衛星軌道上における環境放射線検出器CUBES データ圧縮機能の実装

広島大学理学部物理学科
高エネルギー宇宙・可視赤外天文学研究室
B176073　山口 拓真
主査：高橋 弘充
副査：三好 隆博

2021 年 2 月

概 要

スウェーデン王立工科大学主導で 2021 年に打ち上げ予定のキューブサット MIST (MIniature STudent satellite) に搭載される CUBES (CUBesat x-ray Explorer using Scintillators) は、衛星軌道上で環境放射線を検出する。MIST ではデータ通信速度の制限から、放射線観測データをすべて地上に送信することができない。そこで、地上からのコマンドで可変なデータ圧縮機能を CUBES に実装することが本研究の目的である。

CUBES の CPU である Cortex-M3 において、ビンまとめによるデータ圧縮機能を C 言語で実装した。ビンまとめ方式は、全エネルギー帯域で同じビン幅を設定する（等間隔）方式と、高エネルギー帯域ほどビン幅を広く設定するログテーブル方式の 2 つである。図 1 では、2048 ビンある元データを示しており、図 2 では、元データを 128 ビンに圧縮したシミュレーション結果を示している。

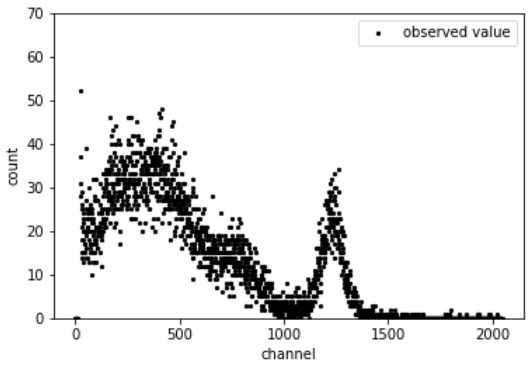

図 1: 元データ

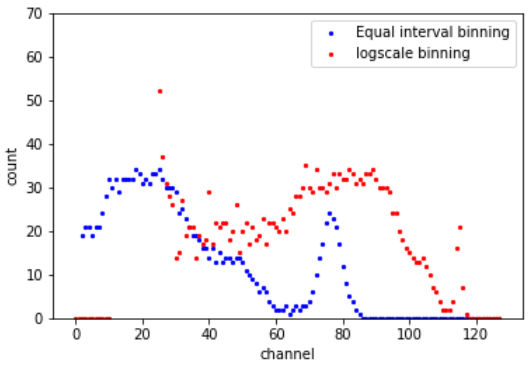

図 2: 等間隔（青）、テーブル（赤）ビンまとめ

目 次

図 目 次

第1章　序論

1.1　背景

MIST (MIniature STudent satellite) は、スウェーデン王立工科大学主導のプロジェクトで製作されているキューブサット（数 kg 程度の超小型人工衛星）であり、そのサイズは $10 \times 10 \times 30$ cm である。打ち上げは 2021 年を予定しており、MIST の構成とイメージをそれぞれ表 1.1、図 1.1 に示す。

MIST に搭載される CUBES (CUBesat x-ray Explorer using Scintillators) は、衛星軌道上の環境放射線を測定する検出器である。CUBES に使用されている GAGG[1] と MPPC 検出器、ASIC[2] は宇宙での使用経験がなく、放射線測定を通して、多くの陽子が衝突する衛星軌道上の環境での可用性の検証もミッションの目的である。

名称	説明
CubeProp	推進システムプロトタイプ
RATEX-J	質量分析計
Piezo LEGS	圧電リニアモーター
CUBES	放射線バックグラウンド測定器
SiC in Space	SiC オペアンプ
MoreBac	衛星軌道上でのバクテリア蘇生
SEUD	エラートラップ

表 1.1: MIST 構成 [2]

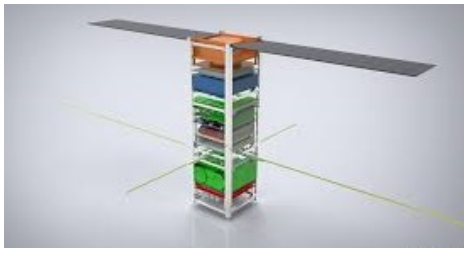

図 1.1: MIST イメージ [2]

1.2　研究目的

MIST のようなキューブサットでは、地上に放射線観測データをすべて送信できるほどデータ通信速度に余裕がなく、送信前にデータを圧縮する必要がある。一方で、衛星軌道上のエリアやエネルギースペクトルの帯域によっては、できるだけ重要なデータを圧縮せずに送信することが望ましい場合もある。したがって本研究では、地上から受信したコマンドに応じた可変なデータ圧縮機能を CUBES に実装することが目的である。

[1] 無機結晶シンチレータ　$Gd_3Al_2Ga_3O_{12}$
[2] 特定の用途のために設計された集積回路

1.3　衛星軌道上の放射線環境

放射線には、荷電粒子、陽子、電子、中性子などからなる高エネルギー粒子線と、X 線や γ 線の短波長電磁波がある。この節では、CUBES で測定する宇宙線や X 線、 γ 線について説明する。

1.3.1　宇宙線

宇宙空間には、（一次）宇宙線と呼ばれる高エネルギー粒子が飛び回っている。観測される宇宙線の割合は（電荷を持たない γ 線、宇宙ニュートリノなどは除く）、陽子が約 89%、α 線が約 10%、これらより重い原子核が約 1%であり、残りのほとんどを電子が占める。図 1.2 に、宇宙線のエネルギースペクトルを示す。そのエネルギー帯域は、$10^8 \sim 10^{20}$ eV のオーダーであり、その起源としては約 10^9 eV が太陽、$10^9 \sim 10^{15}$ eV が銀河系内、$10^{18} \sim$ eV が銀河系外と考えられている。また、観測される宇宙線の個数は、エネルギーが高くなるほど少なくなる。

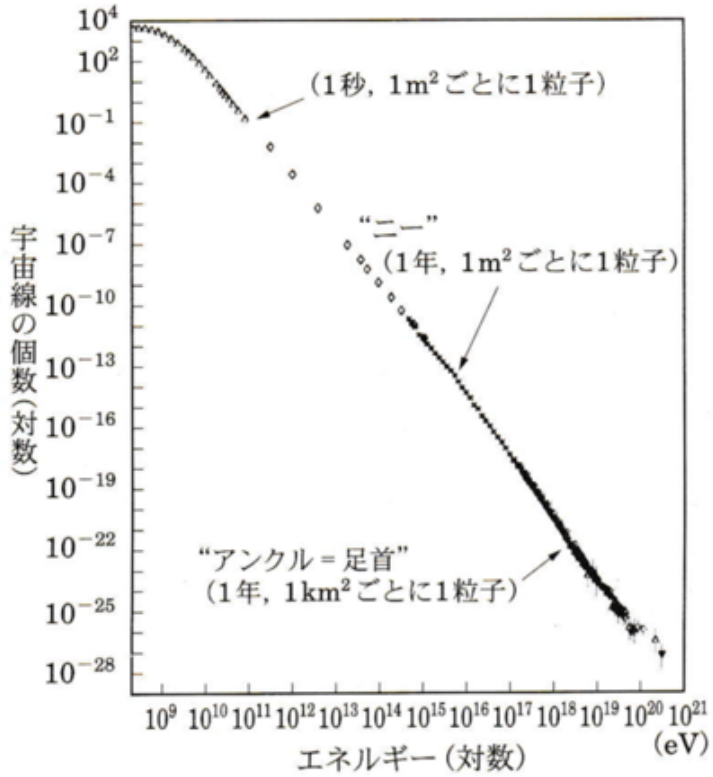

図 1.2: 宇宙線のエネルギースペクトル [6]

1.3.2　地磁気と荷電粒子

地磁気（地球の磁場）は、地下およそ 2266 km にある外核の対流運動によって発生する。これは、外核の主成分である導電性の高い鉄が高温の融解状態で存在し、流体運動で電流を生じることに起因する。図

1.3 に、地表の磁場強度分布を示す。図中の紫色の部分は特に磁場が弱い帯域であり、南大西洋地磁気異常帯と呼ばれる。南大西洋地磁気異常帯や極地においては、地磁気の高度が低くなっているため、荷電宇宙線が数多く存在する。一方で赤道付近においては、地磁気の高度が他に比べて高く、荷電宇宙線が磁力線の回りにトラップされることで、衛星軌道上への侵入を抑制している。

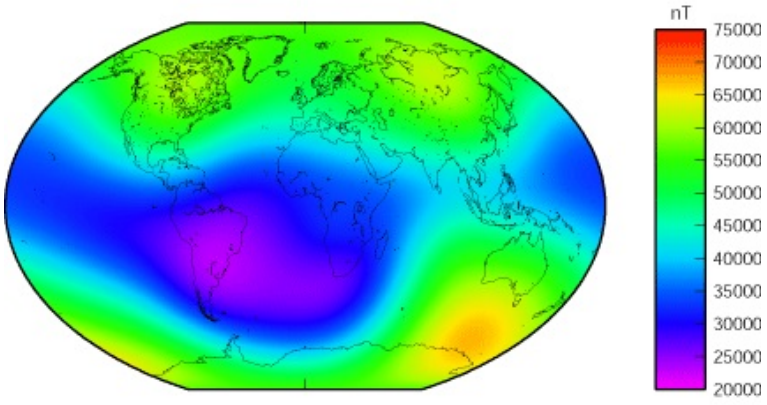

図 1.3: 地表の磁場強度分布 [7]

1.3.3 二次宇宙線

衛星軌道上における放射線は、二次宇宙線によるものが支配的である。二次宇宙線は、地球の大気圏に侵入した一次宇宙線が大気中の窒素や酸素の原子核と衝突することによって生成する。衝突後の二次宇宙線には、陽子、中間子、パイ中間子、ミュー粒子などがある。これらの二次宇宙線はさらに他の原子核と衝突し、多数の二次粒子を生成させる。

1.3.4 X 線

制動 X 線 高速な電子が散乱され方向を変えるとき、もしくは減速されるとき制動 X 線を放出する。制動 X 線のエネルギーは、連続スペクトルである。

特性 X 線 原子の軌道電子は、クーロン力によって束縛されている。放射線の照射などによって励起した軌道電子が基底状態に戻るとき、もしくは電離した電子の空席を埋めるために外側の電子が遷移したとき、軌道エネルギーの差に相当する特性 X 線を放出する。特性 X 線のエネルギーは、軌道エネルギーの差によって決定するため、離散的な値となる。

シンクロトロン放射 磁場によって拘束された電子は磁力線の回りを円運動しながら、電磁波を放出する。特に、電子の速度が光速に近い場合をシンクロトロン放射と呼ぶ。

1.3.5　γ線

ガンマ遷移　アルファ壊変やベータ壊変といった放射性崩壊後の原子核が、励起状態から安定な準位に遷移するとき、エネルギーを光子として放射する。この機構がγ線遷移であり、放射される光子がγ線である。本研究のシミュレーションで用いた ^{137}Cs のガンマ崩壊を例として、図 1.4 で説明する。

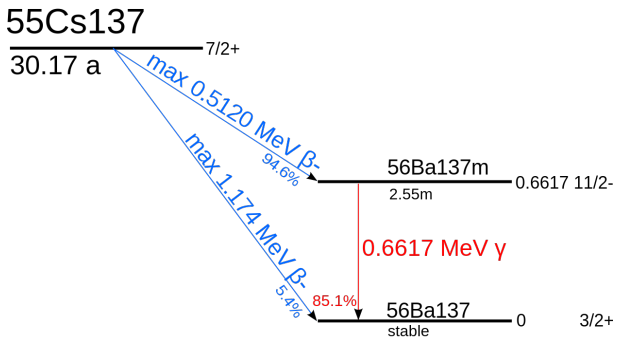

図 1.4: ^{137}Cs の放射性壊変 [5]

^{137}Cs は、ベータ壊変によって 95 %が準安定同位体の ^{137m}Ba に、残りの 5 %が基底状態の ^{137}Ba となる。その後、^{137m}Ba が ^{137}Ba へとガンマ遷移するとき、約 662 keV のγ線を放射する。

