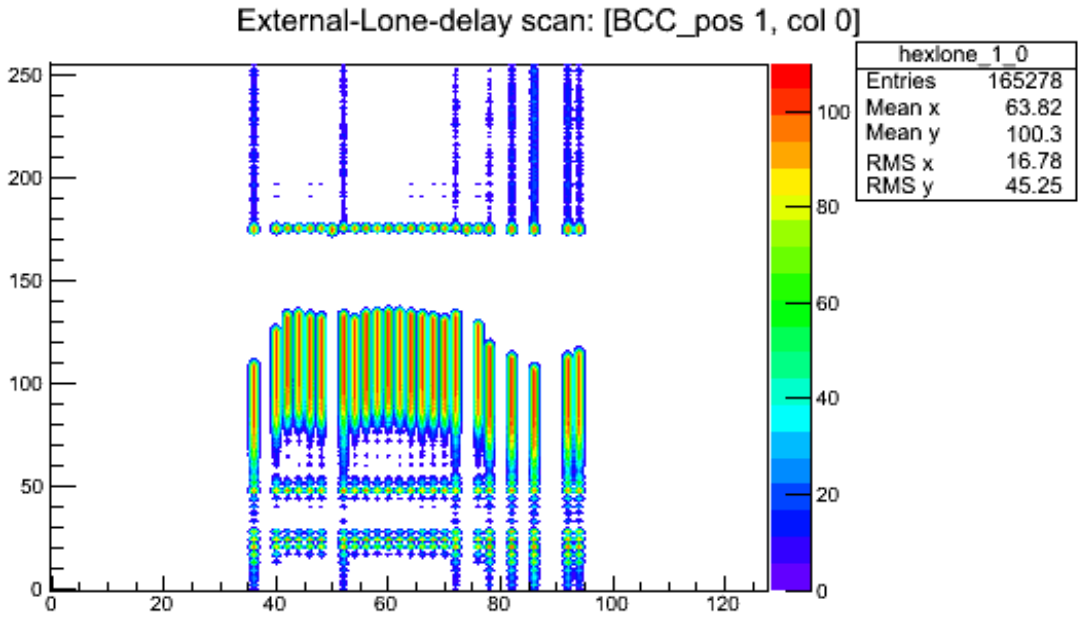

図 4.25: 疑似パルスの遅延時間 $+3\mu$s の読出し結果 (モジュール B)

遅延時間 $+4\mu$s

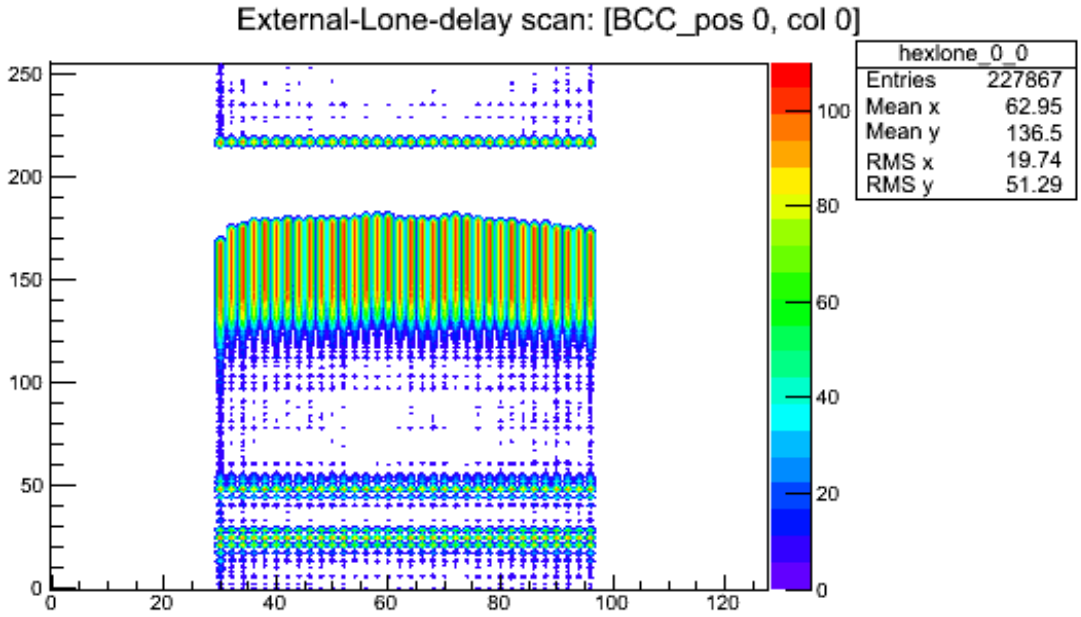

図 4.26: 疑似パルスの遅延時間 $+4\mu$s の読出し結果 (モジュール A)

