修士学位論文

中間赤外線イメージスライサ搭載型分光器 MIRSIS の検出器系開発

2009 年度
(平成 21 年度)

茨城大学大学院理工学研究科

自然機能科学専攻

08NM224G
土佐 政宗

ABSTRACT

中間赤外線イメージスライサ搭載型分光器 MIRSIS(Mid-InfraRed Spectrometer with an Image Slicer) の開発を進めている。MIRSIS には、撮像モードと分光モードの 2 つの観測モードがある。撮像モードでは Boeing 社製 Si:As BIB 128×128 array 検出器、分光モードでは Rockwell 社製 Si:As IBC 320×240 array 検出器を使用する。Si:As 検出器はスペクトル感度が 2-28μm にある半導体検出器で、MIRSIS で行う 8∼13.5μm 帯の観測に適している。

赤外線観測装置の多くは量子型検出器を用いる。この検出器は半導体を用いて、内部光電効果によって発生する伝導電子を測定することで光を検出する。また通常、検出器は格子状に並べて使用する。検出器を格子状に並べた構造を array 構造と呼ぶ。array 型検出器で得たデータは、マルチプレクサと呼ばれる検出器読み出し回路によって pixel(画素) が読み出される。マルチプレクサでは、pixel データを読み出すタイミングを決めるためのクロック電圧と、マルチプレクサに使用されている各電子部品 (MOSFET やオペアンプ) を駆動するためのバイアス電圧が必要となる。これらの電圧を制御するためのハードウェアとして、マルチプレクサと検出器を静電気などの突発的な大電流から守る FOB、バイアス電圧を生成する DRV ボード、検出器で得られた電圧を Analog Digital(AD) 変換するための ADC ボード、検出器制御用 PC で指定したクロックを ADC ボードに転送する DIO ボード等を組合わせて使用する。またソフトウェアとして、MIRSIS では東京大学天文学教育センターが開発した TAC(TAO Array Controller) システムを使用する。このソフトウェアをリアルタイム LinuxOS にインストールして使用することで、時間に厳しい制約のあるクロック電圧のリアルタイム制御が可能となる。検出器制御用 PC の TAC システムソフトウェアで、バイアス電圧やクロックを指定することができる。

検出器を立上げるためには、「1.ADC、DRV ボードの駆動、特性測定試験」、「2.TAC システムを用いたクロック出力試験」、「3.Function Generator、TAC システムを用いた画像取得試験」、「4. マルチプレクサの駆動試験」、「5. 検出器の駆動試験」の順で試験を行う必要がある。これらを MIRSIS 撮像・分光検出器の両方について行う必要がある。まず、MIRSIS 撮像検出器の立上げを行った。検出器システムに初めて触れることになったため、手始めにその時点ですぐにアクセスできた MIRSIS 撮像検出器と同じ pixel 数の Si:Sb 128×128 array マルチプレクサを用いて、TAC システムと各ハードウェアを用いた検出器システムの駆動試験を行った。その結果、本システムを使用し、マルチプレクサからの読み出し出力の取得を行って、fits 形式の画像ファイルを得ることができた。そこで次に、同様のシステムを用いて MIRSIS 撮像検出器の駆動試験を行った。入射光を遮断したダーク画像と赤外線光源を光らせたときの画像を比較することで、駆動を確かめた。その結果、有意な違いが得られたので撮像検出器の駆動試験に成功したといえる。また、積分時間と検出器温度によっても検出器出力が変化すること、いくつかの dead pixel が発生することが判明した。

MIRSIS 撮像検出器の立上げに成功したので、次に MIRSIS 分光検出器の立上げを行った。MIRSIS 分光検出器は MIRSIS 撮像検出器とマルチプレクサの構造が大きく異なる。故にハードウェア、ソフトウェアの両方で撮像検出器の TAC システムをもとに様々な改善が必要となる。分光検出器を立上げるため、分光検出器用 ADC ボードの特性測定試験と、分光検出器用 DRV ボードの駆動試験を行った。その結果、ADC ボードでは直流、交流電圧について理論値に近い特性を測定することができた。また、DRV ボードは TAC システムを用いた駆動試験に成功した。分光検出器に関しては今後、TAC システムの設定をいくつか変更し、Si:As 320×240 array マルチプレクサ、検出器の駆動試験を行っていく予定である。更に分光検出器の立上げが成功後、撮像、分光両検出器とも MIRSIS に搭載して、性能評価と最適化を行う必要がある。

目次

1 赤外線観測装置における検出器 1

1.1 赤外線観測 . . . 1
1.2 中間赤外線分光観測装置 . . . 2
1.3 MIRSIS とイメージスライサ . . . 3
1.4 SN と感度 . . . 4
1.5 量子型検出器 . . . 5
1.5.1 半導体の構造 . . . 5
1.5.2 フォトコンダクター . . . 5
1.5.3 フォトダイオード . . . 6
1.6 赤外線観測で使用する検出器 . . . 7
1.6.1 フォトコンダクター型検出器とフォトダイオード型検出器 . . . 7
1.6.2 大規模 2 次元アレイ検出器 . . . 8
1.6.3 MOSFET . . . 9
1.6.4 ソースフォロワ回路 . . . 11

2 検出器の動作原理とデータ読み出しシステム構成 12

2.1 MIRSIS に用いる検出器 . . . 12
2.2 マルチプレクサの動作原理とクロック . . . 13
2.2.1 マルチプレクサの動作原理 . . . 13
2.2.2 クロック . . . 17
2.2.3 バイアス電圧 . . . 18
2.3 検出器の駆動およびデータ読み出し回路群 . . . 19
2.3.1 FOB . . . 19
2.3.2 DRV ボード . . . 20
2.3.3 ADC ボード . . . 21
2.3.4 LVDS ボード . . . 23
2.3.5 DIO ボード . . . 23
2.3.6 検出器の駆動およびデータ読み出し回路群の接続 . . . 24
2.4 検出器の駆動およびデータ取得システム . . . 26
2.4.1 検出器の駆動システム . . . 26
2.4.2 検出器のデータ取得システム . . . 26

3 検出器制御ソフトウェア 28

3.1 検出器制御ソフトウェア TAC システムの概要 . . . 28
3.2 デバイスドライバ (dev ディレクトリ) . . . 29
3.3 クロック関係 (clk ディレクトリ) . . . 30
3.3.1 cpc . . . 30
3.3.2 clkgen . . . 41
3.4 DIO ボードの制御 (dio ディレクトリ) . . . 43
3.5 BIAS 制御 (bis ディレクトリ) . . . 43
3.5.1 ハードウェアによるバイアス値の制限 . . . 43
3.5.2 ソフトウェアによる設定 . . . 44
3.5.3 バイアス出力の制御と管理 . . . 45
3.6 ステータスファイル管理 . . . 45

3.7 オペレーション 45

4 MIRSIS **撮像検出器** Si:As 128×128 array **検出器駆動試験** 48
4.1 Si:Sb 128×128 マルチプレクサ駆動試験 48
4.1.1 試験目的 48
4.1.2 試験環境 48
4.1.3 試験方法 50
4.1.4 実験結果 51
4.2 Si:As 128×128 array 検出器駆動試験 56
4.2.1 試験目的 56
4.2.2 試験環境 56
4.2.3 試験方法 57
4.2.4 実験結果 58

5 MIRSIS **分光検出器** Si:As 320×240 array **検出器の立上げ** 61
5.1 ADC ボードの特性測定試験 61
5.1.1 ADC ボードの AD 変換回路群概要 62
5.1.2 ADC ボードの特性測定試験 1(アンプ回路の動作確認) 66
5.1.3 ADC ボードの特性測定試験 (フィルター回路の動作確認) 68
5.2 TAC システムを用いた DRV ボードの駆動試験 69
5.2.1 TAC システムの変更点 70
5.2.2 DRV ボード駆動試験 70