1.4　無機結晶シンチレータによる放射線測定

CUBES に搭載する GAGG と本研究のソフトウェア動作検証のために用いた CsI は、無機結晶シンチレータである。この節では、無機結晶シンチレータの発光原理と、MPPC による光子検出の機構ついて説明する。

1.4.1　無機結晶シンチレータの発光機構

図 1.5 に示すように、無機結晶シンチレータ内の電子のエネルギー状態は、格子上に束縛されている価電子帯、結晶内を自由に移動する伝導帯、そして電子の存在しない禁制帯に分かれる。放射線の入射により価電子帯の電子が励起されると、伝導帯とその下部にある幅数 10 eV ほどの励起子帯に自由電子が移動し、価電子帯には正孔が残る。その後、伝導帯の電子が価電子帯に戻るとき、そのエネルギーギャップに相当する波長のシンチレーション光を放出する。また、結晶に活性体が添加されているときは、禁制帯内に特別なエネルギー準位を形成する。この場合、励起子帯にある電子と正孔が静電的に結合した励起子、電離した電子や正孔によって活性体が励起状態へと遷移し、それが基底状態に戻るとき、シンチレーション光子を放出する。無機結晶シンチレータにおいては、この活性体の励起が主な発光の要因である。

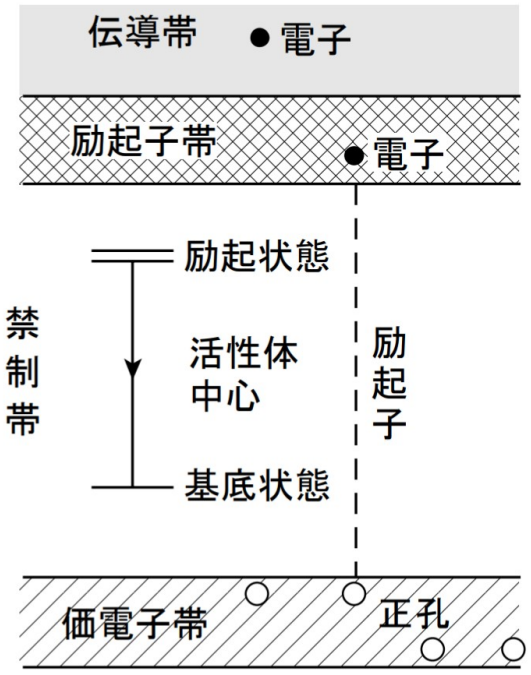

図 1.5: 無機シンチレータの発光機構 [9]

1.4.2 MPPC 検出器

MPPC (Multi-Pixel Photon Counter) は、フォトンカウンティング（光子計測）デバイスである。MPPC のフォトンカウンティングのイメージを、図 1.6 に示す。MPPC のフォトンカウンティングは、1 ピクセルに１つのフォトンが入ると、それに応じたパルスを出力する機構である。また、複数のピクセルでフォトンが検出された場合は、同じパルスが重ね合わされ、検出したフォトン数に応じた波高値を出力する。

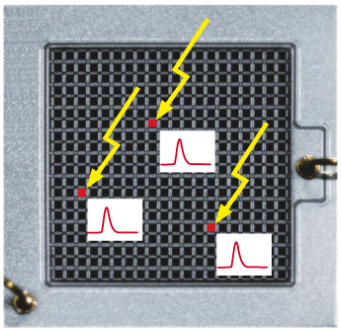

図 1.6: MPPC のフォトンカウンティング [8]

第2章　CUBESにおける放射線測定

2.1　構成とミッション

CUBES は地球低軌道における環境放射線検出器であり、国際宇宙ステーションと同様のおよそ 300~500 km の高度を予定している。CUBES の主な構成は 1cm 角の GAGG、MPPC、Citiroc ASIC（Weeroc 社製 Citiroc 製品）、ADC（アナログデジタル変換回路）、SoC (System on a Chip) である。それらを PBC（プリント基板）に設置した CUBES PBC の写真を、図 2.1 に示す。CUBES PBC には、図中にある黄色の GAGG シンチレータと MPPC 検出器が３つ配置されており、CUBES は CUBES PBC が２セットで構成される。したがって、GAGG シンチレータは合計で 6 個搭載される。

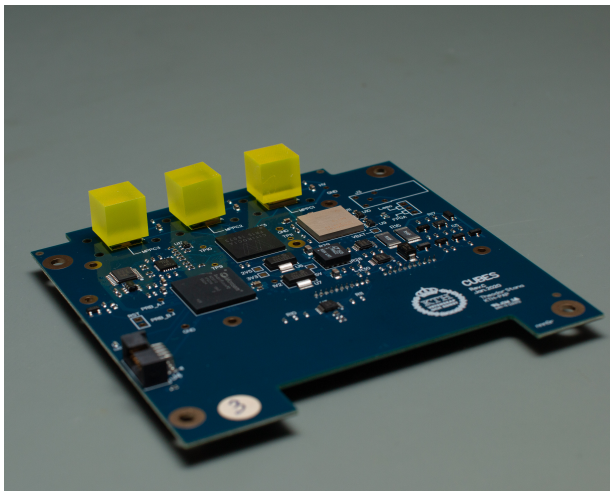

図 2.1: CUBES PBC

以下では、CUBES のミッションについて説明する。衛星軌道上の放射線強度や性質は、衛星の位置によって大きく異なる。1.3.2 で述べたように、赤道付近では宇宙線の個数が少ない一方で、極地や南大西洋地磁気異常帯では荷電粒子である陽子や電子が数多く存在する。この環境は、太陽活動などの影響により随時変化しているため、CUBES によって最新のデータを取得する。また、宇宙での使用経験のない GAGG シンチレータや ASIC の可用性を検証することもミッションの１つである。これらの測定は、CUBES で観測された放射線のヒストグラムデータを地上に送信することで実現する。したがって次の節では、放射線データを地上に送信するまでの工程を説明する。

2.2　放射線測定の流れ

CUBES の放射線測定の流れを表したダイアグラムを、図 2.2 に示す。観測された放射線をデータとして地上に送信するまでの工程は、以下のような流れである。

1. GAGG シンチレータによって、X 線やガンマ線が可視光へと変換される。
2. MPPC で光子を検出し、光子数に応じたパルスを出力する。
3. Citiroc ASIC でパルスの波高整形と増幅を行う。
   ASIC ではスレッシュホールド（閾値となる電圧）を設定することができ、その電圧以上のエネルギーをもったパルスだけをアナログデジタル変換することが可能である。
4. ADC によってアナログ信号をデジタル信号へと変換する。
5. SoC 内部の FPGA により、1 イベントごとにデータが蓄積し、Histo-RAM でヒストグラムを作成する。
6. 一定時間を積分した後、SoC 内部の CPU である Cortex-M3 でヒストグラムデータを処理し、インターフェースである MSP(MIST Space Protocol) を介して CUBES 外部の OBC (On-Board Computer) に送信する。
   OBC では、地上へのデータ送信や地上からのコマンド受信などを行っている。

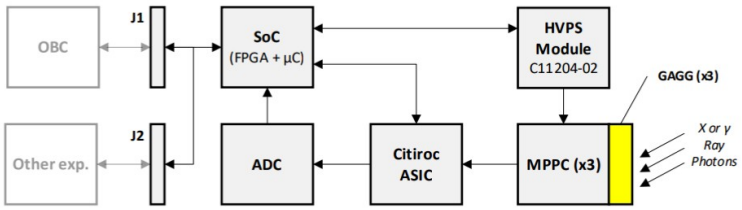

図 2.2: CUBES 放射線測定の流れ [4]

スウェーデン王立工科大学では、CUBES を地上で動作させる際に衛星 OBC を介さず、オペレーターとして Weeroc 社製のソフト (Citiroc UI) を利用している。

2.3　ヒストグラムデータの処理

CUBES の放射線観測によって取得されたヒストグラムデータは、図 2.2 で示した処理の工程を通して地上に送信される。この節では、1 つの CUBES PBC における場所ごとのデータ処理過程について説明する。

2.3.1　Citiroc ASIC・Dual ADC

Citiroc ASIC と Dual ADC では、MPPC からのパルスを以下のような流れで処理する。

1. 電荷信号を電圧信号へ変換する。MPPC からの 1 入力につき、増幅率の高い HG (High Gain) と低い LG (Low Gain) の 2 つの電圧信号を出力する。ASIC では、32 個の MPPC 信号を処理することができ、それぞれから hg, lg を出力するため、計 64 信号を処理する。

2. 計 64 の HG, LG 信号のいずれについて、スレッシュホールドを超えたエネルギーの信号が MPPC から検出されると、全 64 信号の電圧値をピークホールドする。

3. ピークホールドした電圧値をアナログ値として、1 チャンネル毎に HG, LG を 1 セットとして、ADC に出力する。

4. ADC におけるアナログデジタル変換の機構を、図 2.3 に示す。図中では、ASIC から出力された 2 つ (HG と LG) のピンを out_hg、out_lg、波形を読み取るタイミングを clk_read としている。ASIC のピンである out_hg と out_lg から出力された電圧のアナログ値は、ADC において clk_read の 1 サイクル（<200 ns）の間に、12 bits のデジタル値へと変換され、それぞれが FPGA に出力される。32 個の入力を Ch.0-Ch.31 とした際に、CUBES で 3 個の GAGG が接続されているチャンネルは Ch.0、Ch.16、Ch.31 である。

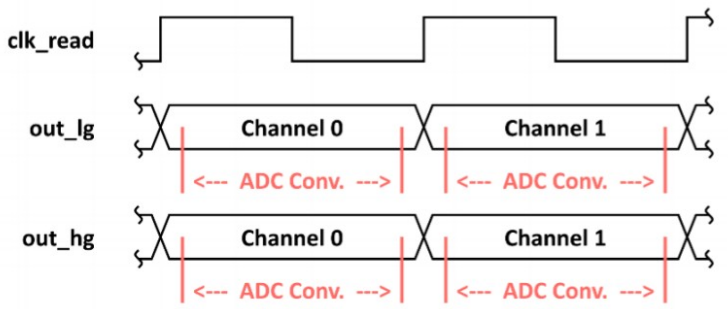

図 2.3: ADC によるアナログデジタル変換 [4]

2.3.2 Histo-RAM

ADC から出力された HG（高ゲインで増幅された信号）と LG（低ゲインで増幅された信号）は、FPGA 内の Histo-RAM に蓄積する。Histo-RAM では、HG と LG のそれぞれ 3 チャンネル (Ch.0、Ch.16、Ch. 31) から書き込まれたデータにより、合計 6 ヒストグラムが作成される。このとき最下位の 1 ビットを削除するビットシフトによって、12 bits (4096 ビン) のチャンネルは 11 bits (2048 ビン) に変換される。それぞれのエネルギー帯域としては、LG では 0∼1 MeV、HG では 0∼300 keV を予定している。1 つのヒストグラムは 2048 ビンで構成され、1 ビンあたり 2 bytes の容量（0∼65535 の値域）をもつ。ここまでで述べた Histo-RAM におけるデータ構成を、表 2.1 に示す。ヘッダー情報を含めると、1 サイクル（60 秒積分）のデータ取得におけるデータ容量は、**24832 bytes** である。

バイトインデックス	バイト数	内容
0〜255	256	ヘッダー情報
256〜4351	4096	ヒストグラム (ASIC Ch.0) HG
4352〜8447	4096	ヒストグラム (ASIC Ch.0) LG
8448〜12543	4096	ヒストグラム (ASIC Ch.16) HG
12544〜16639	4096	ヒストグラム (ASIC Ch.16) LG
16640〜20735	4096	ヒストグラム (ASIC Ch.31) HG
20736〜24831	4096	ヒストグラム (ASIC Ch.31) LG

表 2.1: Histo-RAM におけるデータの構成 [4]

2.3.3 APB・Cortex-M3

1 サイクルのデータ取得が完了すると、APB の制御によって Histo-RAM のヒストグラムデータが読み取られる。その後、SoC 内の CPU である Cortex-M3 でデータ処理が行われる。Histo-RAM から APB (Advanced Peripheral Bus) を介して、Cortex-M3 で行われるデータ処理の過程を、図 2.4 に示す。図のように Histo-RAM では、データ幅を 2 bytes から 4 bytes (32 bits) にシフトさせ、APB でそのデータを読み取る。さらに Cortex-M3 では、それらのデータを 1 bytes ごとに分けて、OBC へと送信する。