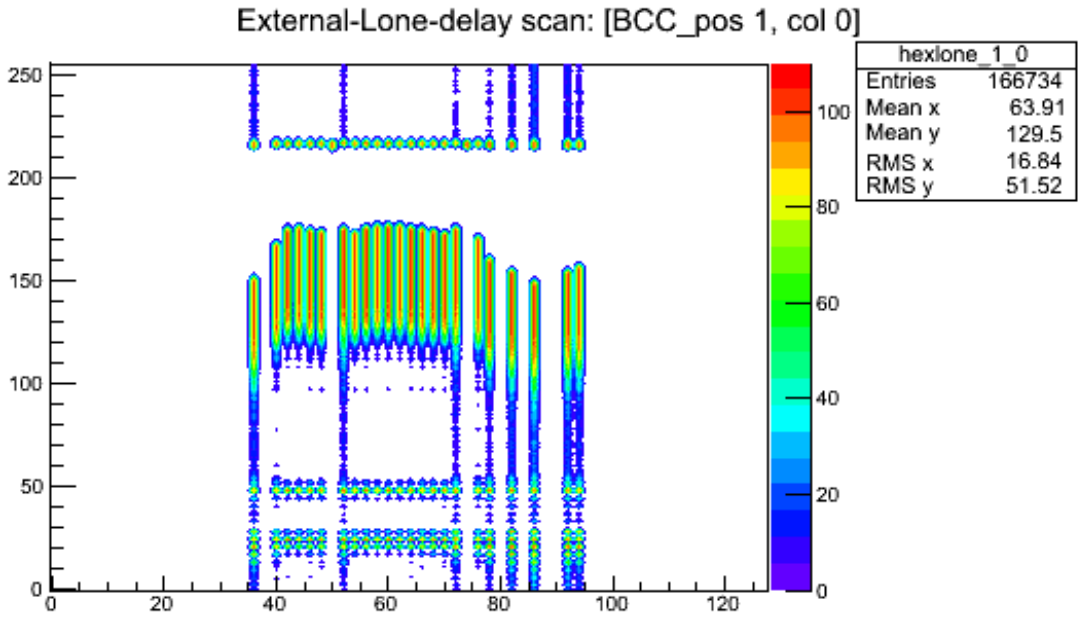

図 4.27: 疑似パルスの遅延時間 $+4\mu$s の読出し結果 (モジュール B)

4.3.2 試験の結果について

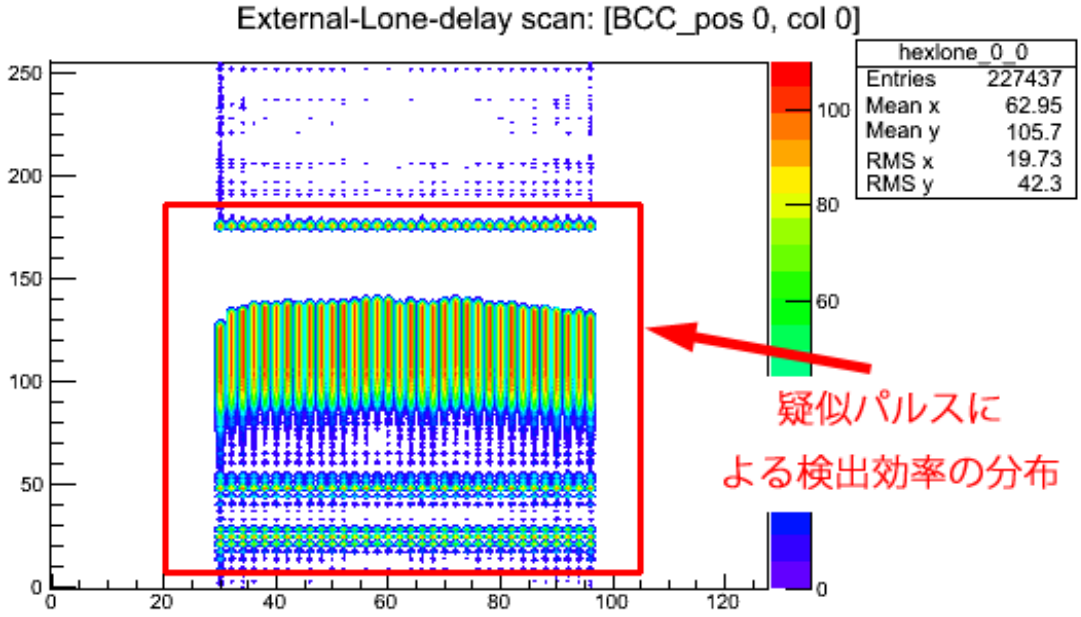

図 4.28: ヒストグラム上に観察できる疑似パルスによるの検出効率の分布。赤色実線で囲んだ部分が疑似パルスによる分布。(図 4.26 より)

入力した疑似パルスは、ヒストグラム上においては図 4.28 に示すような検出効率の分布として観察できる。$1[\mu s]/25[ns] = 40[L1Delay]$ より、疑似パルスの遅延時間を $1\mu s$ 変化させる毎に検出効率の分布は L1 Delay 40 カウント分ずれることが期待される。実際の試験結果について、すべての結果で確認することができる検出効率分布の上端のおよその位置は表 4.2 のような結果となった。