6 **まとめと今後の展望** 72
6.1 まとめ 72
6.2 今後の展望 72

図目次

1 大気の窓 (http://www.iac.es/project/CCam より引用) . . . 1
2 分光観測における観測装置 . . . 2
3 イメージスライサの原理 . . . 3
4 半導体のバンド構造 (家 正則・岩室史英・舞原俊憲・水本好彦・吉田道利編、「宇宙の観測 I －光 · 赤外線天文学」、日本評論社、2007、図 6.2 より引用) . . . 5
5 n 型半導体と p 型半導体 (家 他、2007、図 6.3 より引用) . . . 6
6 フォトダイオード (家 他、2007、図 6.4 より引用) . . . 7
7 BIB 構造 . . . 8
8 2 次元アレイ検出器 (家 他、2007、図 6.8 より引用) . . . 8
9 MOSFET の回路図記号 . . . 9
10 MOSFET の断面図 . . . 10
11 NMOS の $I_{\rm D}$-$V_{\rm G}$ 特性 . . . 10
12 NMOS の $I_{\rm D}$-$V_{\rm D}$ 特性 . . . 10
13 ソースフォロワ回路 . . . 11
14 ソースフォロワ回路における $V_{\rm in}$ と $V_{\rm out}$ の関係 . . . 11
15 MIRSIS の検出器系システムの模式図 . . . 12
16 Si:As 128x128 array 読み出し回路図 (Device Description and Operation Guide for the DRS Technologies MF-128 BIB Hybrid Focal Plane Array p13 改) . . . 14
17 光子によって発生した光電流の流れ: 検出器の感光部である「Detector」の部分でバンドギャップ以上のエネルギーを持った光子を受光すると電流が発生し、この電流は $C_{\rm int}$ へ蓄積される . . . 15
18 リセットが起きる仕組み . . . 15
19 Si:As 128×128 array の output と pixel の関係: Si:As 128×128 array は 4 つの output を持っている。そこで、この output によって一度に読み出せる 4pixels を 1Block とし、各 pixel を 1∼4096 Block まで分割している . . . 15
20 single-sampling mode で読み出す電圧 . . . 16
21 double-sampling mode で読み出す電圧 . . . 16
22 128×128 pixel へアクセスを行うクロックの例 (FSYNC の high/low の割合は可変で、128×128 pixel へアクセスを行う時間に対する FSYNC が high の割合を変化させることで、リセット電圧を与える時間を変化させる) . . . 17
23 Si:As 128×128 array 検出器用 FOB の写真 . . . 19
24 Si:As 128×128 array 検出器および Si:Sb 128×128 array 検出器用 DRV ボードの写真 . . . 20
25 1 個のバイアス電圧に注目した際の DRV ボードの動作を表現したブロック図 . . . 20
26 ジャンパピンの写真 (左:FOB を使用する場合、右:FOB を使用しない場合) . . . 21
27 Si:As 128×128 array 検出器および Si:Sb 128×128 array 検出器用 ADC ボードの写真 . . . 21
28 ADC ボードの動作を表現したブロック図 . . . 22
29 ADC ボードのディップスイッチ (写真は 4 チャンネル使用する場合) . . . 22
30 LVDS ボードの写真 . . . 23
31 DIO ボードの写真 . . . 23
32 DIO ボードの 4 つのジャンパ端子 . . . 23
33 検出器制御用 PC の PCI バスに接続された DIO ボード . . . 24
34 エレキボックスに接続された DRV ボードと ADC ボード . . . 24
35 検出器箱内で検出器を固定している FOB . . . 24
36 検出器の駆動およびデータ読み出し回路群の接続 . . . 25
37 MIRSIS 検出器システムの概観 . . . 27
38 TAC システムのコンセプト . . . 28

39 TAC システムのディレクトリ構造 . . . 29
40 sudo コマンドの設定 . . . 29
41 各モジュールのインストール/リムーブの端末での流れ . . . 30
42 CLP クロックメモリ領域の 2 バンク構造 . . . 33
43 クロックソースファイルの例 (ページ 1) . . . 34
44 クロックソースファイルの例 (ページ 2) . . . 35
45 クロックソースファイルの例 (ページ 3) . . . 36
46 クロックソースファイルの例 (ページ 4) . . . 37
47 クロックソースファイルの例 (ページ 5) . . . 38
48 クロックソースファイルの例 (ページ 6) . . . 39
49 クロックソースファイルの例 (ページ 7) . . . 40
50 ビットデータとクロックパターンの関係 (図 4649∼61 行目の例の場合) . . . 41
51 クロックパラメータが BPP:100 PP:100 ND:1 NEx:10 Wip:0 ChL:1 ChB:1 CoAdd:0 Chop_Mode:0 Clk_Typ:0 SRow:1 SCol:1 WRow:128 WCol:128 で/home/mirsis/data/clock にクロックソースファイル、クロックバイナリファイルを保存する場合の clkgen コマンドの使用例 . . . 42
52 バイアス電圧生成回路 . . . 44
53 バイアス電圧の DDAC 値を記述する header ファイルの一部 . . . 44
54 検出器の起動手順 1 . . . 46
55 検出器の起動手順 2 . . . 47
56 フレームデータの取得 . . . 47
57 検出器の終了手順 . . . 47
58 マルチプレクサ駆動試験時の配線概略図 . . . 49
59 プリント基板ケーブル (左:銅、右:マンガニン) . . . 49
60 サーマルアンカ基板 . . . 49
61 ADC ボード上でクロックと Analog データを確認するための測定点 . . . 50
62 マルチプレクサの暗電流による出力 (常温、検出器箱の蓋は閉じているが、真空容器の蓋は開いている状態) 51
63 マルチプレクサに可視光を当てたときの出力 (常温、検出器箱と真空容器の蓋が開いて光が入っている状態) . . . 51
64 マルチプレクサの暗電流による出力 (常温、検出器箱と真空容器の蓋が閉じている状態) . . . 51
65 マルチプレクサの暗電流による出力 (低温、検出器箱と真空容器の蓋が閉じている状態) . . . 51
66 オフセット付き非反転増幅器を用いた出力端子 1 の出力増幅回路 . . . 52
67 1frame の Analog データ . . . 53
68 1frame の Analog データ (光を入れた時) . . . 53
69 1line の Analog データ . . . 54
70 1line の Analog データ (光を入れた時) . . . 54
71 1pixel の Analog データ . . . 55
72 1pixel の Analog データ (光を入れた時) . . . 55
73 検出器駆動試験時の配線概略図 . . . 56
74 検出器箱と光源の位置関係 . . . 57
75 光源の周りにアルミ板とアルミテープをも用いて実際に遮光を施した撮像検出器箱の写真 . . . 57
76 撮像検出器の駆動に成功し、初めて取得した画像 (クロックタイプ 3 を使用) . . . 58
77 光源が ON の状態 . . . 59
78 光源が OFF の状態 . . . 59
79 各温度、積分時間での暗電流による出力 (全て同じ表示レンジ) . . . 60
80 ADC ボードの検出器出力の入力から AD 変換器までの回路 . . . 61
81 Si:As 320×240 array 検出器用 ADC ボード (17 チャンネル分の AD 変換回路群が実装されているが、実際には 16 チャンネル分のみ使用する) . . . 62

82 プリアンプ回路 63
83 ADC ボード上の可変抵抗 63
84 差動アンプ回路 64
85 2 次ベッセルフィルター回路 65
86 16bit AD 変換器の動作 65
87 ADC ボードの特性測定試験状況 (アンプ回路の動作確認) 66
88 V_{DF}^{-} 電圧の比較 67
89 V_{DF}^{+} 電圧の比較 67
90 ADC ボードの特性測定試験状況 (フィルター回路の動作確認) 68
91 Gain の比較 69
92 Si:As 320×240 array 検出器用 DRV ボード 69
93 Si:As 320×240 array 検出器用バイアス電圧の設定をするための header ファイル 70
94 Si:As 320×240 array 検出器用 DRV ボードの駆動試験を行った際の端末のキャプチャ 71
95 Si:As 320×240 array 検出器用 DRV ボードの電圧出力地点 71

表目次

1 MIRSIS のおもな仕様 3
2 各検出器の主な仕様 (各検出器マニュアルより) 12
3 MIRSIS 撮像検出器で使用するバイアス電圧 19
4 MIRSIS 撮像検出器で使用するバイアス電圧 19
5 クロックの種類と内容 27
6 dev ディレクトリ内に存在する主なファイルとその内容 29
7 clk ディレクトリ内に存在する主なファイルとその内容 30
8 clkgen プログラムで指定するパラメータと内容 42
9 dio ディレクトリ内に存在する主なファイルとその内容 43

1　赤外線観測装置における検出器

1.1　赤外線観測

赤外線はおよそ 1〜1000μm の波長域での電磁波であり、1〜3μm の近赤外線、3〜30μm の中間赤外線、30〜200μm の遠赤外線に分類することができる。この帯域は黒体輻射におけるウィーンの変位則より温度にして、15〜3000K 程度の熱放射領域に対応する。また、赤外線の光子はエネルギー的にはおよそ 1meV〜1eV に対応し、これらの光子は高い準位にある原子の再結合線やさまざまな分子の振動、回転線に関わった現象を調べるのに有効である。一方で、可視光より長波長に位置する赤外線は、星間塵などの固体微粒子による散乱、吸収が可視光に比べて著しく減少し、星間空間における減光が大幅に軽減される。その結果、可視光では見ることのできない星形成領域の内部などを観測できる。

しかし、これらの観測にはいくつかの難点があり、その一つが地球大気による吸収の問題である。地上からの観測の場合、大気中に含まれる、水、二酸化炭素、一酸化炭素、メタンなどの微量成分が赤外線の強い吸収体であるため、観測できる赤外線は一部に限られる。地上から観測できる赤外線は、「大気の窓」と呼ばれているところどころに開かれた透過率のよい波長域に限定されている。図 1 はマウナケアにおける中間赤外域の大気の窓で、横軸に波長 (μm)、縦軸に透過率をとったグラフである。赤外線観測におけるもう一つの難点は、観測装置や周囲からの熱放射がとても強いことである。装置や大気の温度は 200~300K に分布し、この温度領域からの熱放射は赤外線領域に対応している。よって、このバックグランド放射を抑えるために装置の極低温冷却が必要になっている。

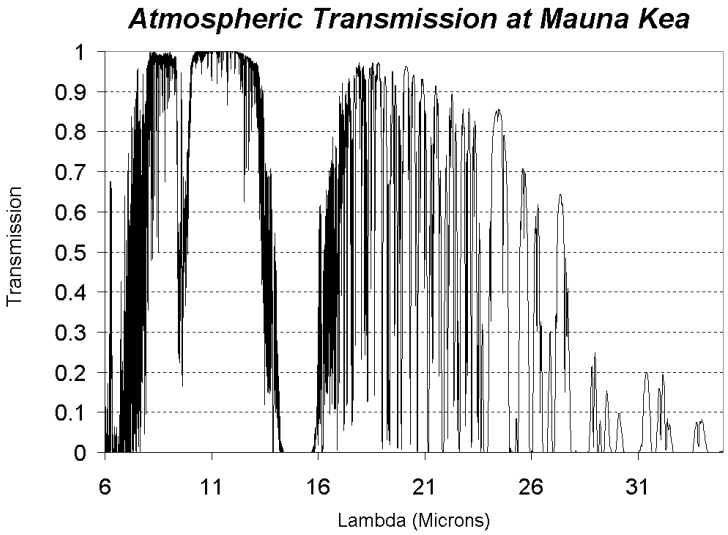

図 1　大気の窓 (http://www.iac.es/project/CCam より引用)