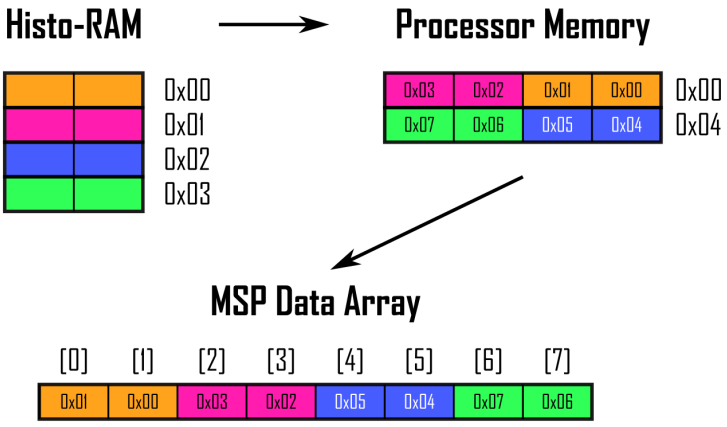

図 2.4: ヒストグラムデータの処理過程 [3]

第3章　データ圧縮プログラムの実装

3.1　データ圧縮の目標

CUBES の稼働時間は 1 サイクル（4 日間）あたり 14 時間である。また、1 回のデータ取得にかかる時間は約 61 秒（60 秒の積分時間＋データ処理 <1 秒）であることから、1 サイクルあたりのデータ取得回数を以下のような計算で求めることができる。（小数点以下切り上げ）

$$(14 \times 60 \times 60)/61 \simeq 827 \tag{3.1}$$

よって、１台の CUBES PBC が OBC で使用可能なデータストレージ容量を A bytes とすると、1 回のデータ取得で保存できる最大のヒストグラムデータ量と 1 ヒストグラムあたりのビン数を、以下のような計算で導出することができる。

$$\text{ヒストグラムデータ量} = \mathrm{A}/827 \ \text{bytes} \tag{3.2}$$

ヒストグラムデータにはヘッダー情報が 256 bytes あり、1 ビンあたり 2 bytes のデータ量をもつヒストグラムが 6 つ作成されるから、

$$\text{ビン数} = (\mathrm{A}/827 - 256)/(2 \times 6) \ \text{bins} \tag{3.3}$$

また MIST の通信速度には、ダウンリンク が最大 9.6 kbps と制限があり、それによって CUBES PBC が使用できるデータストレージ容量は、2 MB、2.5 MB、3 MB の 3 つが想定されている。[1] それぞれに対応したデータ量と 1 ヒストグラムあたりのビン数を、表 3.1 に示す。

データストレージ容量 (bytes)	ヒストグラムデータ (bytes)	ビン数 (bins)
2,000,000	2,418	180
2,500,000	3,022	230
3,000,000	3,627	280

表 3.1: データストレージ容量に対して取得可能なヒストグラムデータ量とビン数

2.3.2 で述べたように、圧縮前のヒストグラムデータは 24832 bytes であり、表 3.1 で示した取得可能なデータ量を超過している。したがって、最も少ないデータストレージ容量では **2418 bytes 以下**、最も多い容量では **3627 bytes 以下**までデータを圧縮することが目標となる。

3.2　ビンニングによるヒストグラムデータの圧縮

ヒストグラムデータの圧縮は、複数のビンを 1 つのビンにまとめる**ビンニング**で行っており、その処理の流れを以下に示す。

1. 地上から受信したコマンドの値によって、各ヒストグラムのビン二ング方式を決定する。

2. それぞれのビン二ング方式で定義されているビン幅の範囲で、カウント数の総和をとる。

3. カウント数の総和をビン幅で割り、平均をとる。

4. 導出した値を新たなデータとして代入する。

5. 全てのヒストグラムデータをビン二ングするまで、2.~4. の行程を繰り返す。

ビン幅を決定するビン二ング方式には、**等間隔ビン二ング**と**テーブルビン二ング**の 2 パターンを採用している。等間隔ビン二ングにおいては、全エネルギー帯域でビン幅を一定としている。一方でテーブルビン二ングにおいては、テーブルを用いてビン幅をログスケールに設定し、高エネルギー帯域ほどビン幅を広くしている。それぞれのビン二ング方式における詳しい処理の流れについては、3.4.1 と 3.5.1 で述べる。

3.3 実装場所の検討

2.2 で述べたように、CUBES のデジタル処理は FPGA の Histo-RAM と CPU で行われる。したがってこれらを比較することで、ヒストグラムデータの圧縮機能を実装する場所を決定する。

表 3.2 に実装場所の優位性を比較した結果を示す。CUBES の CPU においては、1 秒より小さいオーダーで処理が可能な見込みであり、データ取得のサイクルに影響を及ぼすほどの処理時間ではないと考えられる。実際にハードウェアを用いて検証した処理時間については、3.6.4 で説明する。FPGA においては、複数の処理を並列して行うことができるため、CPU よりも高速な処理が可能である。よって処理時間においては、両者とも問題ないと考えられる。

改修規模については図 2.4 で示したように、OBC に近い CPU の Cortex-M3 で、機能の実装をより柔軟に行うことが可能である。一方で、Histo-RAM から OBC までには APB など複数の処理過程があるため、FPGA での実装は難しいと考えられる。したがって、データ圧縮機能を実装する場所を SoC 内部の CPU である、Cortex-M3 に決定した。

場所	処理時間	改修規模	実装
CPU	○	○	○
FPGA	◎	×	×

表 3.2: 実装場所の優位性の比較

3.4 等間隔ビン二ング機能の開発

3.4.1 等間隔ビン二ングの処理の流れ

等間隔ビン二ングの処理過程を、巻末の付録 ListingA.1 に示す。ソースコードでは、ビン二ング方式を決定するコマンドである引数を bin_config、ビン幅を bin_size、ビン数を num_bins、総データ量を len と定義している。以下では、ListingA.1 で行われている処理過程について説明する。

1 行目 bin_config（引数）= 1, 2, 3, 4, 5, 6 のとき、等間隔ビン二ングを実行する。
bin_config = 0 のときはビン二ングを行わない。

3、4 行目 bin_size（ビン幅）と num_bins（ビン数）は、bin_config のビットシフトによって決定する。
bin_config = N とすると、bin_size = 2^N、num_bins = 2048/bin_size=$2048/2^N$ となる。

7∼10 行目 bin_size の範囲で総和をとる。
ビットシフトとビット論理積を用いて、4 bytes の幅をもつ APB のヒストグラムデータから、2 bytes のビンを取り出している。

11 行目 ビンの総和をビットシフトで平均化する。

12、13 行目 2.4 で示した方式で、平均化した値を send_data_payload（OBC に送信するデータ）に代入する。

5∼14 行目 for ループで num_bins の回数繰り返す。

15 行目 1 ヒストグラムのデータ量を len（総データ量）に加算する。

このように、bin_config に対応して決まる bin_size、1 ヒストグラムあたりの num_bins の値を表 3.3 に示す。

bin_config	bin_size	num_ bins
1	2	1024
2	4	512
3	8	256
4	16	128
5	32	64
6	64	32

表 3.3: 等間隔ビン二ングにおける引数、ビン幅、1 ヒストグラムあたりのビン数

3.4.2 等間隔ビン二ングシミュレーション結果

初めに、等間隔ビン二ングのシミュレーションとして、チャンネル 0∼2047 の階段状に増加するヒストグラムを用いたビン二ングを行った。このヒストグラムは、0 からカウント数が 8 ずつ増加し、そのグラフを図 3.1 に示す。

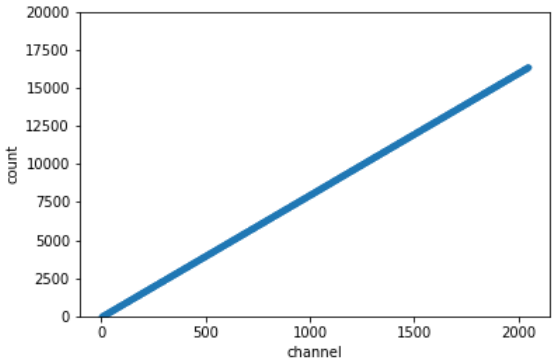

図 3.1: シミュレーションに用いた階段状ヒストグラム

シミュレーションによって得たビン二ング後の値を理論値と比較することで、等間隔ビン二ングが正しく機能しているかどうかを確認した。その結果の初めの部分を、表 3.4 に示す。

理論値の導出方法を以下のように行った。ビン二ング後の値は、公差 8 の等差数列の和を項数で割ったものとして考えることができる。このとき、初項と末項は総和をとるチャンネルの最初と最後の値を 8 倍したものであり、それぞれ $8 \times \mathrm{ch_f}$、$8 \times \mathrm{ch_l}$ と表す。また、総和をとる項数は bin_size（ビン数）の値に対応することから、総和は以下のような計算式となる。

$$\frac{\mathrm{bin\_size} \times 8 \times (\mathrm{ch_f} + \mathrm{ch_l})}{2} \tag{3.4}$$

さらに bin_size で割り平均化するので、理論値は $4 \times (\mathrm{ch_f} + \mathrm{ch_l})$ を用いて導出でき、この理論値どおりの出力となることを確認した。

bin_config	理論値	出力値
1	$4 \times (0+1) = 4$	4
2	$4 \times (0+3) = 12$	12
3	$4 \times (0+7) = 28$	28
4	$4 \times (0+15) = 60$	60
5	$4 \times (0+31) = 124$	124
6	$4 \times (0+63) = 252$	252

表 3.4: 等間隔ビン二ングにおける理論値と出力値の比較

次に、CsI シンチレータと MPPC 検出器で取得した ^{137}Cs エネルギースペクトルを用いて、等間隔ビン二ングのシミュレーションを行った。ビン二ング前のエネルギースペクトルを、図 3.2 に示す。また、bin_config に対応した、ビン二ング後のエネルギースペクトルを図 3.3 に示す。これらのヒストグラムから、チャンネル数が引数の値に応じて減少しており、ビン二ングによってデータ圧縮されていることが確認できる。さらに、最もデータ圧縮された bin_config=6 のヒストグラムにおいても、^{137}Cs のガンマ線ピークの概形を再現できている。

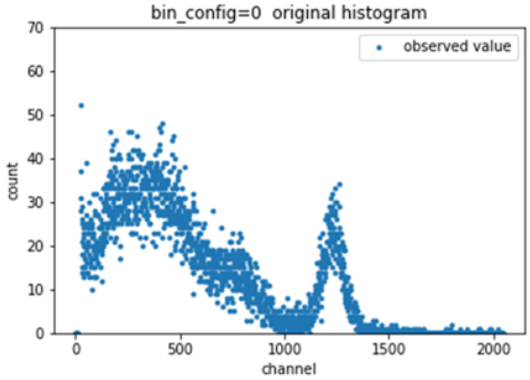

図 3.2: シミュレーションに用いた線源 ^{137}Cs のエネルギースペクトル

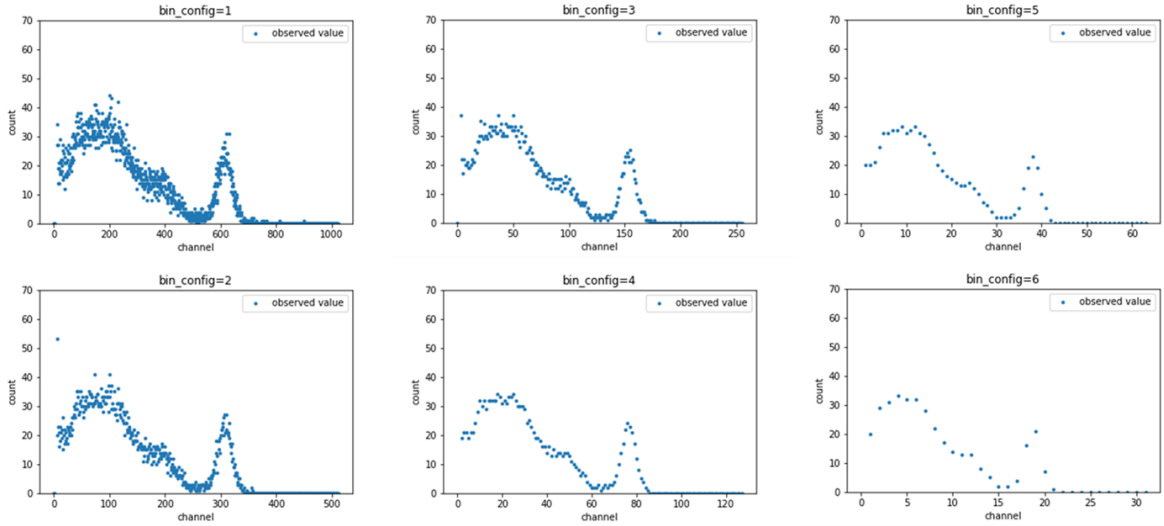

図 3.3: bin_config（各グラフのヘッダーに記載）の値に対応した等間隔ビン二ングのシミュレーション結果

3.5　テーブルビン二ング機能の開発

3.5.1　テーブルビン二ングの処理の流れ

テーブルビン二ングの処理過程を、付録 ListingA.2 に示す。今回のテーブルビン二ングでは、高エネルギー帯域になるにつれビン幅がログスケールで増大するテーブルを設定し、低エネルギー帯域で詳細なエネルギースペクトルを取得することが可能となる。ログスケールに増大するビン幅は、カウント数の総和をとるチャンネルの範囲をテーブルで区切ることによって決定され、以下でその詳細な定義について説明する。また、ソースコードでの table の値を付録 ListingA.3 に示す。