遅延時間	検出効率の分布の位置 (L1 Delay)
$-1\mu s$ (図 4.16、4.17)	10
$0\mu s$ (図 4.18、4.19)	50
$+1\mu s$ (図 4.20、4.21)	90
$+2\mu s$ (図 4.22、4.23)	130
$+3\mu s$ (図 4.24、4.25)	170
$+4\mu s$ (図 4.26、4.27)	210

表 4.2: 疑似パルスによる試験結果における検出効率の分布の位置 (上端)

表 4.2 より、遅延時間 $1\mu s$ につき L1 Delay が約 40 カウント変化していることがわかる。よって、疑似パルスの遅延時間の変化が ABCn250 の読出し結果に反映されているこ

とが確認できたことから、疑似パルスによる信号の読出しは正しく行うことができていると考えられる。

第5章　まとめと今後の課題

5.1　まとめ

新型のストリップ型シリコン検出器を評価するためのビームテスト用 DAQ システムを構築した。ABCn250 の内部テストを従来型の DAQ システムと同様に正常に実行できること、疑似パルスによるデータ取得が正常に行えていること、追加した Time Stamp について、データビットストリームへの情報付加が正常に実行できていることから、構築したシステムは正しく動作していると考える。

5.2　今後の課題

本研究においては、構築したビームテスト用 DAQ システムについて、ABCn250 からの読出し試験を行い、期待される動作が確認できた。今後は、大阪大学で開発が進められているテレスコープを組み込み、複数の測定機器が同時に動作している環境下においても正しくシステムが動作することを試験する必要がある。また、最終的な目標として、ビームテストによる動作試験を実施し、本 DAQ システムを実際にビームテストに用いることができるものとして完成させることが課題となる。

付 録A　SCTJDAQ ABCn250用モジュール マニュアル

A.1　概要

本研究において開発したSCTJDAQのうち、ABCn250チップの制御及びデータ記録・解析にかかわるモジュールについて、それぞれの機能と起動時の引数について説明する。また、ABCn250コンフィギュレーション用JSONファイル、SCTJDAQ構成用JSONファイルについて、必要な項目について説明し、実際に使用するJSONファイルを例として添付する。

A.2　実行できる試験項目

現時点のSCTJDAQにおいて実行可能な試験項目について表A.1に示す。試験実行時にはABCNReader、ABCNLogger、ABCNAnalyzerの起動時引数‘-t’に実行したい試験の番号を指定する。なお、一部番号は今後の試験用にリザーブされているため、実装されている機能の項目は連番にはなっていない。

試験番号	実行する試験の内容
0	Beam Test
1	L1 Delay Scan
2	Strobe Delay Scan
3	Threshold Scan
4	Three Point Gain Scan
5	Trim Scan
7	External Trigger L1 Delay Scan
11	Burst No Charge Scan
12	Burst Charge Scan

表 A.1: 試験項目一覧

A.3　各モジュールの機能について

A.3.1　ABCNReader

A.3.1.1　機能概要

ABCNReader は以下の機能を担うモジュールである。

- ABCn250 を SEABAS を介して制御する
- ABCn250 から読み出したデータビットストリームをデコードする
- デコードしたデータを 1 トリガーイベントずつまとめてイベントフラグメントに構成する
- イベントフラグメントを Message Queue を介して他のモジュールに送信する
- ABCn250 チップにパラメータ正しく書き込まれているかを検証する
- 各種試験に合わせたパラメータ変更、データ取得を行う

本モジュールは ABCn250 に書き込むパラメータを Memcached による共有メモリから取得する。また、実行中の試験項目や試験終了フラグなど、自己の動作状況を共有メモリ上に記録する。

A.3.1.2　起動引数

表 A.2 に起動時の引数と設定できるパラメータを示す。なお、指定されないパラメータについては既定値を使用する。

試験時の初期パラメータ、試験時の終了パラメータについては、実行する試験内容に応じて両方を指定する必要がある。一方のみの指定の場合、または初期パラメータが終了パラメータより大きい値をとる場合については指定を無効とし、既定値を使用する。表 A.3 に各試験内容について、試験時の初期パラメータ、試験時の終了パラメータを示す。

起動時引数 '-bccoff' は、データビットストリームのデコードについて、BCC チップが存在しない環境において MQ に送信するデータが 2 重になることを防ぐためのオプションである。ABCn250 読出し用ファームウェアから送信されるデータビットストリームは

引数	設定できるパラメータ	既定値
-t	試験項目	0
-pS	試験時の初期パラメータ	表 A.3
-pE	試験時の終了パラメータ	表 A.3
-portc	データ送信先の MQ ポート数	1
-bccoff	BCC 対応のデコードを OFF にする (1 で OFF)	0
-run	Beam Test 時の試験終了トリガー回数指定 (ABCn250 用 Ntrigger の回数 × 本引数指定値)	1
-ip	通信する SEABAS の IP アドレス	192.168.0.16