星間空間における透過率では電波領域のほうが赤外線領域より優れているが、観測の際に考えるべき重要な点として空間分解能があげられる。空間分解能は以下の式で与えられる。

$$(\text{空間分解能}) = 1.22 \times \frac{\lambda}{D} \quad [\text{rad}] \tag{1}$$

式 (1) において λ は波長 (m)、D は望遠鏡の口径 (m) である。この式から分かるように波長が長くなるほど、空間分解能は悪くなっていく。よって、同じ口径の望遠鏡では電波に比べ、赤外線の空間分解能が優れることになる。また式 (1) より、望遠鏡の口径が大きくなるほど、空間分解能が良くなっていく。そこで近年、口径 8.2m のすばる望遠鏡を始めとした望遠鏡の大型化が進められてきて、現在では TMT(Thirty Meter Telescope) 計画における口径が 30m の超大型地上望遠鏡の開発が始まっている。

1.2　中間赤外線分光観測装置

この節では、一般的な中間赤外線分光観測装置について述べていく。図 2 はスリット分光器を用いた観測装置の例である。装置は大きく分けて、スリットを挟んで前光学系と分光系が置かれた構造になっている。また前節でも述べたとおり赤外線観測では装置自体が熱放射を行うので、中間赤外線観測装置の内部は極低温に冷却する。まず常温の外壁から光学系への熱輻射を防ぐため、ラディエーションシールドと呼ばれる薄い金属の板で光学系を覆う。更に冷凍機によって、ラディエーションシールドは 50K 程度に、光学系は <30K 程度に、検出器は <10K 程度まで冷却する。また、装置内部が空気にみたされていると装置外部と内部の間で気体分子による熱流入が起こるため、装置内部は 10^{-6}Torr 程度の真空状態に保たれている。このようにして、中間赤外線観測装置内部を冷却することで装置の放射による熱雑音を取り除く。

天体からの光は望遠鏡の主鏡に集められ、副鏡によって反射された光は、まず前光学系へと入射する。ここではコリメータレンズによって光を平行光にし、cold stop によって副鏡を通らなかった迷光や副鏡を支えているスパイダーからの熱放射を除去する。この平行光はカメラレンズによってスリット上で一度結像し、分光系へと入射する。このスリットにおいて、観測対象のどの部分を分光するかを決めている。つまり、スリットの開口を通過した光のみが分光されることになる。スリットを通過した光は、ここでもコリメータレンズによって光を平行光にされた後、分光素子によって分光される。この光がカメラレンズによって結像された像が、検出器面へ照射される。

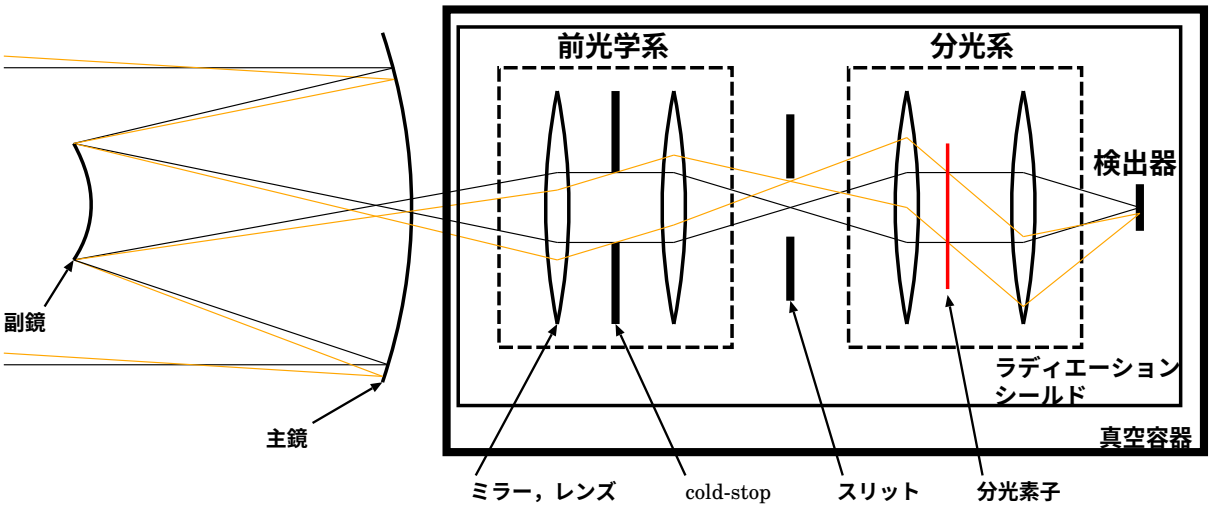

図 2　分光観測における観測装置

1.3 MIRSIS とイメージスライサ

MIRSIS(Mid-InfraRed Spectrometer with an Image Slicer) は次世代望遠鏡で有効と期待されるイメージスライサ型の面分光器を載せたプロトタイプの装置であり、現在、茨城大学、宇宙航空研究開発機構、国立天文台、東京大学で共同開発が行われている。真空容器の大きさは約 65cm×51cm×32cm とコンパクトな設計で、8〜13μm の中間赤外線の撮像と分光を予定している。表 1 に MIRSIS の撮像、分光における仕様をのせる。

イメージスライサ型分光器は図 3 に示す機構で分光を行う。通常の分光器と異なる点は図 2 のスリットの替わりにイメージスライサを配置する点である。イメージスライサではスライシングミラーと呼ばれるわずかに角度が異なった反射型スリットの集合体を図 2 のスリットの位置に設置することで、天体の光をミラー面の傾きにより異なる方向へ反射させる。この分割された光はヒトミミラーによって擬スリットミラー上で再び結像する。ヒトミミラーはスライシングミラーから擬スリットミラーまでの光路でヒトミとなる場所に置かれ、スライシングミラーで分割された光を一列に並べ直して擬スリットミラー上で結像させる役割を持つ。擬スリットミラーはヒトミミラーからの光の方向を揃えて分光器に入射させるための反射平面鏡の集合体で、この後ろに続く分光器からはあたかもスリットであるかのように見えるのでこのような名称で呼ばれている。このイメージスライサは狭い視野に固まっている天体の面分光を可能にする。スライシングミラーを用いることで一度に天体の波長分散の観測を行えるので、観測効率の向上、点源の分光観測におけるスリットのせの労力の改善などが期待できる。

全般	観測波長	8〜13.5μm
	真空容器サイズ	65cm × 51cm × 32cm
	光学系サイズ	57cm × 43cm × 29cm
	冷却	4KGM 冷凍機 (SHI SRDK408D 型)
	検出器温度	<8K
分光モード	ピクセルスケール	0.45”/pix@IRTF、0.2”/pix@Subaru
	視野	22”× 4.5”@IRTF
	波長分解能	R = $\frac{\lambda}{\Delta\lambda}$ ∼200
	検出器	Rockwell 社製 Si:As 320×240 array
	最終 F 値	7.7
撮像モード	ピクセルスケール	0.31”/pix@IRTF、0.14”/pix@Subaru
	視野	40”× 40”@IRTF
	波長分解能	R∼10
	検出器	Boeing 社製 Si:As 128×128 array
	最終 F 値	16.5

表 1　MIRSIS のおもな仕様

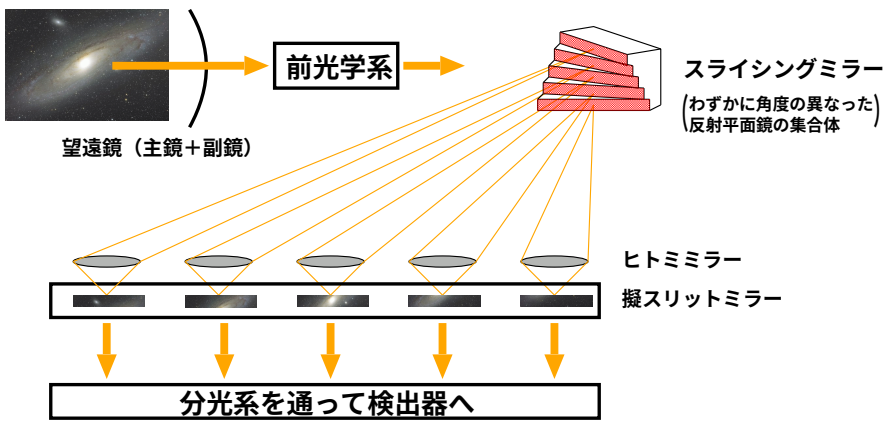

図 3　イメージスライサの原理

1.4 SN と感度

信号とノイズの比を用いた SN(Signal/Noise) は信号検出の質を表す指標としてよく用いられる。量子型検出器を用いる場合、光子を半導体内で自由電子に変換して検出する (詳細は後述)。この稀な確率でやってくる光子を自由電子に変換する検出過程を大量に観測するとき、この検出の確率分布はポアソン分布に依存する。ポアソン分布におけるノイズは、信号平均値の平方根で表すことができる。故に、SN は以下の式で表される。

$$\mathrm{SN} = \frac{\text{信号強度}}{\text{ノイズ強度}} = \frac{i_\mathrm{S} \cdot t}{\sqrt{(i_\mathrm{S} + i_\mathrm{BG} + i_\mathrm{DC})t + M \cdot n_\mathrm{R}^2}} \tag{2}$$