- bin_config = 11 のとき table1 を設定し、そのビン幅は {1, 1, 2, 4} のように増大する。
  各ビン幅に対して 256 ビンを設定するため、ビン数は $4 \times 256 = 1024$ となる。

- bin_config = 12 のとき table2 を設定し、そのビン幅は {1, 1, 2, 4, 8, 16, 32, 64} のように増大する。
  各ビン幅に対して 16 ビンを設定するため、ビン数は $8 \times 16 = 128$ となる。

- bin_config = 13 のとき、シミュレーション用の table3 を設定する。
  ビン幅は {1, 1, 2, 4, 7, 8, 9, 31, 32, 33, 63, 64, 65} のように増大し、1 より大きい奇数のビン幅を含んでいる。
  {1, 1, 2} のビン幅に対して 256 ビン、{4} のビン幅に対して 178 ビン、{7, 8, 9, 31, 32, 33, 63, 64, 65} のビン幅に対して 1 ビンを設定するため、ビン数は $3 \times 256 + 1 \times 178 + 9 \times 1 = 955$ となる。

以下では、ListingA.2 で行われている処理過程について説明する。

1 行目　bin_config（引数）= 11, 12, 13 のとき、ログスケールビン二ングが実行される。

5〜16 行目　bin_config の値に応じて、table を決定する。
bin_config = 11 のとき table1、bin_config = 12 のとき table2、bin_config = 13 のとき table3 をポインタで渡す。
また sizeof 演算子によって、設定される table 全体のメモリサイズを取得する。
それを table の 1 要素あたりのメモリサイズで割ることで、table の要素数を算出する。
さらに、table の要素数から 1 引いたものを、num_bin（ビン数）として代入する。

18 行目　table によって定義されている bin_size（ビン幅）を決定する。

19〜66 行目　table によって指定された範囲で、ヒストグラムデータの総和をとり、その後ビンの総和を bin_size で割り、平均化する。
このとき、carry_over と bin_size の値で処理を分けており、その工程を図 3.4 に示す。
図では例として、APB のヒストグラムデータを 0×00 から、それぞれ bin_size = 6（左）と bin_size = 5（右）の範囲で総和をとる場合を考える。
bin_size = 6 で総和をとったとき、0×08 まで全てのビンを取り出すことができる。
一方で、bin_size = 5 で総和をとったとき、ビット論理積によって Bin4 までが取り出され、Bin5 が残ってしまう。
このように、前の処理のビンが残っている状態のときを、carry_over = 1 とすることで判別している。
（carry_over = 0 のときは、前の処理でビンが残っていないことを表す。）

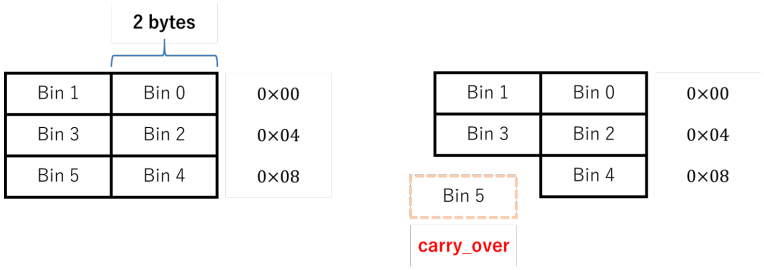

図 3.4: テーブルビン二ングにおける carry_over

carry_over の有無は、前回の処理での carry_over と bin_size の偶奇によって決定することができ、表 3.5 にその内訳を示す。

前の carry_over	bin_size	carry_over
0	偶数	0
0	奇数	1
1	偶数	1
1	奇数	0

表 3.5: carry_over の判別

さらに bin_size が奇数のときは、その値が 1 の場合と、そうでない場合で処理を分けている。
bin_size = 1 のときは複数のビンで総和をとる必要がないため、carry_over = 1 かつ bin_size = 1 のときは、前回 carry_over として残ったビンをビットシフトとビット論理積で取り出し、carry_over = 0 かつ bin_size = 1 のときは、新たにビット論理積で 1 ビンだけ取り出す。

67、68 行目 2.4 で示した方式で、平均化した値を send_data_payload に代入する。

17∼70 行目 for ループで num_bins の回数繰り返す。

71 行目 1 ヒストグラムのデータ量を len(総データ量) に加算する。

このように、bin_config の値に応じて決まる 1 ヒストグラムあたりの num_bins を、表 3.6 に示す。

bin_config	num_ bins
11	1024
12	128
13	955

表 3.6: テーブルビン二ングにおける引数とビン数

3.5.2 テーブルビン二ングシミュレーション結果

図 3.2 に示した ^{137}Cs のエネルギースペクトルを用いて、テーブルビン二ングのシミュレーションを行った結果を図 3.5 に示す。これらのヒストグラムから、等間隔ビン二ングと同様にチャンネル数が引数の値に応じて減少しており、ビン二ングによってデータ圧縮されていることが確認できる。また、図 3.3 で示したチャンネル数が同じ bin_config=1,4 のヒストグラムと比較すると、ログスケールのテーブルを用いたビン二ングでは低エネルギー帯域のエネルギースペクトルを詳細に再現できていることが確認できる。一方で高エネルギー帯域においては、特に bin_config=12 のヒストグラムで、ガンマ線ピークの概形が失われていることが分かる。

また、シミュレーション用のテーブルを設定した bin_config=13 においては、carry_over に応じた特別な処理が必要であったが、問題なくエネルギースペクトルの圧縮を行うことができた。bin_config=11,12 と同様に、チャンネル数が 955 まで減少しており、低エネルギー帯域のエネルギースペクトルを詳細に再現することができた。

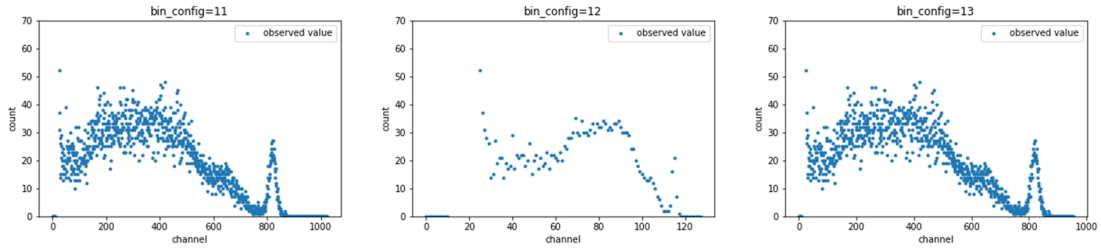

図 3.5: bin_config（各グラフのヘッダーに記載）の値に対応したテーブルビン二ングのシミュレーション結果

3.6 ビンニング機能の実装

3.6.1 ビンニング後のデータ量

1 回のデータ取得で作成される 6 ヒストグラムを、同じ方式でビンニングするとき、bin_config の値とビン数、総ヒストグラムデータ量は表 3.7 のようになる。

引数	ビン数	総データ量
1	1024×6	12544
2	512×6	6400
3	256×6	3328
4	128×6	1792
5	64×6	1024
6	32×6	640
11	1024×6	12544
12	128×6	1792
13	955×6	11716

表 3.7: 各ビン二ング方式における引数、ビン数、総データ量

3.1 で述べたように、最も少ないデータストレージ容量では、1 つの CUBES PBC で **2418 bytes 以下**までヒストグラムデータを圧縮する必要がある。したがって、**bin_config = 4, 5, 6, 12** に対応したビンニング方式では、データ圧縮の目標を達成しており、地上から送信するコマンドの決定の目安となる。

3.6.2　6 ヒストグラムに対応したビンニング機能の開発と実装

CUBES に実装するビンニング機能のソースコードを、ListingA.4 に示す。ソースコードでは、地上からビンニング方式を決定するコマンドである bin_config を要素数が 6 つの配列で定義し、それぞれが 1 ヒストグラムのビンニング方式を決定する。すなわち、1 回のデータ取得で作成される 6 つのヒストグラム（3 個の MPPC それぞれが、hg, lg の 2 ヒストグラムを持つ）を、各ヒストグラムごとに、ビン幅やビン数の異なったビンニング方式でビンニングすることが可能である。以下では、ListingA.4 で行われている処理過程について説明する。

19 行目 ~149 行目　テーブルを設定する。

162 行目 ~166 行目　ヒストグラムのヘッダー情報を send_data_payload に代入する。

167 行目　ヘッダー情報のデータ量を len に代入する。

170 行目 ~268 行目　6 ヒストグラムに対応したビン二ングを行う。bin_config[i] = 1, 2, 3, 4, 5, 6 のときは 3.4.1 で説明した等間隔ビン二ング、bin_config[i] = 11, 12, 13 のときは、3.5.1 で説明したテーブルビン二ングを行う。

181、198、270 行目　1 ヒストグラムのデータ量を加え、len に代入する。

169~272 行目　全てのヒストグラムをビン二ングするため、6 回繰り返す。

273 行目　戻り値として、総データ量である len を返す。

3.6.3　ハードウェアを用いたシミュレーション方法

ハードウェアを用いたビン二ング機能のシミュレーションは、スウェーデン王立工科大学で実施される。シミュレーションにおけるデータ送受信の流れを、図 3.6 に示す。図中の Citiroc UI は、検出器の設定コマンドを送信したり、データを受信して表示するために利用する。このオペレータは、地上での試験の際に利用している模擬システムである。Arduino Due は、MIST 衛星で地上との通信を制御する OBC のシミュレーターである。Cortex-M3 は、CUBES の SoC 内にある CPU で今回ビン二ング機能を実装した場所である。

また、実際に Citiroc UI と OBC シミュレータ、CUBES PBC 用いてシミュレーションを行っている様子を、図 3.7 に示す。

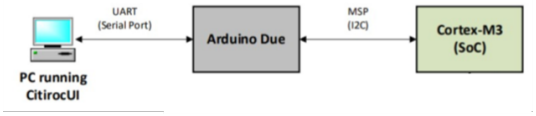

図 3.6: ハードウェアを用いたシミュレーションにおけるデータ送受信の流れ

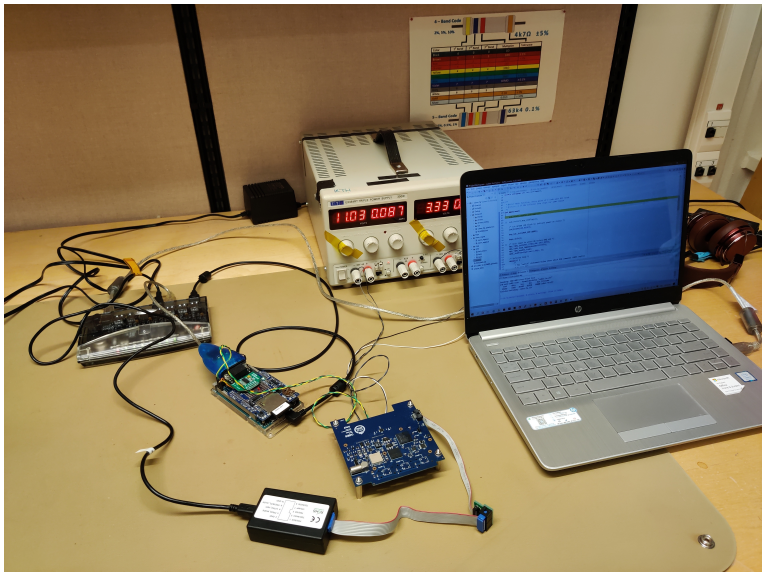

図 3.7: ハードウェアを用いたシミュレーションの様子

2.3.2 で述べたように、FPGA には Ch.0、Ch.16、Ch.31 が接続されており、それぞれ HG と LG を含んだ合計 6 ヒストグラムが作成される。ハードウェアを用いたシミュレーションでは、1 ヒストグラムごとに異なったビン二ング方式で圧縮を行い、Citiroc UI で受信されたヒスグラムデータが正確にビン二ングされていることを確かめる。