表 A.2: ABCNReader 起動時引数一覧

40Mhz の信号を 80Mhz で読み出しているために同じデータが 2 重になっている。よって、BCC が存在しない環境においてデコードしたデータの一方を破棄する必要がある。SCTJDAQ の ABCn250 チップ読出しモジュールは BCC の有り/無し両方に対応できるようにするため、起動時引数によってデータの破棄を行うかどうかを指定する。‘1’ でデータを破棄し、‘0’ または指定なしでデータの破棄を無効にする。

実行する試験の内容	初期値	終了値	備考
Beam Test	-	-	値指定なし
L1 Delay Scan	100	130	
Strobe Delay Scan	0	63	
Threshold Scan	30	150	入力電荷の試験値は 固定値 (40)
Three Point Gain Scan (Threshold 値)	30	150	入力電荷の試験値は 固定値 (40 , 50 , 60)
Trim Scan (Threshold 値)	30	150	入力電荷の試験値は Range : 1 , Data : 0 〜 Range : 2 , Data : 1
External Trigger L1 Delay Scan	0	255	
Burst No Charge Scan	-	-	値指定なし
Burst Charge Scan	-	-	値指定なし

表 A.3: 各試験内容についての ScanRange の初期設定値

A.3.2 ABCNLogger(Logger)

A.3.2.1 機能概要

ABCNLogger は、Message Queue を介して受信したイベントフラグメントを ROOT Tree に詰め、Ntuple として ROOT ファイルに記録する機能を担うモジュールである。

A.3.2.2 起動引数

表 A.4 に起動時の引数と設定できるパラメータを示す。

引数	設定できるパラメータ	既定値
-t	試験項目	0
-d	記録用ファイル名	(絶対パスにて指定必須)

表 A.4: ABCNLogger 起動時引数一覧

A.3.3 ABCNAnalyzer(Analyzer)

A.3.3.1 機能概要

ABCNAnalyzer は、Message Queue を介して受信したイベントフラグメントを指定された試験項目に応じて処理し、結果をヒストグラムやパラメータの修正値として出力する

機能を担うモジュールである。本モジュールは、ヒストグラムを ROOT ファイルとして記録するが、この時 GUI との間でファイルの排他制御を行うため実行前にあらかじめセマフォ (POSIX Semaphore)‘/ABCNAnalyzer’ が作成されている必要がある。SCTJDAQ Software にはセマフォ作成 (createLock)、削除 (unlinkLock) のプログラムが用意されている。

A.3.3.2 起動引数

表 A.5 に起動時の引数と設定できるパラメータを示す。なお、Hybrid のタイプ判定 [1] は通常はコンフィギュレーションパラメータから決定されるが、1Chip モジュールにおいては column が 1 つであるために判別ができず、エラーとなるため、1Chip モジュール使用時については起動時の引数によって指定する必要がある。また、1Chip モジュール用 column 設定はヒストグラム作成時に不要な column1 を作成しないようにするためのものである。‘-OneChip’ については引数に値をつけずに使用する。

引数	設定できるパラメータ	既定値
-t	試験項目	0
-d	記録用ファイル名	-(絶対パスにて指定必須)
-forceHyb	Hybrid のタイプ判定	-
-OneChip	1Chip モジュール用 column 設定	-

表 A.5: ABCNAnalyzer 起動時引数一覧

A.3.4 Dispatcher

A.3.4.1 機能概要

Dispatcher は、Message Queue を介して ABCNReader から受信したイベントフラグメントを ABCNLogger、ABCNAnalyzer に Message Queue を介して分配するモジュールである。本モジュールについては、特別な起動時の引数は存在しない。

[1] 製造者によって Hybird のチップアドレスに違いがあるため、判別を行う

A.4 ABCn250コンフィギュレーション用JSONファイル

このファイルは、ABCn250 に設定を行うためのすべてのコンフィギュレーションパラメータを記述したものである。SCTJDAQ 使用時には GUI から JSON ファイルを指定して用いる。

ソースコード A.1 に ABCn250 1Chip モジュールを 2 台を接続する場合の JSON ファイルの例を示す。設定値は ABCn250 の Specification[4] に基づく。すべてのモジュールに共通する設定項目については 'Common-' で始まるキー、ハイブリッドモジュールごとに個別の値については 'Hybrid-(Hybridnumber)-' で始まるキーに設定する。'Hybrid-(Hybridnumber)-BCC-' で始まるキーは BCC チップに関連するものであり、BCC のない環境においては原則不要な設定となるが、'Hybrid-(Hybridnumber)-BCC-pos' は接続されている LINK の位置、'Hybrid-(Hybridnumber)-BCC-address' は接続されている LINK の FIFO に割り当てられた固有値を示すので、この 2 項目については BCC の有無に依らず設定が必要となる。