式 (2) 中で使用されている変数は以下の式で表される。

- i_S:天体からの光で発生する電子数 (electrons/sec)
  天体は点源天体と拡散源天体の 2 種類に分別することができる。それぞれの場合での天体からの光で発生する電子数は以下の式で表される。

$$\begin{aligned} i_{\mathrm{S(点源)}} &= \int_{\lambda_1}^{\lambda_2} \frac{F_\lambda}{hc/\lambda} d\lambda \cdot \tau_\mathrm{A}^\lambda \cdot S \cdot \eta \\ i_{\mathrm{S(拡散源)}} &= \int_{\lambda_1}^{\lambda_2} \frac{I_\lambda \cdot \Omega}{hc/\lambda} d\lambda \cdot \tau_\mathrm{A}^\lambda \cdot S \cdot \eta \end{aligned} \tag{3}$$

$$\left\{ \begin{array}{l} \lambda\text{: 観測波長 (m)} \\ \lambda_1 \sim \lambda_2\text{: 観測波長帯 (m)} \\ F_\lambda\text{: 点源のフラックス } (\mathrm{W \cdot m^{-2} \cdot m^{-1}}) \\ I_\lambda\text{: 拡散源の輝度 } (\mathrm{W \cdot m^{-2} \cdot m^{-1} \cdot str^{-1}}) \\ \Omega\text{: 考えている画素数が見る立体角 (str)} \\ h\text{: プランク定数 } (\mathrm{J \cdot sec}) \\ c\text{: 光速 } (\mathrm{m \cdot sec^{-1}}) \\ \tau_\mathrm{A}^\lambda\text{: 大気、望遠鏡、装置を含む総合的な透過率} \\ S\text{: 望遠鏡の有効面積 } (\mathrm{m^2}) \\ \eta\text{: 検出器の量子効率} \end{array} \right.$$

- i_BG:背景光で生じる電子数 (electrons/sec)
  大気、望遠鏡、観測装置からの熱放射を合わせて背景光と呼ぶ。大気による放射が最も支配的な場合、背景光で生じる電子数は以下の式で表される。

$$i_\mathrm{BG} = \int_{\lambda_1}^{\lambda_2} \frac{I_\lambda^\mathrm{BG} \cdot \Omega}{hc/\lambda} d\lambda \cdot \tau_\mathrm{B}^\lambda \cdot S \cdot \eta \tag{4}$$

$$\left\{ \begin{array}{l} I_\lambda^\mathrm{BG}\text{: 大気による背景放射の輝度 } (\mathrm{W \cdot m^{-2} \cdot m^{-1} \cdot str^{-1}}) \\ \tau_\mathrm{B}^\lambda\text{: 望遠鏡、装置を含む総合的な透過率} \\ \text{(他の変数は式 (3) と同様)} \end{array} \right.$$

- i_DC:暗電流 (electrons/sec)
  検出器に光を入射させない状態でも発生する電子を暗電流と呼ぶ。半導体検出器内で価電子帯の電子が熱的に励起され、自由電子になることで発生する (詳細は後述)。半導体内エネルギーギャップを熱的に越えることが支配的な原因の場合は、暗電流はボルツマン分布に従うので以下の式で表される。

$$i_\mathrm{DC} \propto \exp\left(-\frac{\Delta E}{kT}\right) \tag{5}$$

$$\left\{ \begin{array}{l} \Delta E\text{: 半導体のエネルギーギャップ (J)} \\ k\text{: ボルツマン定数 } \mathrm{J \cdot K^{-1}} \\ T \text{ 半導体検出器の温度 (T)} \end{array} \right.$$

- n_R:読み出しノイズ (electrons/sec)
  検出器で発生した電子を読み出す際、検出回路内で発生するノイズで 1pixel あたりの値。積分時間に依存しない。

- M:読み出し回数

- t:積分時間 (sec)

SN の値が大きい程、感度が良い観測といえる。式 (2) より、SN を上げるためには $i_{\rm S}$、t を大きくし $i_{\rm BG}$、$i_{\rm DC}$、M、$n_{\rm R}$ を小さくすればよい。しかし $i_{\rm S}$ は観測する天体によって決まるので変化させることができない。よって、他のパラメータを変化させることが重要となる。$i_{\rm DC}$ は式 (5) より検出器を冷却することで減少させることが可能である。また $n_{\rm R}$ は性能の良い検出回路を製作することで減少させることができる。しかし $i_{\rm BG}$ を減少させるには大気や大型の望遠鏡が関わってくるので限界がある。ノイズが $i_{\rm BG}$ に支配される観測を背景限界での観測といい、この場合の SN は以下の式で表される。

$$SN = \frac{i_{\rm S} \cdot t}{\sqrt{i_{\rm BG} \cdot t}} = \frac{i_{\rm S}}{\sqrt{i_{\rm BG}}}\sqrt{t} \tag{6}$$

式 (6) より、SN は天体の明るさ $i_{\rm S}$ と積分時間の平方根 $\sqrt{t}$ に比例することが分かる。また点源天体の場合、$i_{\rm S}$ は式 (3) より望遠鏡の有効面積に比例するので、SN を上げるためには望遠鏡の口径の巨大化と積分時間の増加が必要となる。

1.5 量子型検出器

1.5.1 半導体の構造

赤外線観測装置の多くは量子型検出器を用いる。この検出器は半導体を用いて、内部光電効果によってできる伝導電子を測定する。固体中の電子の状態は、価電子が占有するエネルギー領域の価電子帯、電子が存在できない領域の禁制帯、遷移電子により電気伝導が生じる領域の伝導帯、の 3 つの帯 (バンド) から成り立っている。半導体はフェルミレベルがバンドギャップ中にあるが、価電子帯から熱的に励起された電子が伝導帯にもある程度存在することができるため、電気伝導度を持つ物質である (図 4)。この半導体を用いた量子型検出器は、次にあげるフォトコンダクターとフォトダイオードの 2 つのタイプがある。

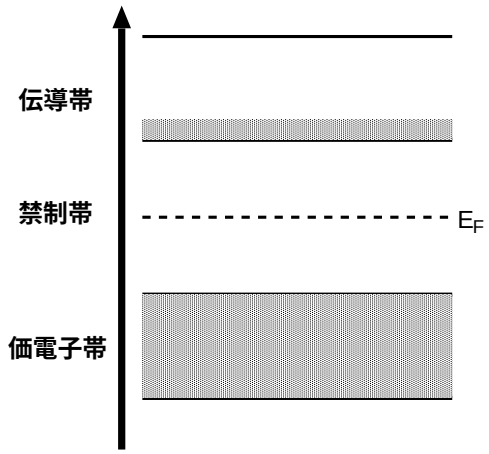

図 4　半導体のバンド構造 (家 正則・岩室史英・舞原俊憲・水本好彦・吉田道利編、「宇宙の観測 I －光 · 赤外線天文学」、日本評論社、2007、図 6.2 より引用)

1.5.2 フォトコンダクター

真性半導体もしくは不純物半導体を用いた検出器をフォトコンダクターと呼ぶ。

半導体の性質をもつ純粋な結晶のみでできた物質を真性半導体と呼ぶ。この真性半導体にバンドギャップよりも大きなエネルギーを持つ光子が入射すると光子のエネルギーによって電子が価電子帯から伝導帯に励起する。また電子が励起すると同時に価電子帯には電子の空席ができる。この空席はあたかも正の電荷を持った粒子のように振る舞い、これをホールと呼ぶ。このように電子とホールが対生成される。半導体中を電流が流れるときに電荷を運ぶのは電子とホールであり、この 2 つを合わせてキャリアと呼ぶ。電子とホールの対生成によってできるキャリアの移動による電気伝導度の増加から、真性半導体は光を検出することができる。

真性半導体にキャリアを発生させるための不純物元素を添加 (ドープ) したものを不純物半導体と呼ぶ。Si などの IV 族元素で 4 個の価電子の共有結合によって結晶を構成しているものに、P などの 5 個の価電子を持つ V 族の元素を加えると、電子が 1 個余分になる。この余分な電子の増加に伴って、エネルギーバンド構造には不純物準位が発生する (図 5 左)。この準位は不純物の余った価電子が存在する準位であり、もともと Si が形成していたバンドギャップのエネルギー

よりも低いエネルギーで電子が伝導帯に励起することが可能となる。このような不純物半導体を n 型半導体と呼ぶ。電子が伝導帯に励起されると負の電荷を持ったキャリアとなる。このとき、不純物元素からは電荷が奪われて正のイオンになるので、このエネルギーをイオン化エネルギーと呼ぶ。逆に B などの III 族元素を加えると価電子が 1 つ不足する。このとき、価電子帯のすぐ上に不純物準位ができる (図 5 右)。この準位は価電子の不足に対応して、電子が入り込める空席となっている。この場合でも、もともと Si が形成していたバンドギャップに比べて低いエネルギーで価電子帯の電子が不純物準位に励起されて入ることができる。このような不純物半導体を p 型半導体と呼ぶ。価電子帯の電子が励起されるとホールができる。そしてこのホールが正の電荷をもったキャリアとなる。このエネルギーもイオン化エネルギーと呼ぶ。n 型半導体や p 型半導体の不純物半導体では、イオン化エネルギーより大きなエネルギーを持った光子が入射するとキャリアが発生し、光を検出することができる。

真性半導体では電子とホールの対がキャリアとして働くのに対し、不純物半導体でのキャリアは n 型では電子、p 型ではホールに限られる。フォトコンダクターはキャリアを移動させたい方向に電極をとり、電極間に電圧をかけて使用する。この電場は、半導体に入射した光によって生成されたキャリアを移動させる。このキャリアの移動による電流を検出することで光を検出する。このとき、かけた電場に対してキャリアの流れる速度を移動度と呼ぶ。光子が検出器に入射したとき、キャリアを生成する割合を量子効率 η と呼び、生成されたキャリアのうち電極から取り出せる割合をフォトコンダクティブゲイン G と呼ぶ。検出器の感度はこの 2 つの掛算 ηG で決まる。フォトコンダクティブゲインはキャリアの移動度、電極にかける電場、再結合時間、電極間の距離の関数になる。G を大きくするためには高純度で格子欠陥の少ない結晶を用いればよい。しかし検出器のサイズやバイアス電圧で相反する条件があるため、さらに G を大きくし、感度を高くすることは簡単にはできない。また検出器の抵抗を高くして熱雑音を避けるのが難しい。そこで、これらの点を改善したのが次に述べるフォトダイオードである。