3.6.4　ハードウェアを用いたシミュレーション結果

bin_config=0,1,2,3,4,5 をコマンドとして、等間隔ビン二ング機能のシミュレーションを行った。テーブルビン二ングのシミュレーションは、今後行う予定である。ハードウェアを用いたシミュレーション結果として、HG と LG のヒストグラムを図 3.8 (Ch.31)、図 3.9 (Ch.16)、図 3.10 (Ch.0) に示した。ここでは、元のビン数である 2048 ビンを保持して表示するため、ビンまとめされたビンには、ビン幅の分だけ同じ値を補完する。

これらの図が示すように、1 ヒストグラムごとにビン二ング方式を決定するデータ圧縮が行われていることが分かる。さらに、bin_config の値が増大するにつれて、ヒストグラムのビン数が減少し、データの圧縮を確認できる。

また結果から、Individual DAQ time(1 回のデータ取得時間) が 20 秒、actual acq. time（正味のデータ取得時間）が 19 秒であることが分かる。この差は、CUBES からデータをダウンロードする時間を 1 秒に設定することで生じたタイムラグであり、ビン二ング機能実装後のデジタル処理が 1 秒未満で行われていることを示している。

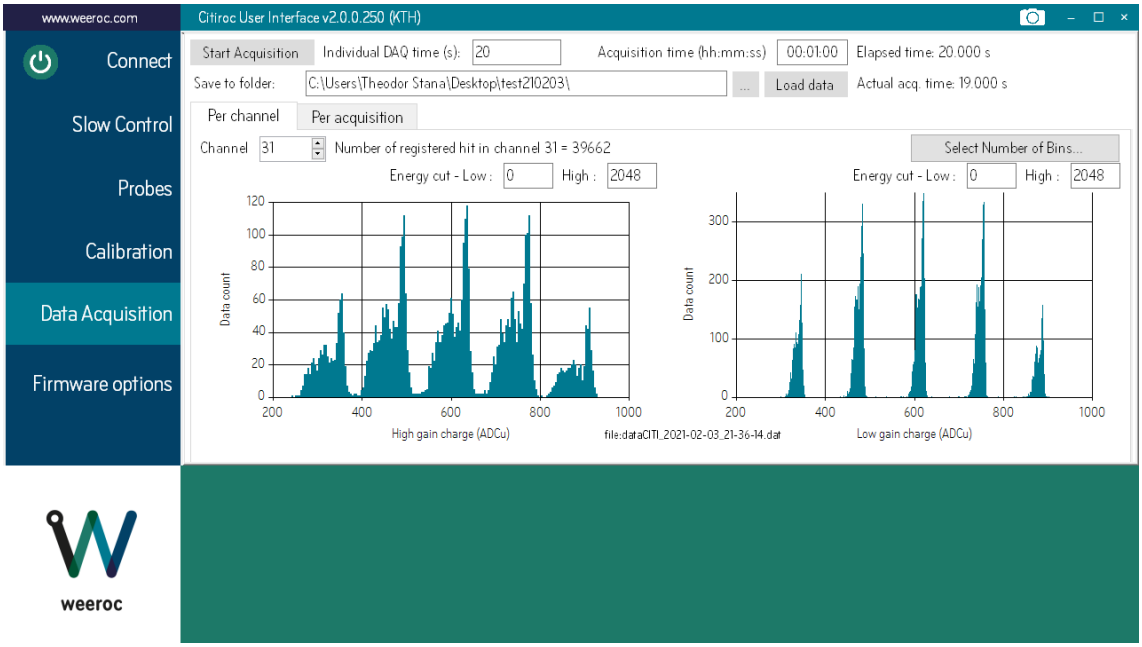

図 3.8: ハードウェアを用いたシミュレーション結果 Ch.31,HG bin_config=2（左）、LG bin_config=0（右）

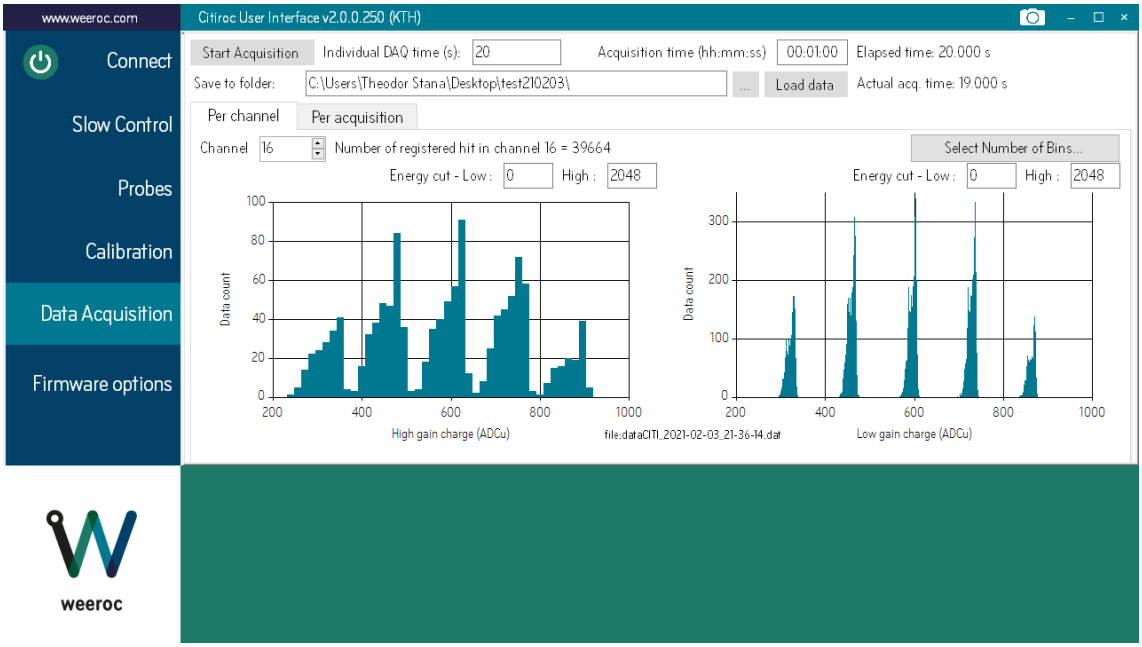

図 3.9: ハードウェアを用いたシミュレーション結果 Ch.16,HG bin_config=4（左）、LG bin_config=1（右）

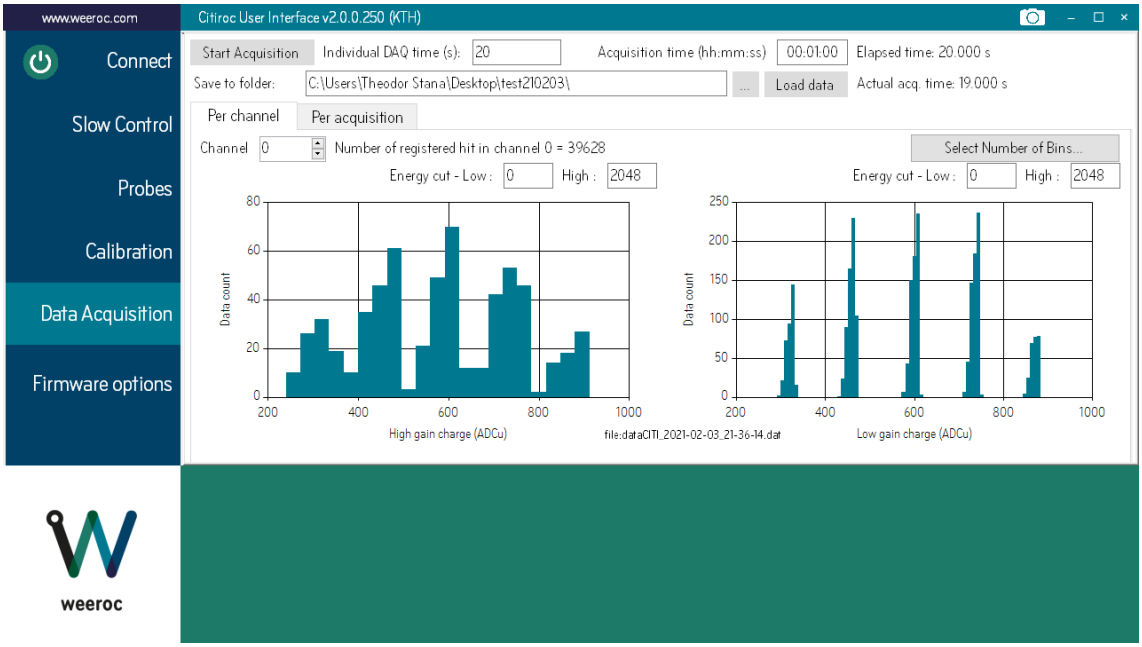

図 3.10: ハードウェアを用いたシミュレーション結果 Ch.0,HG bin_config=5（左）、LG bin_config=3（右）

第4章　まとめと今後

4.1　まとめ

本研究の目的は、衛星軌道上の環境放射線検出器である CUBES に、データ圧縮機能を実装することであった。本研究におけるまとめを、以下に示す。

- CUBES が使用可能なデータストレージ容量まで、ヒストグラムデータを圧縮する**ビンニング機能**を実装することができた。
- データ圧縮には、**等間隔ビンニング**と**テーブルビンニング**の 2 方式を開発し、地上からのコマンドによって可変な圧縮機能を実現した。
- 実験で得た放射線観測データや実機の CUBES PBC を用いたシミュレーションを通して、ビン二ングがうまく機能することを確認できた。

4.2　今後の課題

本研究を通して見つかった今後の課題について、以下で述べる。

- CUBES PBC を用いたシミュレーションにおいては、等間隔ビン二ングのみ検証を行ったため、今後ハードウェアを用いたテーブルビン二ングの検証を行う。
- 線源 ^{137}Cs のエネルギースペクトルを用いたテーブルビン二ングのシミュレーションにおいて、高エネルギー帯域の γ 線ピークの概形が失われてしまう問題点があった。そこで、地上からのコマンドによって観測したいエネルギー帯域のスペクトルを詳細に取得するビン二ング機能の開発が必要であると考える。例えば、^{137}Cs のエネルギースペクトルであれば、地上からのコマンドで高エネルギー帯域に小さい bin_size（ビン幅）をシフトさせることで、詳細な γ 線ピークの概形を取得することが可能となる。
  また、衛星軌道上でソフトウェアの書き換えが確立すれば、用途に応じたテーブルを追加することで、輝線構造などを詳細に取得するテーブルビン二ングを行うことも可能である。

付 録A

Listing A.1: 等間隔ビン二ング

```
else if (bin_config[i]<7) {
/* equal interval binning */
    num_bins = HISTO_NUM_BINS_GW>>bin_config[i];
        bin_size = 1<<bin_config[i];
        for (j=0; j<num_bins; j++) {
            bin = 0;
                for (k=j*bin_size/2; k<(j+1)*bin_size/2; k++) {
                        bin += (histo_data[k + i*HISTO_NUM_BINS_GW/2 +
                            HISTO_HDR_NUM_BYTES/4]>>16 & 0xFFFF)
                                        +(histo_data[k + i*HISTO_NUM_BINS_GW/2 +
                                            HISTO_HDR_NUM_BYTES/4] & 0xFFFF);
                }
                bin >>= bin_config[i];
                send_data_payload[j*2 + 1 + len] = bin & 0xFF;
                send_data_payload[j*2 + len] = (bin>>8) & 0xFF;
        }
    len = len + num_bins*2;
}
```