ソースコード A.1: ABCn250 1Chip モジュール 2 台を接続する場合のコンフィギュレーション用 JSON ファイル

```
{ "SCTJDAQ":
  { "ABCNReader":
    { "Common-ClockMode": ["int",1],
      "Common-Ntrigger": ["int",100],
      "Common-DTrigLatency": ["int",124],
      "Common-L1Delay": ["int",113],
      "Common-L1Mode0": ["int",1],
      "Common-Edge": ["int",0],
      "Common-Vthn": ["int",50],
      "Common-Vthp": ["int",0],
      "Common-CalAmp": ["int",50],
      "Common-CalPol": ["int",0],
      "Common-SigPol": ["int",0],
      "Common-PIPE_STOP": ["int",0],
      "Common-CD_STOP": ["int",0],
      "Common-REG_STOP": ["int",0],
      "Common-BPreamp": ["int",16],
      "Common-BPreampBuff": ["int",16],
      "Common-BPreampFbck": ["int",1],
      "Common-BShap": ["int",16],
      "Common-BShapFbck": ["int",16],
      "Common-BDiff": ["int",16],
      "Common-BComp": ["int",16],
      "Hybrid-1-id": ["string","Hybrid_0"],
      "Hybrid-1-num": ["int",0],
      "Hybrid-1-BCC-pos": ["int",0],
```

```
27      "Hybrid-1-BCC-active": ["int",1],
28      "Hybrid-1-BCC-address": ["int",0],
29      "Hybrid-1-BCC-R80": ["int",0],
30      "Hybrid-1-BCC-Dmux": ["int",1],
31      "Hybrid-1-BCC-Qmode": ["int",0],
32      "Hybrid-1-BCC-SclkInv": ["int",0],
33      "Hybrid-1-BCC-AclkInv": ["int",1],
34      "Hybrid-1-BCC-BcoInv": ["int",0],
35      "Hybrid-1-BCC-DclkInv": ["int",0],
36      "Hybrid-1-Chip-1-id": ["string","M0"],
37      "Hybrid-1-Chip-1-col": ["int",0],
38      "Hybrid-1-Chip-1-address": ["int",0],
39      "Hybrid-1-Chip-1-Role": ["int",3],
40      "Hybrid-1-Chip-1-Flow": ["int",0],
41      "Hybrid-1-Chip-1-DriveUp": ["int",10],
42      "Hybrid-1-Chip-1-DriveBot": ["int",10],
43      "Hybrid-1-Chip-1-CalDelay": ["int",6],
44      "Hybrid-1-Chip-1-CalStep": ["int",2],
45      "Hybrid-1-Chip-1-Mask": ["string","00000000␣00000000␣00000000␣
            00000000"],
46      "Hybrid-1-Chip-1-TrimMode": ["int",1],
47      "Hybrid-1-Chip-1-TrimRange": ["int",4],
48      "Hybrid-1-Chip-1-TrimTarget": ["int",0],
49      "Hybrid-1-Chip-1-Trim-1-start": ["int",0],
50      "Hybrid-1-Chip-1-Trim-1-end": ["int",15],
51      "Hybrid-1-Chip-1-Trim-1-value": ["string","0E111816␣15191717␣181
            A1112␣1A131F18"],
52      "Hybrid-1-Chip-1-Trim-2-start": ["int",16],
53      "Hybrid-1-Chip-1-Trim-2-end": ["int",31],
54      "Hybrid-1-Chip-1-Trim-2-value": ["string","1B151719␣18111C17␣
            14101D13␣1E101810"],
55      "Hybrid-1-Chip-1-Trim-3-start": ["int",32],
56      "Hybrid-1-Chip-1-Trim-3-end": ["int",47],
57      "Hybrid-1-Chip-1-Trim-3-value": ["string","15101913␣0F101910␣
            16110F0F␣16101215"],
58      "Hybrid-1-Chip-1-Trim-4-start": ["int",48],
59      "Hybrid-1-Chip-1-Trim-4-end": ["int",63],
60      "Hybrid-1-Chip-1-Trim-4-value": ["string","17161613␣1C171211␣
            1715170F␣1A151B16"],
61      "Hybrid-1-Chip-1-Trim-5-start": ["int",64],
62      "Hybrid-1-Chip-1-Trim-5-end": ["int",79],
63      "Hybrid-1-Chip-1-Trim-5-value": ["string","15121211␣16171812␣
            1417180E␣15191314"],
64      "Hybrid-1-Chip-1-Trim-6-start": ["int",80],
65      "Hybrid-1-Chip-1-Trim-6-end": ["int",95],
66      "Hybrid-1-Chip-1-Trim-6-value": ["string","1914190C␣13151414␣170
            B130D␣13121416"],
67      "Hybrid-1-Chip-1-Trim-7-start": ["int",96],
68      "Hybrid-1-Chip-1-Trim-7-end": ["int",111],