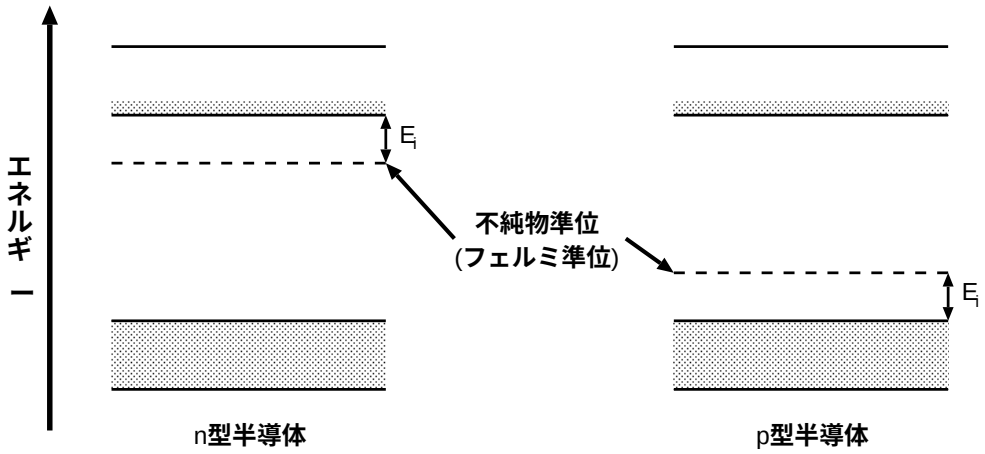

図 5　n 型半導体と p 型半導体 (家 他、2007、図 6.3 より引用)

1.5.3　フォトダイオード

フォトダイオードは p 型半導体と n 型半導体を接合した pn 接合によるダイオードによって光を検出するものである (図 6)。p 型半導体のフェルミ準位は真性フェルミ準位より価電子帯寄りに位置し、n 型半導体のフェルミ準位は伝導帯寄りに位置している (図 5)。またホールは価電子帯の上端付近に多数存在し、電子のエネルギー準位が下がるほど (ホールのエネルギーが高くなるほど) 少なくなる。一方、電子は価電子帯に多数存在し伝導帯の下端にも存在するが、電子のエネルギー準位が上がるほど存在確率は少なくなる。

いま p 型半導体と n 型半導体を平らな面を境として接触させ、p 領域と n 領域の間で自由に電子とホールが行き来できるようにした状態を想定する。p 領域と n 領域の間には電子密度とホール密度の大きな違いがあるので、p 領域のホールは n 領域に向かって、また n 領域の電子は p 領域に向かって拡散する。その結果、p 領域は電子を受け入れて負に帯電し、一方 n 領域はホールを受け入れて正に帯電する。p 領域の負の帯電と、n 領域の正の帯電により起電力が発生する。この起電力による電子とホールの移動をドリフトと呼ぶ。拡散とドリフトがつり合うところで平衡状態に達する。pn 接合に電圧が加わっていないときには電流が流れないので、p 型と n 型の場所によらずフェルミ準位は一定となり、図 6 に示すように $E_{\mathrm{F}}^{(\mathrm{p})} = E_{\mathrm{F}}^{(\mathrm{n})}$ となる。また接合部では電子とホールが接合することでキャリアがなくなる層が形成され、この

層を空乏層と呼ぶ。この pn 接合に対して n 型に正の電圧、p 型に負の電圧 (逆方向バイアス) をかけると、空乏層の電位差がさらに広がる。キャリアのない空乏層が広がるので、電流はほとんど流れることができない。

逆方向バイアスをかけた pn 接合に対して空乏層付近で光によってキャリアが生じると、このキャリアで接合をこえて電流を流すことができる。この光電流を測定したり、この電流によって蓄積される電荷量を測定することで光の検出を行う。生じたキャリアは空乏層の電場で加速され起電力を発生し、これを光起電力効果という。フォトコンダクターはフォトダイオードにはない空乏層がある分、高抵抗にすることができ、熱雑音を少なくすることができる利点がある。

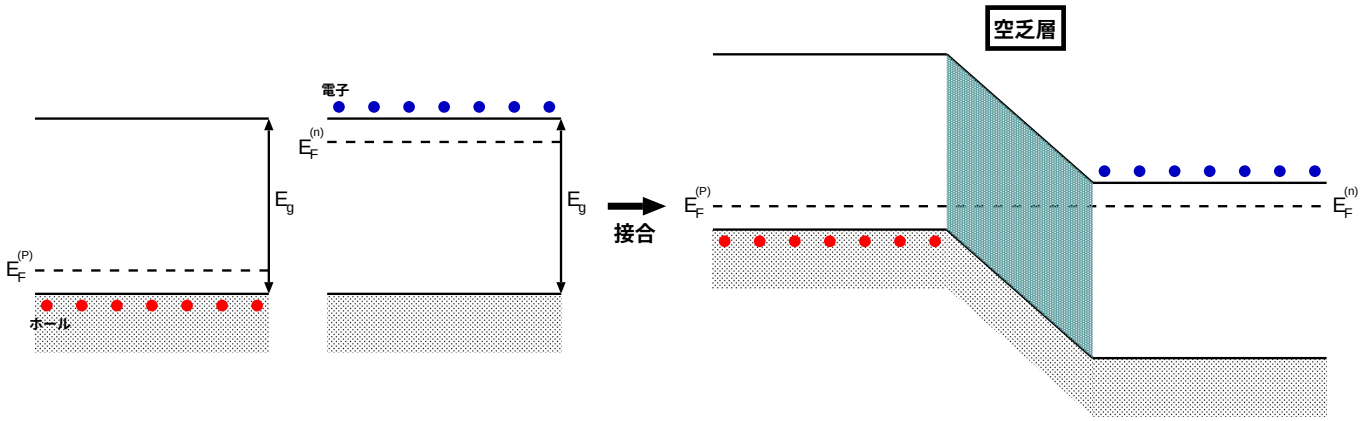

図 6　フォトダイオード (家 他、2007、図 6.4 より引用)

1.6　赤外線観測で使用する検出器

1.6.1　フォトコンダクター型検出器とフォトダイオード型検出器

これまで述べてきたフォトコンダクターとフォトダイオードを用いた検出器が赤外線の観測で用いられる。主に近赤外線ではフォトダイオード型の InSb や HgCdTe を、中間・遠赤外線ではフォトコンダクター型の Si:Sb、Si:As や Ge:Ga が使われる。現在、低いエネルギーに対応するバンドギャップを持つフォトダイオードが製作できないエネルギーの赤外線に対してはフォトコンダクターが用いられる。しかし、フォトコンダクターは放射線の影響によって感度が変化してしまう点と入射光に対する出力の応答が単純な比例関係にないという欠点を持つ。検出器応答の詳細な実験データをもとに観測データを解析の段階で補正をすることで実際の観測は行われている。更に、フォトコンダクターの持つ欠点を克服するために考え出された構造が BIB(Blocked Impurity Band)、もしくは IBC(Impurity Band Conduction) 構造である。これは、シリコン基板上に不純物を多く添加して光を吸収しやすくした層を作り、その上に不純物をほとんど含まない純粋な結晶の層を重ねた構造である (図 7)。不純物を多く添加した層で発生する暗電流を不純物をほとんど含まない純粋な結晶の層でブロックする。BIB 構造では光を検出する層は薄いが高い量子効率を得られ、物理的に薄いために放射線が当たる確率が少ないという利点がある。また添加不純物が多いので不純物準位のレベルが広がって伝導帯とのエネルギー差が小さくなり、感度が長波長側に伸びる。更に BIB ではフォトコンダクターの欠点である非線形性も小さくなる。ブロック層として極めて純粋な層を必要とするため、結晶成長技術の進んだシリコン基板を用いたものが実用化されている。これらの BIB 検出器はフォトダイオードの検出器と同様に大規模 2 次元アレイ検出器として、多くの装置に利用されている。