Listing A.2: テーブルビン二ング

```
else if (10<bin_config[i] && bin_config[i]<14) {
        /* logscale binning */
        unsigned char carry_over;
        carry_over = 0;
                if (bin_config[i] == 11) {
                        num_bins = sizeof (table1)/sizeof (table1[0]) - 1;
                        table = table1;
                }
                else if (bin_config[i] == 12) {
                        num_bins = sizeof (table2)/sizeof (table2[0]) - 1;
                        table = table2;
                }
                else if (bin_config[i] == 13) {
                    num_bins = sizeof (table3)/sizeof (table3[0])-1;
                    table = table3;
                }
                for (j=0; j<num_bins; j++) {
```

```
18                  bin_size = *(table+j+1) - *(table+j);
19                      if (carry_over == 0 && bin_size%2 == 0) {
20                              carry_over = 0;
21                              for (k=*(table+j)/2; k<*(table+j+1)/2; k++) {
22                                  bin = bin + (histo_data[k + i*HISTO_NUM_BINS_GW/2 +
                                        HISTO_HDR_NUM_BYTES/4]>>16 & 0xFFFF)
23                 + (histo_data[k + i*HISTO_NUM_BINS_GW/2 + HISTO_HDR_NUM_BYTES/4] & 0
                       xFFFF);
24                      }
25                              bin = bin/bin_size;
26                      }
27                      else if (carry_over == 0 && bin_size%2 == 1) {
28                              carry_over = 1;
29                              if (bin_size != 1) {
30                                      for (k=*(table+j)/2; k<(*(table+j+1)-1)/2; k
                                            ++) {
31                                  bin = bin + (histo_data[k + i*HISTO_NUM_BINS_GW/2 +
                                        HISTO_HDR_NUM_BYTES/4]>>16 & 0xFFFF)
32                     + (histo_data[k + i*HISTO_NUM_BINS_GW/2 + HISTO_HDR_NUM_BYTES/4]
                           & 0xFFFF);
33
34                              }
35                                      bin = bin + (histo_data[k + i*HISTO_NUM_BINS_GW
                                            /2 + HISTO_HDR_NUM_BYTES/4] & 0xFFFF);
36
37                                      bin = bin/bin_size;
38                              }
39                              else if (bin_size == 1) {
40                                  k = *(table+j)/2;
41                                      bin = histo_data[k + i*HISTO_NUM_BINS_GW/2 +
                                            HISTO_HDR_NUM_BYTES/4] & 0xFFFF;
42                              }
43                      }
44                      else if (carry_over == 1 && bin_size%2 == 0) {
45                              carry_over = 1;
46                              k = (*(table+j)-1)/2;
47                              bin = histo_data[k + i*HISTO_NUM_BINS_GW/2 +
                                    HISTO_HDR_NUM_BYTES/4]>>16 & 0xFFFF;
48                      for (k=(*(table+j)+1)/2; k<(*(table+j+1)-1)/2; k++) {
49                                  bin = bin + (histo_data[k + i*HISTO_NUM_BINS_GW/2 +
                                        HISTO_HDR_NUM_BYTES/4]>>16 & 0xFFFF)
50                  + (histo_data[k + i*HISTO_NUM_BINS_GW/2 + HISTO_HDR_NUM_BYTES/4] & 0
                       xFFFF);
51                      }
52                              bin = bin + (histo_data[k + i*HISTO_NUM_BINS_GW/2 +
                                    HISTO_HDR_NUM_BYTES/4] & 0xFFFF);
53                              bin = bin/bin_size;
54                      }
```

```
55                      else if (carry_over == 1 && bin_size%2 == 1) {
56                              carry_over = 0;
57                              k = (*(table+j)-1)/2;
58                              bin = histo_data[k + i*HISTO_NUM_BINS_GW/2 +
                                    HISTO_HDR_NUM_BYTES/4]>>16 & 0xFFFF;
59                              if (bin_size != 1) {
60                                      for (k=(*(table+j)+1)/2; k<*(table+j+1)/2; k
                                            ++) {
61                                          bin = bin + (histo_data[k + i*
                                                HISTO_NUM_BINS_GW/2 + HISTO_HDR_NUM_BYTES
                                                /4]>>16 & 0xFFFF)
62                      + (histo_data[k + i*HISTO_NUM_BINS_GW/2 + HISTO_HDR_NUM_BYTES/4]
                            & 0xFFFF);
63                                      }
64                                              bin = bin/bin_size;
65                              }
66                      }
67                      send_data_payload[j*2 + 1 + len] = bin & 0xFF;
68                      send_data_payload[j*2 + len] = bin>>8 & 0xFF;
69                      bin = 0;
70              }
71              len = len + num_bins*2;
72  }
```

Listing A.3: ビンニングに用いるテーブル

```
1  unsigned short int table1[1025] = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,
2         18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,
3         41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,
4         64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,
5         87,88,89,90,91,92,93,94,95,96,97,98,99,100,101,102,103,104,105,106,107,
6         108,109,110,111,112,113,114,115,116,117,118,119,120,121,122,123,124,
7         125,126,127,128,129,130,131,132,133,134,135,136,137,138,139,140,141,
8         142,143,144,145,146,147,148,149,150,151,152,153,154,155,156,157,158,
9         159,160,161,162,163,164,165,166,167,168,169,170,171,172,173,174,175,
10        176,177,178,179,180,181,182,183,184,185,186,187,188,189,190,191,192,
11        193,194,195,196,197,198,199,200,201,202,203,204,205,206,207,208,209,
12        210,211,212,213,214,215,216,217,218,219,220,221,222,223,224,225,226,
13        227,228,229,230,231,232,233,234,235,236,237,238,239,240,241,242,243,
14        244,245,246,247,248,249,250,251,252,253,254,255,256,257,258,259,260,
15        261,262,263,264,265,266,267,268,269,270,271,272,273,274,275,276,277,
16        278,279,280,281,282,283,284,285,286,287,288,289,290,291,292,293,294,
17        295,296,297,298,299,300,301,302,303,304,305,306,307,308,309,310,311,
18        312,313,314,315,316,317,318,319,320,321,322,323,324,325,326,327,328,
19        329,330,331,332,333,334,335,336,337,338,339,340,341,342,343,344,345,
20        346,347,348,349,350,351,352,353,354,355,356,357,358,359,360,361,362,
21        363,364,365,366,367,368,369,370,371,372,373,374,375,376,377,378,379,
22        380,381,382,383,384,385,386,387,388,389,390,391,392,393,394,395,396,
23        397,398,399,400,401,402,403,404,405,406,407,408,409,410,411,412,413,
```

```
        414,415,416,417,418,419,420,421,422,423,424,425,426,427,428,429,430,
        431,432,433,434,435,436,437,438,439,440,441,442,443,444,445,446,447,
        448,449,450,451,452,453,454,455,456,457,458,459,460,461,462,463,464,
        465,466,467,468,469,470,471,472,473,474,475,476,477,478,479,480,481,
        482,483,484,485,486,487,488,489,490,491,492,493,494,495,496,497,498,
        499,500,501,502,503,504,505,506,507,508,509,510,511,512,514,516,518,
        520,522,524,526,528,530,532,534,536,538,540,542,544,546,548,550,552,
        554,556,558,560,562,564,566,568,570,572,574,576,578,580,582,584,586,
        588,590,592,594,596,598,600,602,604,606,608,610,612,614,616,618,620,
        622,624,626,628,630,632,634,636,638,640,642,644,646,648,650,652,654,
        656,658,660,662,664,666,668,670,672,674,676,678,680,682,684,686,688,
        690,692,694,696,698,700,702,704,706,708,710,712,714,716,718,720,722,
        724,726,728,730,732,734,736,738,740,742,744,746,748,750,752,754,756,
        758,760,762,764,766,768,770,772,774,776,778,780,782,784,786,788,790,
        792,794,796,798,800,802,804,806,808,810,812,814,816,818,820,822,824,
        826,828,830,832,834,836,838,840,842,844,846,848,850,852,854,856,858,
        860,862,864,866,868,870,872,874,876,878,880,882,884,886,888,890,892,
        894,896,898,900,902,904,906,908,910,912,914,916,918,920,922,924,926,
        928,930,932,934,936,938,940,942,944,946,948,950,952,954,956,958,960,
        962,964,966,968,970,972,974,976,978,980,982,984,986,988,990,992,994,
        996,998,1000,1002,1004,1006,1008,1010,1012,1014,1016,1018,1020,1022,
        1024,1028,1032,1036,1040,1044,1048,1052,1056,1060,1064,1068,1072,1076,
        1080,1084,1088,1092,1096,1100,1104,1108,1112,1116,1120,1124,1128,1132,
        1136,1140,1144,1148,1152,1156,1160,1164,1168,1172,1176,1180,1184,1188,
        1192,1196,1200,1204,1208,1212,1216,1220,1224,1228,1232,1236,1240,1244,
        1248,1252,1256,1260,1264,1268,1272,1276,1280,1284,1288,1292,1296,1300,
        1304,1308,1312,1316,1320,1324,1328,1332,1336,1340,1344,1348,1352,1356,
        1360,1364,1368,1372,1376,1380,1384,1388,1392,1396,1400,1404,1408,1412,
        1416,1420,1424,1428,1432,1436,1440,1444,1448,1452,1456,1460,1464,1468,
        1472,1476,1480,1484,1488,1492,1496,1500,1504,1508,1512,1516,1520,1524,
        1528,1532,1536,1540,1544,1548,1552,1556,1560,1564,1568,1572,1576,1580,
        1584,1588,1592,1596,1600,1604,1608,1612,1616,1620,1624,1628,1632,1636,
        1640,1644,1648,1652,1656,1660,1664,1668,1672,1676,1680,1684,1688,1692,
        1696,1700,1704,1708,1712,1716,1720,1724,1728,1732,1736,1740,1744,1748,
        1752,1756,1760,1764,1768,1772,1776,1780,1784,1788,1792,1796,1800,1804,
        1808,1812,1816,1820,1824,1828,1832,1836,1840,1844,1848,1852,1856,1860,
        1864,1868,1872,1876,1880,1884,1888,1892,1896,1900,1904,1908,1912,1916,
        1920,1924,1928,1932,1936,1940,1944,1948,1952,1956,1960,1964,1968,1972,
        1976,1980,1984,1988,1992,1996,2000,2004,2008,2012,2016,2020,2024,2028,
        2032,2036,2040,2044,2048};

unsigned short int table2[129] = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,
        18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,34,36,38,40,42,44,46,48,
        50,52,54,56,58,60,62,64,68,72,76,80,84,88,92,96,100,104,108,112,116,
        120,124,128,136,144,152,160,168,176,184,192,200,208,216,224,232,240,
        248,256,272,288,304,320,336,352,368,384,400,416,432,448,464,480,496,
        512,544,576,608,640,672,704,736,768,800,832,864,896,928,960,992,1024,
        1088,1152,1216,1280,1344,1408,1472,1536,1600,1664,1728,1792,1856,1920,
```

```
        1984,2048};

unsigned short int table3[956]={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,
        142,143,144,145,146,147,148,149,150,151,152,153,154,155,156,157,158,
        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,
        193,194,195,196,197,198,199,200,201,202,203,204,205,206,207,208,209,
        210,211,212,213,214,215,216,217,218,219,220,221,222,223,224,225,226,
        227,228,229,230,231,232,233,234,235,236,237,238,239,240,241,242,243,
        244,245,246,247,248,249,250,251,252,253,254,255,256,257,258,259,260,
        261,262,263,264,265,266,267,268,269,270,271,272,273,274,275,276,277,
        278,279,280,281,282,283,284,285,286,287,288,289,290,291,292,293,294,
        295,296,297,298,299,300,301,302,303,304,305,306,307,308,309,310,311,
        312,313,314,315,316,317,318,319,320,321,322,323,324,325,326,327,328,
        329,330,331,332,333,334,335,336,337,338,339,340,341,342,343,344,345,
        346,347,348,349,350,351,352,353,354,355,356,357,358,359,360,361,362,
        363,364,365,366,367,368,369,370,371,372,373,374,375,376,377,378,379,
        380,381,382,383,384,385,386,387,388,389,390,391,392,393,394,395,396,
        397,398,399,400,401,402,403,404,405,406,407,408,409,410,411,412,413,
        414,415,416,417,418,419,420,421,422,423,424,425,426,427,428,429,430,
        431,432,433,434,435,436,437,438,439,440,441,442,443,444,445,446,447,
        448,449,450,451,452,453,454,455,456,457,458,459,460,461,462,463,464,
        465,466,467,468,469,470,471,472,473,474,475,476,477,478,479,480,481,
        482,483,484,485,486,487,488,489,490,491,492,493,494,495,496,497,498,
        499,500,501,502,503,504,505,506,507,508,509,510,511,512,514,516,518,
        520,522,524,526,528,530,532,534,536,538,540,542,544,546,548,550,552,
        554,556,558,560,562,564,566,568,570,572,574,576,578,580,582,584,586,
        588,590,592,594,596,598,600,602,604,606,608,610,612,614,616,618,620,
        622,624,626,628,630,632,634,636,638,640,642,644,646,648,650,652,654,
        656,658,660,662,664,666,668,670,672,674,676,678,680,682,684,686,688,
        690,692,694,696,698,700,702,704,706,708,710,712,714,716,718,720,722,
        724,726,728,730,732,734,736,738,740,742,744,746,748,750,752,754,756,
        758,760,762,764,766,768,770,772,774,776,778,780,782,784,786,788,790,
        792,794,796,798,800,802,804,806,808,810,812,814,816,818,820,822,824,
        826,828,830,832,834,836,838,840,842,844,846,848,850,852,854,856,858,
        860,862,864,866,868,870,872,874,876,878,880,882,884,886,888,890,892,
        894,896,898,900,902,904,906,908,910,912,914,916,918,920,922,924,926,
        928,930,932,934,936,938,940,942,944,946,948,950,952,954,956,958,960,
        962,964,966,968,970,972,974,976,978,980,982,984,986,988,990,992,994,
        996,998,1000,1002,1004,1006,1008,1010,1012,1014,1016,1018,1020,1022,
        1024,1028,1032,1036,1040,1044,1048,1052,1056,1060,1064,1068,1072,1076,
        1080,1084,1088,1092,1096,1100,1104,1108,1112,1116,1120,1124,1128,1132,
```

```
        1136,1140,1144,1148,1152,1156,1160,1164,1168,1172,1176,1180,1184,1188,
        1192,1196,1200,1204,1208,1212,1216,1220,1224,1228,1232,1236,1240,1244,
        1248,1252,1256,1260,1264,1268,1272,1276,1280,1284,1288,1292,1296,1300,
        1304,1308,1312,1316,1320,1324,1328,1332,1336,1340,1344,1348,1352,1356,
        1360,1364,1368,1372,1376,1380,1384,1388,1392,1396,1400,1404,1408,1412,
        1416,1420,1424,1428,1432,1436,1440,1444,1448,1452,1456,1460,1464,1468,
        1472,1476,1480,1484,1488,1492,1496,1500,1504,1508,1512,1516,1520,1524,
        1528,1532,1536,1540,1544,1548,1552,1556,1560,1564,1568,1572,1576,1580,
        1584,1588,1592,1596,1600,1604,1608,1612,1616,1620,1624,1628,1632,1636,
        1640,1644,1648,1652,1656,1660,1664,1668,1672,1676,1680,1684,1688,1692,
        1696,1700,1704,1708,1712,1716,1720,1724,1728,1732,1736,1743,1751,1760,
        1791,1823,1856,1919,1983,2048};
```