69      "Hybrid-1-Chip-1-Trim-7-value": ["string","14141713␣1614180E␣
            10131A0E␣13121113"],
70      "Hybrid-1-Chip-1-Trim-8-start": ["int",112],
71      "Hybrid-1-Chip-1-Trim-8-end": ["int",127],
72      "Hybrid-1-Chip-1-Trim-8-value": ["string","0E1A1812␣0E0D190F␣130
```

```
      D0E16␣131C1611"],
    "Hybrid-1-Chip-1-rcfit-p0": ["double",0.0],
    "Hybrid-1-Chip-1-rcfit-p1": ["double",0.0],
    "Hybrid-1-Chip-1-rcfit-p2": ["double",0.0],
    "Hybrid-2-id": ["string","Hybrid_1"],
    "Hybrid-2-num": ["int",1],
    "Hybrid-2-BCC-pos": ["int",1],
    "Hybrid-2-BCC-active": ["int",1],
    "Hybrid-2-BCC-address": ["int",1],
    "Hybrid-2-BCC-R80": ["int",0],
    "Hybrid-2-BCC-Dmux": ["int",1],
    "Hybrid-2-BCC-Qmode": ["int",0],
    "Hybrid-2-BCC-SclkInv": ["int",0],
    "Hybrid-2-BCC-AclkInv": ["int",1],
    "Hybrid-2-BCC-BcoInv": ["int",0],
    "Hybrid-2-BCC-DclkInv": ["int",0],
    "Hybrid-2-Chip-1-id": ["string","M0"],
    "Hybrid-2-Chip-1-col": ["int",0],
    "Hybrid-2-Chip-1-address": ["int",0],
    "Hybrid-2-Chip-1-Role": ["int",3],
    "Hybrid-2-Chip-1-Flow": ["int",0],
    "Hybrid-2-Chip-1-DriveUp": ["int",10],
    "Hybrid-2-Chip-1-DriveBot": ["int",10],
    "Hybrid-2-Chip-1-CalDelay": ["int",6],
    "Hybrid-2-Chip-1-CalStep": ["int",2],
    "Hybrid-2-Chip-1-Mask": ["string","00000000␣00000000␣00000000␣
      00000000"],
    "Hybrid-2-Chip-1-TrimMode": ["int",1],
    "Hybrid-2-Chip-1-TrimRange": ["int",4],
    "Hybrid-2-Chip-1-TrimTarget": ["int",0],
    "Hybrid-2-Chip-1-Trim-1-start": ["int",0],
    "Hybrid-2-Chip-1-Trim-1-end": ["int",15],
    "Hybrid-2-Chip-1-Trim-1-value": ["string","0E111816␣15191717␣181
      A1112␣1A131F18"],
    "Hybrid-2-Chip-1-Trim-2-start": ["int",16],
    "Hybrid-2-Chip-1-Trim-2-end": ["int",31],
    "Hybrid-2-Chip-1-Trim-2-value": ["string","1B151719␣18111C17␣
      14101D13␣1E101810"],
    "Hybrid-2-Chip-1-Trim-3-start": ["int",32],
    "Hybrid-2-Chip-1-Trim-3-end": ["int",47],
    "Hybrid-2-Chip-1-Trim-3-value": ["string","15101913␣0F101910␣
      16110F0F␣16101215"],
    "Hybrid-2-Chip-1-Trim-4-start": ["int",48],
    "Hybrid-2-Chip-1-Trim-4-end": ["int",63],
    "Hybrid-2-Chip-1-Trim-4-value": ["string","17161613␣1C171211␣
      1715170F␣1A151B16"],
    "Hybrid-2-Chip-1-Trim-5-start": ["int",64],
    "Hybrid-2-Chip-1-Trim-5-end": ["int",79],
    "Hybrid-2-Chip-1-Trim-5-value": ["string","15121211␣16171812␣
      1417180E␣15191314"],
    "Hybrid-2-Chip-1-Trim-6-start": ["int",80],
    "Hybrid-2-Chip-1-Trim-6-end": ["int",95],
    "Hybrid-2-Chip-1-Trim-6-value": ["string","1914190C␣13151414␣170
      B130D␣13121416"],
```

```
      "Hybrid-2-Chip-1-Trim-7-start": ["int",96],
      "Hybrid-2-Chip-1-Trim-7-end": ["int",111],
      "Hybrid-2-Chip-1-Trim-7-value": ["string","14141713␣1614180E␣
          10131A0E␣13121113"],
      "Hybrid-2-Chip-1-Trim-8-start": ["int",112],
      "Hybrid-2-Chip-1-Trim-8-end": ["int",127],
      "Hybrid-2-Chip-1-Trim-8-value": ["string","0E1A1812␣0E0D190F␣130
          D0E16␣131C1611"],
      "Hybrid-2-Chip-1-rcfit-p0": ["double",0.0],
      "Hybrid-2-Chip-1-rcfit-p1": ["double",0.0],
      "Hybrid-2-Chip-1-rcfit-p2": ["double",0.0],
      "Hybrid-TotalHyb": ["int",2],
      "Hybrid-TotalChips": ["int",2],
      "Hybrid-1-InChips": ["int",1],
      "Hybrid-2-InChips": ["int",1]
    }
  }
}
```