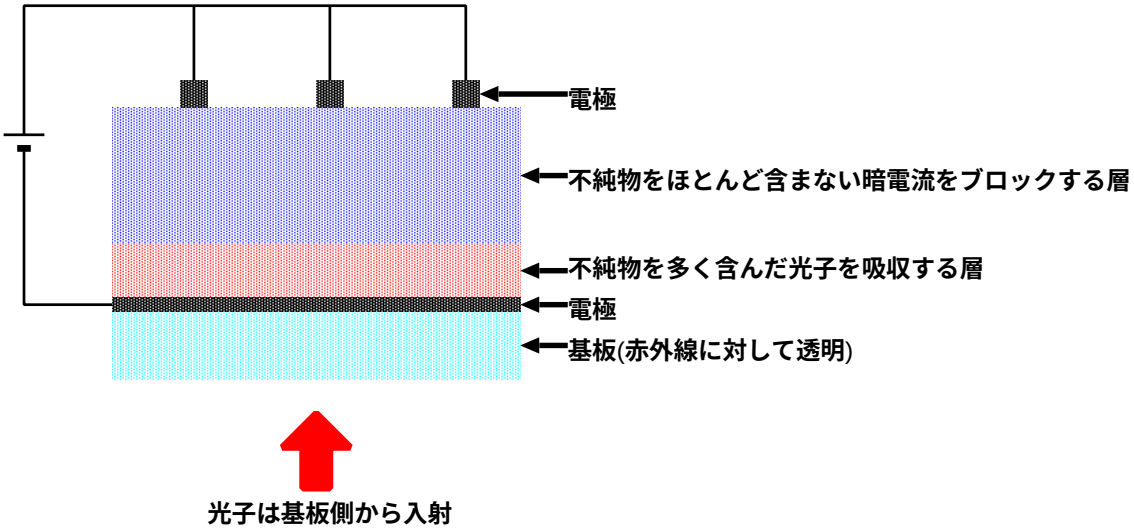

図 7　BIB 構造

1.6.2　大規模 2 次元アレイ検出器

ここまでは 1 素子の検出器について述べてきたが、実際は検出器を格子状に並べて使用する。検出器を格子状に並べた構造を array 構造と呼ぶ。この array 構造は赤外線検出器結晶部を検出器の読み出し回路 (マルチプレクサ) の上にインジウムなどの金属で接合することで可能にしている (図 8)。マルチプレクサとは、検出器の各受光素子に対応したユニットセルを基本とする回路である。ユニットセルは、検出器で発生した電荷を貯める積分コンデンサや下流への読み出しを行う MOSFET などで構成されている。また、マルチプレクサには MOSFET を用いた column select switch、row select switch が備わっている。このマルチプレクサでは、2 つの switch によって選ばれたユニットセルで得た情報のみ読み出すことができる構造になっている。

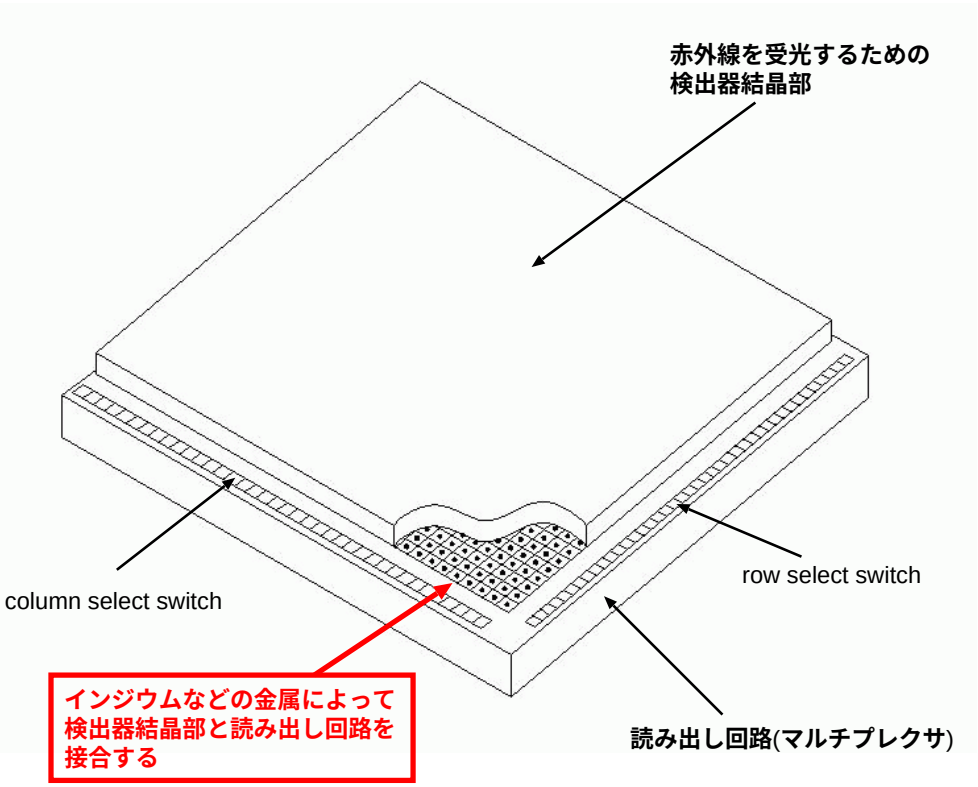

図 8　2 次元アレイ検出器 (家 他、2007、図 6.8 より引用)

1.6.3 MOSFET

次に読み出し回路内で多用されている MOSFET について述べていく。MOSFET(Metal-Oxide-Semiconductor Filed Effect Transistor) は金属と薄い酸化膜と半導体から構成され、半導体でもっともよく使われているトランジスタであり、回路図記号は図 9 で表される。MOSFET は n 型半導体を基板とする PMOS と、p 型半導体を基板とする NMOS の大きく 2 種類に分けられる。NMOS は P 型基板に 2 つの n 型半導体からなる N^+ 領域を埋め込み、N^+ 領域間の表面を覆う薄い酸化膜、その上に金属あるいは導電性のポリシリコンがのっている構造である (図 10)。低い電位を与える N^+ 領域をソース、高い電圧を与える N^+ 領域をドレイン, 酸化膜上の電極をゲートと呼ぶ。PMOS は n 型基板と p 型半導体を用いる以外はすべて NMOS と同じ構造である。以下、NMOS に関して述べるが PMOS の場合も電流等の方向が変わるだけで原理的には同じである。

NMOS のソース・ドレイン間は npn 接合のために電流は流れない。しかし、ゲート・基板間にゲート電圧 Vg が基板電圧より高くなるように電圧をかけると電場が発生し、p 型基板に存在する少数キャリアの電子がゲートに引きつけられる。電子の移動が進むとシリコン酸化膜と基板の境界付近の電子密度が高くなり、電子とホールの密度が同じになるところで空乏層が形成される。さらに逆電圧をかけるとホール密度を電子密度が上まり、p 型基板のゲート付近は n 型半導体へと反転する。この反転層はチャネルと呼ばれ、ソース・ドレイン間が n 型半導体でつながれるのでこのチャネルに電圧をかけることで電流が流れる。そこでソース・ドレイン間にドレイン電圧 Vd がソース電圧 Vs より高くなるように電圧をかけると、ドレインからソースへ向かって電流が流れる。このとき流れる電流をドレイン電流 I_{D} といい、ゲート電圧によって流れる量を制御することができる。

チャネルが形成されドレイン電流が流れ始めるゲート電圧を MOSFET のスレッショルド電圧 V_{th} と呼ぶ。このとき I_{D} と V_{G} の関係は図 11 のようになる。ゲート電圧 V_{G} を固定し、ドレイン電流 I_{D} とドレイン電圧 V_{D} の関係を表したものを $I_{\mathrm{D}} - V_{\mathrm{D}}$ 特性という (図 12)。ゲート電圧を V_{th} 以上で一定の値に固定し、V_{D} を 0 V から増加させるとその値に比例して I_{D} が増加する。この領域を線形領域という。さらに V_{D} を上げていくと、I_{D} の増加は少しずつ減少しほぼ一定となる。この領域を飽和領域という。MOSFET を switch として使用する場合は飽和領域で使用する。