Listing A.4: 6 ヒストグラムに対応したビン二ング

```
/*
 * msp_rebin210130.c
 *
 * Created on: 30 Jan. 2021
 * Author: Takuma Yamaguchi
 *
 */

#include <stdint.h>

#define HISTO_ADDR (0x50030000)
#define HISTO_LEN (24832)
#define HISTO_NUM_BINS_GW ( 2048)
#define HISTO_HDR_NUM_BYTES ( 256)

static unsigned char send_data_payload[HISTO_LEN]="";
unsigned short int *table;

unsigned short int table1[1025] = {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,
        142,143,144,145,146,147,148,149,150,151,152,153,154,155,156,157,158,
        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,
        193,194,195,196,197,198,199,200,201,202,203,204,205,206,207,208,209,
        210,211,212,213,214,215,216,217,218,219,220,221,222,223,224,225,226,
        227,228,229,230,231,232,233,234,235,236,237,238,239,240,241,242,243,
        244,245,246,247,248,249,250,251,252,253,254,255,256,257,258,259,260,
        261,262,263,264,265,266,267,268,269,270,271,272,273,274,275,276,277,
        278,279,280,281,282,283,284,285,286,287,288,289,290,291,292,293,294,
```

```
295,296,297,298,299,300,301,302,303,304,305,306,307,308,309,310,311,
312,313,314,315,316,317,318,319,320,321,322,323,324,325,326,327,328,
329,330,331,332,333,334,335,336,337,338,339,340,341,342,343,344,345,
346,347,348,349,350,351,352,353,354,355,356,357,358,359,360,361,362,
363,364,365,366,367,368,369,370,371,372,373,374,375,376,377,378,379,
380,381,382,383,384,385,386,387,388,389,390,391,392,393,394,395,396,
397,398,399,400,401,402,403,404,405,406,407,408,409,410,411,412,413,
414,415,416,417,418,419,420,421,422,423,424,425,426,427,428,429,430,
431,432,433,434,435,436,437,438,439,440,441,442,443,444,445,446,447,
448,449,450,451,452,453,454,455,456,457,458,459,460,461,462,463,464,
465,466,467,468,469,470,471,472,473,474,475,476,477,478,479,480,481,
482,483,484,485,486,487,488,489,490,491,492,493,494,495,496,497,498,
499,500,501,502,503,504,505,506,507,508,509,510,511,512,514,516,518,
520,522,524,526,528,530,532,534,536,538,540,542,544,546,548,550,552,
554,556,558,560,562,564,566,568,570,572,574,576,578,580,582,584,586,
588,590,592,594,596,598,600,602,604,606,608,610,612,614,616,618,620,
622,624,626,628,630,632,634,636,638,640,642,644,646,648,650,652,654,
656,658,660,662,664,666,668,670,672,674,676,678,680,682,684,686,688,
690,692,694,696,698,700,702,704,706,708,710,712,714,716,718,720,722,
724,726,728,730,732,734,736,738,740,742,744,746,748,750,752,754,756,
758,760,762,764,766,768,770,772,774,776,778,780,782,784,786,788,790,
792,794,796,798,800,802,804,806,808,810,812,814,816,818,820,822,824,
826,828,830,832,834,836,838,840,842,844,846,848,850,852,854,856,858,
860,862,864,866,868,870,872,874,876,878,880,882,884,886,888,890,892,
894,896,898,900,902,904,906,908,910,912,914,916,918,920,922,924,926,
928,930,932,934,936,938,940,942,944,946,948,950,952,954,956,958,960,
962,964,966,968,970,972,974,976,978,980,982,984,986,988,990,992,994,
996,998,1000,1002,1004,1006,1008,1010,1012,1014,1016,1018,1020,1022,
1024,1028,1032,1036,1040,1044,1048,1052,1056,1060,1064,1068,1072,1076,
1080,1084,1088,1092,1096,1100,1104,1108,1112,1116,1120,1124,1128,1132,
1136,1140,1144,1148,1152,1156,1160,1164,1168,1172,1176,1180,1184,1188,
1192,1196,1200,1204,1208,1212,1216,1220,1224,1228,1232,1236,1240,1244,
1248,1252,1256,1260,1264,1268,1272,1276,1280,1284,1288,1292,1296,1300,
1304,1308,1312,1316,1320,1324,1328,1332,1336,1340,1344,1348,1352,1356,
1360,1364,1368,1372,1376,1380,1384,1388,1392,1396,1400,1404,1408,1412,
1416,1420,1424,1428,1432,1436,1440,1444,1448,1452,1456,1460,1464,1468,
1472,1476,1480,1484,1488,1492,1496,1500,1504,1508,1512,1516,1520,1524,
1528,1532,1536,1540,1544,1548,1552,1556,1560,1564,1568,1572,1576,1580,
1584,1588,1592,1596,1600,1604,1608,1612,1616,1620,1624,1628,1632,1636,
1640,1644,1648,1652,1656,1660,1664,1668,1672,1676,1680,1684,1688,1692,
1696,1700,1704,1708,1712,1716,1720,1724,1728,1732,1736,1740,1744,1748,
1752,1756,1760,1764,1768,1772,1776,1780,1784,1788,1792,1796,1800,1804,
1808,1812,1816,1820,1824,1828,1832,1836,1840,1844,1848,1852,1856,1860,
1864,1868,1872,1876,1880,1884,1888,1892,1896,1900,1904,1908,1912,1916,
1920,1924,1928,1932,1936,1940,1944,1948,1952,1956,1960,1964,1968,1972,
1976,1980,1984,1988,1992,1996,2000,2004,2008,2012,2016,2020,2024,2028,
2032,2036,2040,2044,2048};
```

```
unsigned short int table2[129] = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,
        18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,34,36,38,40,42,44,46,48,
        50,52,54,56,58,60,62,64,68,72,76,80,84,88,92,96,100,104,108,112,116,
        120,124,128,136,144,152,160,168,176,184,192,200,208,216,224,232,240,
        248,256,272,288,304,320,336,352,368,384,400,416,432,448,464,480,496,
        512,544,576,608,640,672,704,736,768,800,832,864,896,928,960,992,1024,
        1088,1152,1216,1280,1344,1408,1472,1536,1600,1664,1728,1792,1856,1920,
        1984,2048};

unsigned short int table3[956]={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,
        142,143,144,145,146,147,148,149,150,151,152,153,154,155,156,157,158,
        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,
        193,194,195,196,197,198,199,200,201,202,203,204,205,206,207,208,209,
        210,211,212,213,214,215,216,217,218,219,220,221,222,223,224,225,226,
        227,228,229,230,231,232,233,234,235,236,237,238,239,240,241,242,243,
        244,245,246,247,248,249,250,251,252,253,254,255,256,257,258,259,260,
        261,262,263,264,265,266,267,268,269,270,271,272,273,274,275,276,277,
        278,279,280,281,282,283,284,285,286,287,288,289,290,291,292,293,294,
        295,296,297,298,299,300,301,302,303,304,305,306,307,308,309,310,311,
        312,313,314,315,316,317,318,319,320,321,322,323,324,325,326,327,328,
        329,330,331,332,333,334,335,336,337,338,339,340,341,342,343,344,345,
        346,347,348,349,350,351,352,353,354,355,356,357,358,359,360,361,362,
        363,364,365,366,367,368,369,370,371,372,373,374,375,376,377,378,379,
        380,381,382,383,384,385,386,387,388,389,390,391,392,393,394,395,396,
        397,398,399,400,401,402,403,404,405,406,407,408,409,410,411,412,413,
        414,415,416,417,418,419,420,421,422,423,424,425,426,427,428,429,430,
        431,432,433,434,435,436,437,438,439,440,441,442,443,444,445,446,447,
        448,449,450,451,452,453,454,455,456,457,458,459,460,461,462,463,464,
        465,466,467,468,469,470,471,472,473,474,475,476,477,478,479,480,481,
        482,483,484,485,486,487,488,489,490,491,492,493,494,495,496,497,498,
        499,500,501,502,503,504,505,506,507,508,509,510,511,512,514,516,518,
        520,522,524,526,528,530,532,534,536,538,540,542,544,546,548,550,552,
        554,556,558,560,562,564,566,568,570,572,574,576,578,580,582,584,586,
        588,590,592,594,596,598,600,602,604,606,608,610,612,614,616,618,620,
        622,624,626,628,630,632,634,636,638,640,642,644,646,648,650,652,654,
        656,658,660,662,664,666,668,670,672,674,676,678,680,682,684,686,688,
        690,692,694,696,698,700,702,704,706,708,710,712,714,716,718,720,722,
        724,726,728,730,732,734,736,738,740,742,744,746,748,750,752,754,756,
        758,760,762,764,766,768,770,772,774,776,778,780,782,784,786,788,790,
        792,794,796,798,800,802,804,806,808,810,812,814,816,818,820,822,824,
        826,828,830,832,834,836,838,840,842,844,846,848,850,852,854,856,858,
```

```
131         860,862,864,866,868,870,872,874,876,878,880,882,884,886,888,890,892,
132         894,896,898,900,902,904,906,908,910,912,914,916,918,920,922,924,926,
133         928,930,932,934,936,938,940,942,944,946,948,950,952,954,956,958,960,
134         962,964,966,968,970,972,974,976,978,980,982,984,986,988,990,992,994,
135         996,998,1000,1002,1004,1006,1008,1010,1012,1014,1016,1018,1020,1022,
136         1024,1028,1032,1036,1040,1044,1048,1052,1056,1060,1064,1068,1072,1076,
137         1080,1084,1088,1092,1096,1100,1104,1108,1112,1116,1120,1124,1128,1132,
138         1136,1140,1144,1148,1152,1156,1160,1164,1168,1172,1176,1180,1184,1188,
139         1192,1196,1200,1204,1208,1212,1216,1220,1224,1228,1232,1236,1240,1244,
140         1248,1252,1256,1260,1264,1268,1272,1276,1280,1284,1288,1292,1296,1300,
141         1304,1308,1312,1316,1320,1324,1328,1332,1336,1340,1344,1348,1352,1356,
142         1360,1364,1368,1372,1376,1380,1384,1388,1392,1396,1400,1404,1408,1412,
143         1416,1420,1424,1428,1432,1436,1440,1444,1448,1452,1456,1460,1464,1468,
144         1472,1476,1480,1484,1488,1492,1496,1500,1504,1508,1512,1516,1520,1524,
145         1528,1532,1536,1540,1544,1548,1552,1556,1560,1564,1568,1572,1576,1580,
146         1584,1588,1592,1596,1600,1604,1608,1612,1616,1620,1624,1628,1632,1636,
147         1640,1644,1648,1652,1656,1660,1664,1668,1672,1676,1680,1684,1688,1692,
148         1696,1700,1704,1708,1712,1716,1720,1724,1728,1732,1736,1743,1751,1760,
149         1791,1823,1856,1919,1983,2048};
150
151
152 unsigned long prep_payload_data(uint8_t *bin_config)
153 {
154     unsigned long i, j, k, len;
155     uint16_t num_bins;
156     uint8_t bin_size;
157     uint32_t bin;
158
159         uint32_t *histo_data = (uint32_t *)HISTO_ADDR;
160         /* histogram header into send_data_payload */
161         for (i=0; i<HISTO_HDR_NUM_BYTES/4; i++) {
162                 send_data_payload[i*4 + 1] = histo_data[i] & 0xFF;
163                 send_data_payload[i*4 + 0] = histo_data[i]>>8 & 0xFF;
164                 send_data_payload[i*4 + 3] = histo_data[i]>>16 & 0xFF;
165                 send_data_payload[i*4 + 2] = histo_data[i]>>24 & 0xFF;
166         }
167         len = HISTO_HDR_NUM_BYTES;
168
169         for (i=0; i<6; i++) {
170                 if (bin_config[i] == 0) {
171                 /*
172                  * not re-binning
173                  */
174                    num_bins = HISTO_NUM_BINS_GW;
175                        for (j=0; j<HISTO_NUM_BINS_GW/2; j++) {
176                                send_data_payload[j*4 + 1 + len] = histo_data[j + i*
                                    HISTO_NUM_BINS_GW/2 + HISTO_HDR_NUM_BYTES/4] & 0xFF;
177                                send_data_payload[j*4 + 0 + len] = histo_data[j + i*
```

```
                                HISTO_NUM_BINS_GW/2 + HISTO_HDR_NUM_BYTES/4]>>8 & 0
                                xFF;
178                         send_data_payload[j*4 + 3 + len] = histo_data[j + i*
                                HISTO_NUM_BINS_GW/2 + HISTO_HDR_NUM_BYTES/4]>>16 & 0
                                xFF;
179                         send_data_payload[j*4 + 2 + len] = histo_data[j + i*
                                HISTO_NUM_BINS_GW/2 + HISTO_HDR_NUM_BYTES/4]>>24 & 0
                                xFF;
180                 }
181                 len = len + HISTO_NUM_BINS_GW*2;
182         }
183                 /* Re-binning...*/
184                 else if (bin_config[i]<7) {
185         /* equal interval binning */
186                     num_bins = HISTO_NUM_BINS_GW>>bin_config[i];
187                     bin_size = 1<<bin_config[i];
188                         for (j=0; j<num_bins; j++) {
189                                 bin = 0;
190                                 for (k=j*bin_size/2; k<(j+1)*bin_size/2; k++) {
191                                         bin += (histo_data[k + i*HISTO_NUM_BINS_GW/2 +
                                            HISTO_HDR_NUM_BYTES/4]>>16 & 0xFFFF)
192                                             +(histo_data[k + i*HISTO_NUM_BINS_GW/2 +
                                                  HISTO_HDR_NUM_BYTES/4] & 0xFFFF);
193                                 }
194                                 bin >>= bin_config[i];
195                                 send_data_payload[j*2 + 1 + len] = bin & 0xFF;
196                                 send_data_payload[j*2 + len] = (bin>>8) & 0xFF;
197                         }
198                         len = len + num_bins*2;
199         }
200                 else if (10<bin_config[i] && bin_config[i]<14) {
201                 /* logscale binning */
202                     unsigned char carry_over;
203                     carry_over = 0;
204                         if (bin_config[i] == 11) {
205                                 num_bins = sizeof (table1)/sizeof (table1[0]) - 1;
206                                 table = table1;
207                         }
208                         else if (bin_config[i] == 12) {
209                                 num_bins = sizeof (table2)/sizeof (table2[0]) - 1;
210                                 table = table2;
211                         }
212                         else if (bin_config[i] == 13) {
213                             num_bins = sizeof (table3)/sizeof (table3[0])-1;
214                             table = table3;
215                         }
216                         for (j=0; j<num_bins; j++) {
217                             bin_size = *(table+j+1) - *(table+j);
```

```
218                         if (carry_over == 0 && bin_size%2 == 0) {
219                                 carry_over = 0;
220                                 for (k=*(table+j)/2; k<*(table+j+1)/2; k++) {
221                                     bin = bin + (histo_data[k + i*
                                            HISTO_NUM_BINS_GW/2 + HISTO_HDR_NUM_BYTES
                                            /4]>>16 & 0xFFFF)
222                         + (histo_data[k + i*HISTO_NUM_BINS_GW/2 +
                             HISTO_HDR_NUM_BYTES/4] & 0xFFFF);
223                         }
224                                 bin = bin/bin_size;
225                         }
226                         else if (carry_over == 0 && bin_size%2 == 1) {
227                                 carry_over = 1;
228                                 if (bin_size != 1) {
229                                         for (k=*(table+j)/2; k<(*(table+j+1)-1)
                                                /2; k++) {
230                                             bin = bin + (histo_data[k + i*
                                                 HISTO_NUM_BINS_GW/2 +
                                                 HISTO_HDR_NUM_BYTES/4]>>16 & 0
                                                 xFFFF)
231                             + (histo_data[k + i*HISTO_NUM_BINS_GW/2 +
                                 HISTO_HDR_NUM_BYTES/4] & 0xFFFF);
232
233                                 }
234                                         bin = bin + (histo_data[k + i*
                                             HISTO_NUM_BINS_GW/2 +
                                             HISTO_HDR_NUM_BYTES/4] & 0xFFFF);
235
236                                         bin = bin/bin_size;
237                                 }
238                                 else if (bin_size == 1) {
239                                         k = *(table+j)/2;
240                                         bin = histo_data[k + i*HISTO_NUM_BINS_GW
                                             /2 + HISTO_HDR_NUM_BYTES/4] & 0xFFFF
                                             ;
241                                 }
242                         }
243                         else if (carry_over == 1 && bin_size%2 == 0) {
244                                 carry_over = 1;
245                                 k = (*(table+j)-1)/2;
246                                 bin = histo_data[k + i*HISTO_NUM_BINS_GW/2 +
                                     HISTO_HDR_NUM_BYTES/4]>>16 & 0xFFFF;
247                                 for (k=(*(table+j)+1)/2; k<(*(table+j+1)-1)/2;
                                      k++) {
248                                     bin = bin + (histo_data[k + i*
                                         HISTO_NUM_BINS_GW/2 + HISTO_HDR_NUM_BYTES
                                         /4]>>16 & 0xFFFF)
249                         + (histo_data[k + i*HISTO_NUM_BINS_GW/2 +
```

```
                                HISTO_HDR_NUM_BYTES/4] & 0xFFFF);
250                             }
251                             bin = bin + (histo_data[k + i*HISTO_NUM_BINS_GW
                                    /2 + HISTO_HDR_NUM_BYTES/4] & 0xFFFF);
252                             bin = bin/bin_size;
253                         }
254                         else if (carry_over == 1 && bin_size%2 == 1) {
255                             carry_over = 0;
256                             k = (*(table+j)-1)/2;
257                             bin = histo_data[k + i*HISTO_NUM_BINS_GW/2 +
                                    HISTO_HDR_NUM_BYTES/4]>>16 & 0xFFFF;
258                             if (bin_size != 1) {
259                                 for (k=(*(table+j)+1)/2; k<*(table+j+1)
                                        /2; k++) {
260                                     bin = bin + (histo_data[k + i*
                                            HISTO_NUM_BINS_GW/2 +
                                            HISTO_HDR_NUM_BYTES/4]>>16 & 0
                                            xFFFF)
261                                 + (histo_data[k + i*HISTO_NUM_BINS_GW/2 +
                                        HISTO_HDR_NUM_BYTES/4] & 0xFFFF);
262                                 }
263                                 bin = bin/bin_size;
264                             }
265                         }
266                         send_data_payload[j*2 + 1 + len] = bin & 0xFF;
267                         send_data_payload[j*2 + len] = bin>>8 & 0xFF;
268                         bin = 0;
269                     }
270                     len = len + num_bins*2;
271                 }
272         }
273         return len;
```