A.5 SCTJDAQ 構成用 JSON ファイル

この JSON ファイルは SCTJDAQ において使用するモジュール、及びモジュール間を接続する Message Queue (MQ) のポートを指定するものである。SCTJDAQ には実行前に MQ ポートを作成する Python コード (sctmkmq.py)、MQ ポートを削除する Python コード (sctunlink.py) が用意されている。SCTJDAQ 使用時には GUI から JSON ファイルを指定して用いる。

ソースコード A.2 に示す内容は SCTJDAQ 構成用 JSON ファイルにおいて使用するモジュール 1 つ分の設定内容である。使用したいモジュール分の設定を並べて記述し、全体を '[ ]' で囲む。

ソースコード A.2: 使用するモジュールに関する設定

```
1  {
2    "component": {
3      "name" : "ABCNReader",
4      "exec" : {
5        "path" : "/home/sctjdaq/bin/ABCNReader",
6        "arguments" : "-t␣1␣-pS␣100␣-pE␣130␣-bccoff␣1",
7        "order" : 4
8      },
9      "resource" : "/ABCNReader",
10     "command" : "/sctabcnreader",
11     "inports" : [""],
12     "outports" : ["/sctdispatcher"]
13   }
14 },
```

name

"name" には使用するモジュールの名称を記述する。

path

"path" にはモジュールの絶対パスを記述する。

arguments

"arguments" には起動時の引数を記述する。

order

"order" はフレームワークからコマンドが送信される際の順位である。データフローの下流に位置するモジュールほど高い優先順位 (小さな数字) を指定する。

resource

”resource”にはモジュールのファイル名を記述する。

command

”command”にはフレームワークからコマンドを受け取るための MQ ポートを指定する (重複不可)。

inports

”inports”には他のモジュールからデータを受け取るための MQ ポートを指定する (重複不可)。この時、データの送信元モジュールの”outports”に同じポートを設定する必要がある。

outports

”outports”には他のモジュールにデータを受け取るための MQ ポートを指定する (重複不可)。この時、データの送信先モジュールの”inports”に同じポートを設定する必要がある。

ソースコード A.3 に ABCn250 に対して L1 Delay Scan を実行する場合の構成用 JSON ファイルの例を示す。

ソースコード A.3: ABCn250 に対して L1 Delay Scan を実行する場合の構成用 JSON ファイル

```
[
  {
    "component": {
      "name" : "ABCNReader",
      "exec" : {
        "path" : "/home/sctjdaq/bin/ABCNReader",
        "arguments" : "-t␣1␣-pS␣100␣-pE␣130␣-bccoff␣1",
        "order" : 4
      },
      "resource" : "/ABCNReader",
      "command" : "/sctabcnreader",
      "inports" : [""],
      "outports" : ["/sctdispatcher"]
    }
  },
  {
    "component" : {
      "name" : "Dispatcher",
      "exec" : {
        "path" : "/home/sctjdaq/bin/Dispatcher",
        "arguments" : "",
        "order" : 3
      },
      "resource" : "/Dispatcher",
      "command" : "/sctabcndispatcher",
      "inports" : ["/sctdispatcher"],
      "outports" : ["/sctlogger","/sctanalyzer"]
    }
  },
  {
    "component" : {
      "name" : "ABCNAnalyzer",
      "exec" : {
        "path" : "/home/sctjdaq/bin/ABCNAnalyzer",
        "arguments" : "-d␣/home/sctjdaq/data/lone_hist.root␣-t␣1␣-
            forceHyb␣0␣-OneChip",
        "order" : 2
      },
      "resource" : "/ABCNAnalyzer",
      "command" : "/sctabcnanalyzer",
      "inports" : ["/sctanalyzer"],
      "outports" : [""]
    }
  },
  {
    "component" : {
      "name" : "ABCNLogger",
      "exec" : {
        "path" : "/home/sctjdaq/bin/ABCNLogger",
        "arguments" : "-d␣/home/sctjdaq/data/lone.root␣-t␣1",
        "order" : 1
      },
```

```
52      "resource" : "/ABCNLogger",
53      "command" : "/sctabcnlogger",
54      "inports" : ["/sctlogger"],
55      "outports" : [""]
56    }
57  }
58 ]
```

謝辞

本研究を行うにあたって、指導教官である高嶋隆一准教授、沖花彰教授には、多くの助言、ご指導を頂きましたことを深く御礼申し上げます。高嶋隆一准教授には研究以外の生活面においても多くのサポートをして頂き、お陰様で充実した大学院生活を送ることができました。

KEK の安芳次さんには、DAQ システムのフレームワーク構築に当たって大変お世話になりました。また、DAQ システム開発を進めるに当たって必要な多くの知識・技術を熱心にご指導頂きました。要領を得ない私の質問にもわかりやすく丁寧に答えてくださり、私が研究を進めるにあたって大きな助けになりました。深く御礼申し上げます。

KEK の田窪洋介さんには、ファームウェアの開発に当たって様々な助言を頂きました。また、CERN 滞在時には研究活動や海外での生活について、興味深いお話を聴かせて頂きました。本当にありがとうございました。

KEK の海野義信さん、池上陽一さんにはお忙しい中、本研究において様々な御助言や御助力を頂きました。本当にありがとうございました。

大阪大学の遠藤理樹さんには、ファームウェアの開発に当たって様々な助言を頂き、また先行研究を参考にさせて頂きました。CERN 滞在時には海外に慣れない私に親切にしてくださいました。本当にありがとうございました。