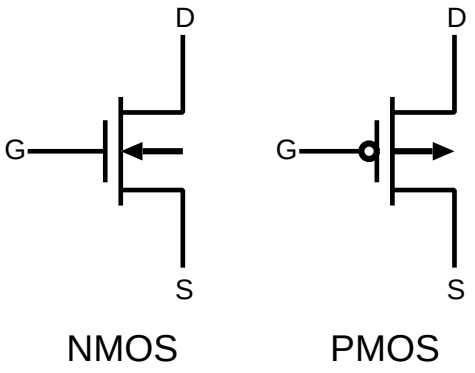

図 9　MOSFET の回路図記号

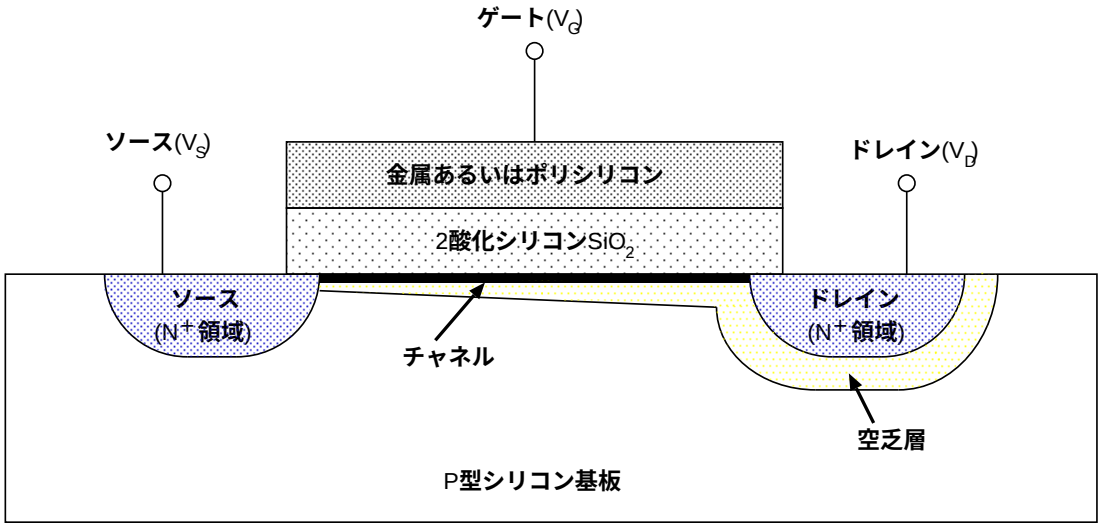

図 10　MOSFET の断面図

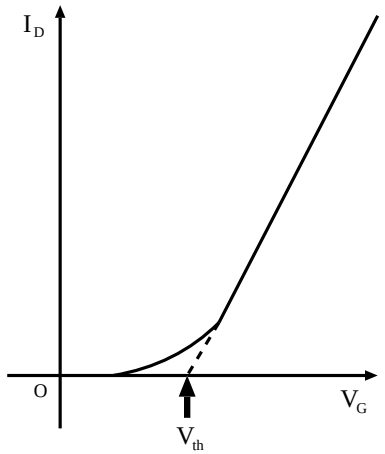

図 11　NMOS の I_D-V_G 特性

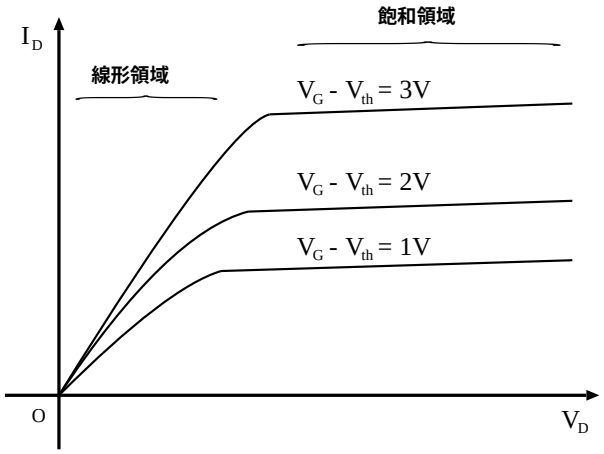

図 12　NMOS の I_D-V_D 特性

1.6.4 ソースフォロワ回路

MOSFET を飽和領域で使うためにドレイン・ソース間に定電流を流した状態で、ゲートに入力電圧 V_{in} を入力し、ソースから出力電圧 V_{out} を取り出す回路をソースフォロワ回路と呼ぶ (図 13)。ゲートへ V_{th} を越える電圧が与えられると、ソース電圧 $\mathrm{V_{out}} \sim (\mathrm{V_{in}} - \mathrm{V_{th}})$ となる。また MOSFET を飽和領域で使用することで、ゲート電圧 $\mathrm{V_{in}}$ とソース電圧 $\mathrm{V_{out}}$ の関係は以下の式で表される。

$$\mathrm{V_{out}} = \frac{g_m \mathrm{R_S}}{1 + g_m \mathrm{R_S}} \cdot \mathrm{V_{in}} \tag{7}$$

$$\begin{cases} \mathrm{R_S}\text{: ソースフォロワ抵抗 } [\Omega] \\ g_m = \mathrm{I_{DS}}/(\mathrm{V_{in}} - \mathrm{V_{out}})\text{: 相互コンダクタンス } [\mathrm{A/V}] \end{cases}$$

上式よりソースフォロワ回路の gain は 1 以下になること、また出力電圧 $\mathrm{V_{out}}$ の変化は入力電圧 $\mathrm{V_{in}}$ の変化を正確に追っていくことが分かる (図 14)。また、$\mathrm{R_S}$ を適切に選ぶことで gain∼1 が実現できる。その際、MOSFET のゲート電圧を制御することで、MOSFET を可変抵抗として $\mathrm{R_S}$ の代わりに使用する場合が多い。ソースフォロワ回路を用いると MOS FET のゲート・ソース間は電流が流れないので、非常に高いインピーダンスが実現できる。この特徴を利用して、検出器の読み出し回路はソースフォロワ回路を用いることで電圧の変化のみを下流へと伝えていく構造になっている。

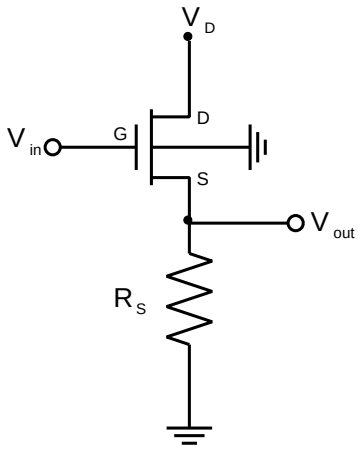

図 13　ソースフォロワ回路

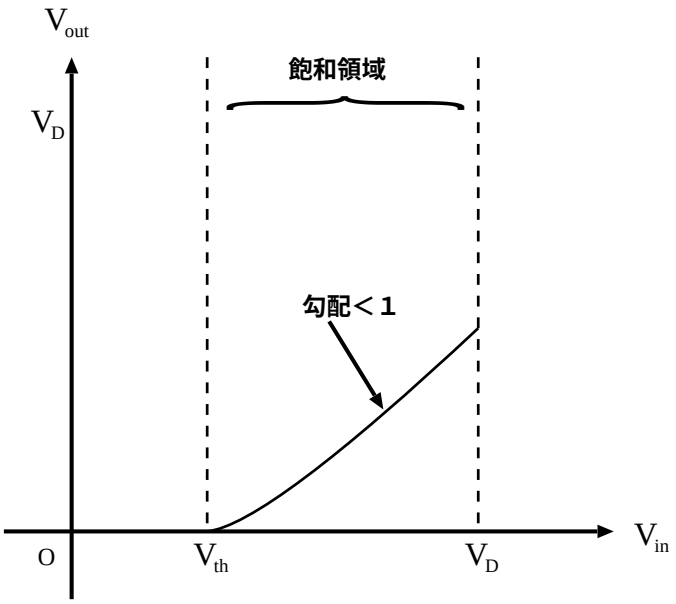

図 14　ソースフォロワ回路における $\mathrm{V_{in}}$ と $\mathrm{V_{out}}$ の関係

2　検出器の動作原理とデータ読み出しシステム構成

MIRSIS の検出器とそのデータ読み出しからなる検出器系システムは大きく 4 つのパートに分けることが出来る。

1. 検出器の感光部 (Si:As BIB 128×128 検出器、Si:As IBC 320×240 検出器)
2. マルチプレクサ
3. 検出器の駆動およびデータ読み出し回路群 (FOB,DRV,ADC,LVDS ボード)
4. 制御用計算機

図 15 は MIRSIS 検出器系システムの模式図で、矢印は Digital、Analog 信号をやりとりをする際の互いの関係を表している。この章では主に MIRSIS 撮像用検出器について、各パートの詳細について述べていく。

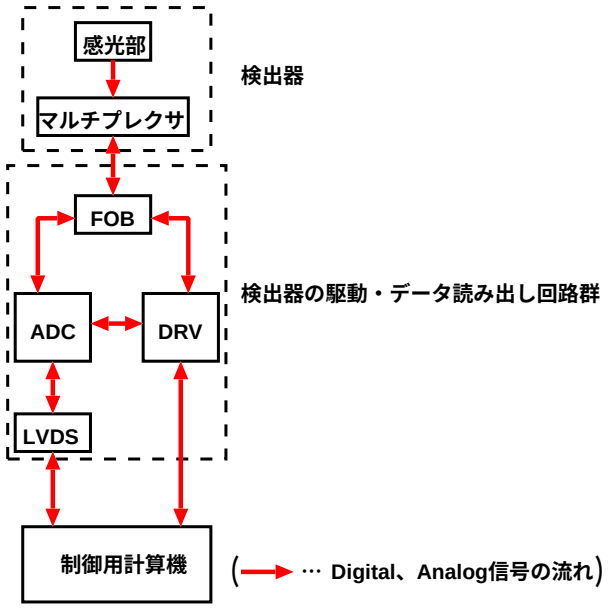

図 15　MIRSIS の検出器系システムの模式図

2.1　MIRSIS に用いる検出器

MIRSIS は撮像用検出器として Boeing 社製 Si:As BIB 128×128array を搭載する。また撮像用検出器のシステムを立ち上げる際の練習として、DRS 社製 Si:Sb BIB 128×128 array の試験も行った。これら 2 つの検出器は、共通のマルチプレクサを用いている。さらに分光用検出器として Raytheon 社製 Si:As 320×240 array を搭載する。これらの検出器の各 pixel は、インジウムバンプによって読み出し回路の各 unit cell に接続される。波長感度は 2-28μm で、このスペクトル領域での量子効率を上げるため anti- reflection coatings が施されている。検出器は <12K で駆動し、内部回路も極低温で動作するように設計が施されている。各検出器の基本仕様を表 2 にまとめた。

	Si:As 128×128 array	Si:Sb 128×128 array	Si:As 320×240 array
目的	MIRSIS 撮像用	試験用 (撮像)	MIRSIS 分光用
ピクセルサイズ (μm)	75×75	75×75	50×50
出力端子数 (outputs)	4	4	16 or 32
フレームレート (frame/sec)	>500	>500	100∼500
スペクトル感度 (μm)	2∼28	7∼40※	2∼28
暗電流 (electrons)	60∼3×10^5	4.4×10^6@T=8K	100@T=6K
動作温度 (K)	2∼12	<12	4∼10
電荷容量 (electrons)	∼10^7	∼10^7	1.0×10^7 or 3.0×10^7

※今回用いたのは検出器半導体ののっていないマルチプレクサのみ。

表 2　各検出器の主な仕様 (各検出器マニュアルより)

2.2 マルチプレクサの動作原理とクロック

2.2.1 マルチプレクサの動作原理

array 状に並べられた検出器の感光部で得たデータを読み出す回路をマルチプレクサと呼ぶ。この節では Si:As 128×128 array マルチプレクサの動作原理について述べていく。図 16 は、この検出器の感光部も含めたマルチプレクサの回路図である。このマルチプレクサは感光部と積分コンデンサを含む UNIT CELL と呼ばれる回路を基本として構成されている。UNIT CELL は検出器 pixel 数と同じ 128×128 個あり、array 状に配置されている。UNIT CELL 内の図 16 における「Detector」の部分で赤外線を受光し光子を電子へと変換したものを、積分コンデンサ C_{int} で蓄える。積分コンデンサでは電子を蓄えた分の電圧が増加し、Row Enable Select Switch と Column Enable Select Switch に選ばれた積分コンデンサのみが SF1 と SF2 の 2 つのソースフォロワを通じて下流へと読み出される仕組みになっている。