謝辞

本卒業研究において、高エネルギー宇宙・可視赤外天文学研究室の先生方、先輩方及び同期の仲間たちには多くの手助けをいただきました。誠にありがとうございました。

特に主査の高橋先生には、ソフトウェア開発について多くの知識や技術を教えていただきました。それに加え、海外チームと研究開発を行うおもしろさにも、気付かせていただきました。本当に感謝しています。

またスウェーデンチームの Thedi には、毎週の会議でソフトウェア開発のアドバイスを多くいただきました。まだまだ経験の浅い私にもわかりやすい説明で、慣れない英語での報告もサポートしていただけたことに感謝しています。

最終的にデータ圧縮機能を実装完了できたのは、この他にもいろんな支えがあったからだと強く思います。本研究で学んだ知識や技術を今後の研究活動に生かして、精進していきたいと思います。

参考文献

[1] *Miniature Student Satellite A project overview*.

[2] Royal InstituteTechnology（スウェーデン王立工科大学）. Studentsatelliten mist. https://www.kth.se/sci/centra/rymdcenter/studentsatellit/studentsatelliten-mist-1.481707.

[3] Marcus Persson. Software development and quali・cation testing of a cubesat x-ray monitor, 2019.

[4] Theodor Stana. *CUBES Instrument Manual*.

[5] Wikipedia. セシウム 137, 2011. https://ja.wikipedia.org/wiki/%E3%82%BB%E3%82%B7%E3%82%A6%E3%83%A0137.

[6] 小山勝二・嶺重慎. シリーズ現代の天文学第 8 巻 ブラックホールと高エネルギー現象. 日本評論社, 2007.

[7] 気象庁地磁気観測所. 地磁気の基礎知識. https://www.kakioka-jma.go.jp/knowledge/mg_bg.html.

[8] 浜松ホトニクス. Mppc とは. https://www.hamamatsu.com/jp/ja/product/optical-sensors/mppc/what_is_mppc/index.html.

[9] 佐々木慎一 高エネルギー加速器研究機構放射線科学センター. 放射線計測基礎論, 2011.