大阪大学の岡村航さんには、先行研究を参考にさせて頂きました。また、CERN 滞在時には大変お世話になりました。本当にありがとうございました。

同じ研究室の仲間である盛武翔君、山本賢君、池田友哉君にも大変お世話になりました。特に同じ ATLAS のメンバーである山本賢君には GUI の開発等で協力して研究を進める場面もあり、心強く思いました。盛武翔君には、学部の頃からの友人として、相談に乗ってもらうことも多くありました。池田友哉君には、就職していた頃の経験からファームウェア開発についてアドバイスをもらうことも多く、大変参考になりました。皆さん本当にありがとうございました。

沖花研究室の荒川達哉君には、第 3 者の目線から本研究についてアドバイスをもらうこともあり、また学部の頃からの友人として、相談に乗ってもらうこともありました。本当にありがとうございました。

当研究室の先輩である武田彩希先輩には、博士課程進学に当たっての心構え等、参考になるお話を聴かせていただきました。本当にありがとうございました。

理科教育専修の皆様には、仲間として楽しい時間を過ごすことができたことを深く感謝しています。これからも変わらず仲良くしてもらえればありがたいです。

大学院生活の 2 年間、変わらず研究生活を支えてくれた家族には本当に感謝しています。家族の協力なしに本研究を進めることはできませんでした。

その他、この研究を進めるに当たってお世話になったすべての皆様に、お礼を申し上げます。

参考文献

[1] 遠藤理樹, [ATLAS 実験アップグレードに向けた新型シリコン検出器モジュールの読み出しシステムの開発], 大阪大学大学院理学研究科物理学専攻修士論文, (2012).

[2] 岸田拓也, [ATLAS 新型シリコン検出器開発用ビーム試験 DAQ の構築], 東京工業大学大学院理工学研究科基礎物理学専攻修士論文, (2012).

[3] 岡村航, [ATLAS 実験アップグレード用シリコン検出器テストシステムの開発およびプロトタイプ検出器の性能評価], 大阪大学大学院理学研究科物理学専攻修士論文, (2011).

[4] [Project Specication Project Name: ABC-N ASIC Version: 1.3.1], (2008).

[5] [Project Specification Project Name: ABCD3T ASIC Version: V1.2], (2000).

[6] D. Campb ell, et al. [THE ATLAS BINARY CHIP], 2nd Workshop on Electronics for LHC Exp eriments, CERN/LHCC/96-39(1997).

[7] J. Kaplon, et al. [The ABCN front-end chip for ATLAS Inner Detector Upgrade], Topical Workshop on Electronics for Particle Physics, Naxos, Greece,15-19 Sep 2008, pp.116-120(Published Version from CERN 2009).

[8] Tomohisa Uchida, Yasuo Arai, [SEABAS (Soi EvAluation BoArd with Sitcp) User ' s Manual], (2008).

[9] KEK SOI Pixel Detector R&D, [SEABAS(1) Main Board Schematics], (2008).

[10] KEK SOI Pixel Detector R&D, [SEABAS2 Main Board Schematics], (2012).

[11] Josuah Wicht, Herv‘e Fischer, et al. [Test of Buer Control Chip by FPGA] ‘Ecole d ’ ln‘enieurs et d ’ Architectes de Frib ourg part of the University of Applied Sciences Western Switzerland Bachelor thesis, (2009).

[12] S. Gonzalez-Sevilla, et al. [Electrical results of double-sided silicon strip modules for the ATLAS Upgrade Strip Tracker], ATL-UPGRADE-PUB-2012-002, (2012).

[13] A. Clark, [Double-Sided Super-Module R&D for the ATLAS Tracker at HL-LHC], submitted to Nuclear Instruments and Methods in Physics Research A, (2013).

[14] Sergio Diez, [Silicon strip staves and petals for the ATLAS Upgrade tracker of the HL-LHC], Nuclear Instruments and Methods in Physics Research A Volume 699 93-96, (2013).

[15] A. Clark, [Mechanical Studies towards a Silicon Micro-strip Super Module for the ATLAS Inner Detector upgrade at the High Luminosity LHC], submitted to Journal of Instrumentation, (2013).

[16] D. Cussans, [Description of the JRA1 Trigger Logic Unit (TLU), v0.2c], EUDET-Memo-2009-4, (2011)

[17] TEXAS INSTRUMENTS, [SN55110A, SN75110A, SN75112 DUAL LINE DRIVERS], TEXAS INSTRUMENTS Datasheet documents.

[18] TEXAS INSTRUMENTS, [SN65LVDS32B, SN65LVDT32B, SN65LVDS3486B, SN65LVDT3486B, SN65LVDS9637B, SN65LVDT9637B, HIGH SPEED DIFFERENTIAL RECEIVERS], TEXAS INSTRUMENTS Datasheet documents.

[19] Xilinx, [Spartan-3A/3AN スタータ キット ボード ユーザー ガイド (UG334)]