各 UNIT CELL に対応する検出器の感光部は、バンドギャップより大きなエネルギーを持つ光を受光すると光電子を発生する。この感光部に VDSUB 電圧をかけておくことで、インジウムによって結合した UNIT CELL に備えられている積分コンデンサ C_{int} へ電荷が移動する (図 17)。感光部と積分コンデンサの間にある FET は常にドレイン・ソース間電流が流れるようなゲート電圧 VDI がかけられている。積分コンデンサにおいて、リセット電圧レベルを基準とした光電子によるコンデンサ内の電圧増加が、検出される電圧となる。RESET FET にドレイン・ソース間電流が流れるようなゲート電圧をかけると C_{int} の検出器側が VRST 電圧にセットされ、この電圧がリセット電圧レベルとなる。この RESET FET は Slow Logic と Fast Logic によって操作されている。Slow Logic は FSYNC クロック、LSYNC クロックが決めるタイミングで各 UNIT CELL に電圧を出力する回路である。この電圧はオペアンプによって増幅され、UNIT CELL の RESET FET のドレイン・ソース間電流を流すためのゲート電圧を与えるタイミングや、Row Enable Select Switch のドレイン・ソース間電流を流すタイミングを決める。またこの Logic は LSYNC クロックの high/low 状態が切り替わるタイミングで Row を切替えていく。Fast Logic は LSYNC クロック、CLK クロック、RST クロックが決めるタイミングで各 UNIT CELL へ電圧を出力する回路である。この電圧はオペアンプによって増幅され、UNIT CELL の RESET FET のドレイン・ソース間電流を流すためのゲート電圧を生成し、Column Enable Select Switch のドレイン・ソース間電流を流すタイミングを決める。またこの Logic は CLK クロックの high/low 状態が切り替わるタイミングで Column を切替えていく。ここで出てきた 4 つのクロックについては次節で詳細を説明する。この 2 つの Logic に選ばれた UNIT CELL の Reset FET のみドレイン・ソース間電流が流れるようなゲート電圧が与えられ、C_{int} の検出器側にリセット電圧がセットされる (図 18)。

C_{int} の電圧変化は SF1 でソースフォロワによって下流へと読み出される。全 pixel は Row(行) と Column(列) でアドレス付けがされている。128 × 128 個ある unit cell に備えられた C_{int} の中で、Slow logic によって操作される Row Enable Select Switch と Fast Logic によって操作される Column Enable Select Switch によって選ばれた C_{int} の電圧のみが SF2 でソースフォロワによって下流へ読み出される。このマルチプレクサは、4 つの分離した output を持っている (図 19)。そこで、この output を用いて一度に読み出すことができる 4 つの pixel を 1 ブロックとして各 pixel は BL1(BLOCK1) から BL4096 のブロックに分けられ、ブロックごとに電圧は読み出される。

コンデンサの電圧変化を読み出す方法は single-sampling mode と multiple-sampling mode から選択できる。single-samplig mode ではリセット後から積分コンデンサに蓄積された積分電圧 V_{int} のみ読み出す (図 20)。リセットは次の pixel の読み出しを行っているときに起き、ピクセルアクセス時間 (ある pixel の電圧が 4 つの output のいずれかから読み出されている時間) の中では 1 レベルの電圧のみを読み出す。1 レベルの電圧のみの読み出しを行うので、このモードではより速く pixel に積分された電圧の読み出しが可能である。しかし、リセット電圧を読み出すことが出来ないのでリセット電圧と積分電圧の差を知りたい場合の使用には不適である。次に double-sampling mode では、リセットがピクセルアクセス時間の中で起きるので、2 レベルの電圧を読み取ることができる (図 21)。このモードでは積分電圧 V_{int} とリセット後に回路の熱雑音を蓄積することで発生するリセットノイズが上乗せされたリセット-リラックス電圧 V_{rr} の 2 レベルを読みとる。よって 1pixel で積分された電圧は $V_{int} - V_{rr}$ となるので、single-sampling mode と比較するとリセット-リラックス電圧の変動が除去されている分、正確に 1pixel で積分された電圧を得ることができる。MIRSIS ではこの double-sampling mode を採用している。

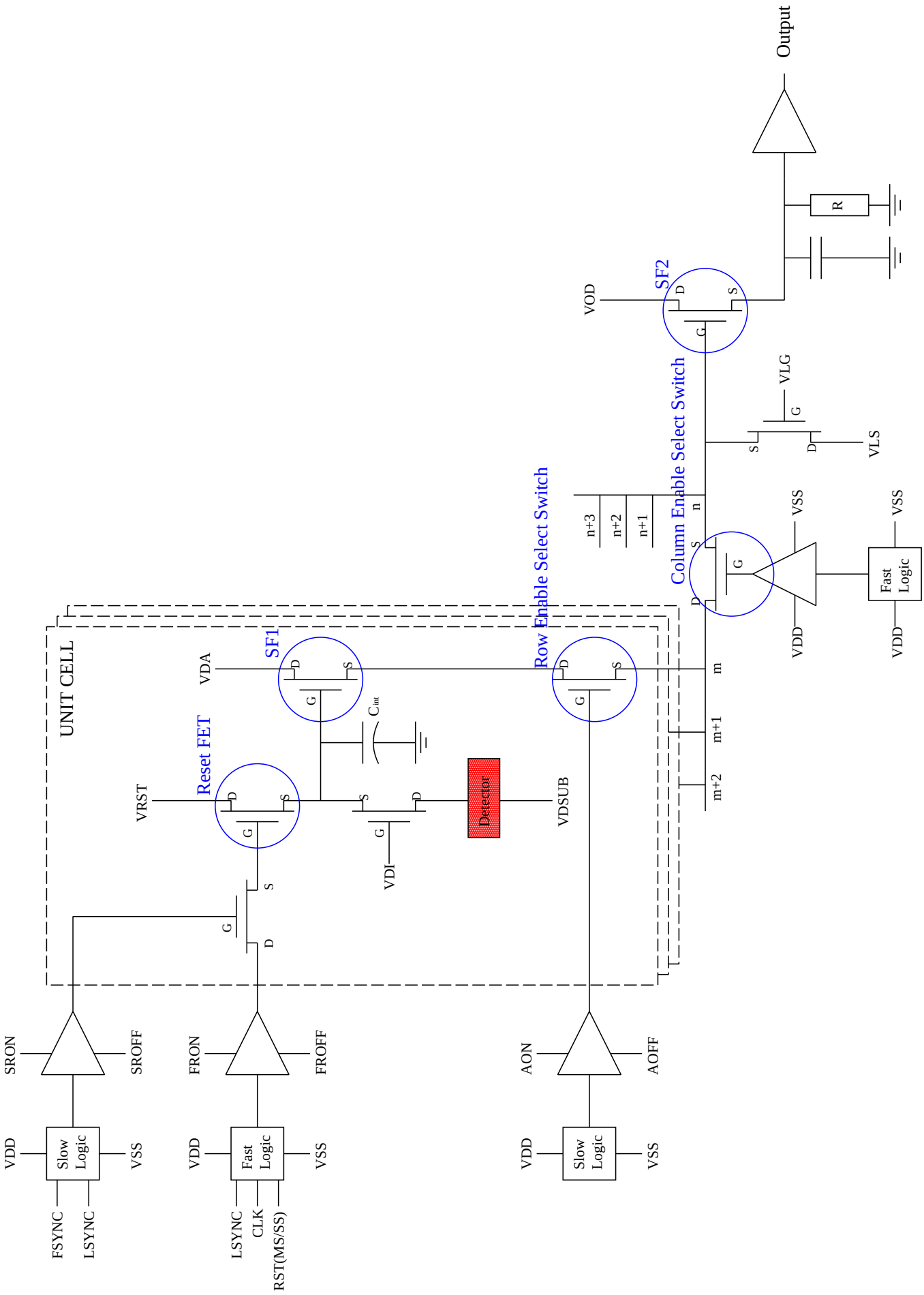

図 16　Si:As 128x128 array 読み出し回路図 (Device Description and Operation Guide for the DRS Technologies MF-128 BIB Hybrid Focal Plane Array p13 改)

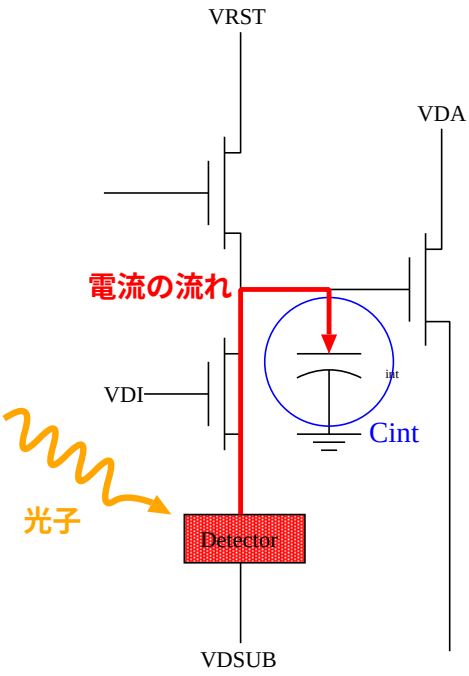

図 17　光子によって発生した光電流の流れ: 検出器の感光部である「Detector」の部分でバンドギャップ以上のエネルギーを持った光子を受光すると電流が発生し、この電流は C_{int} へ蓄積される

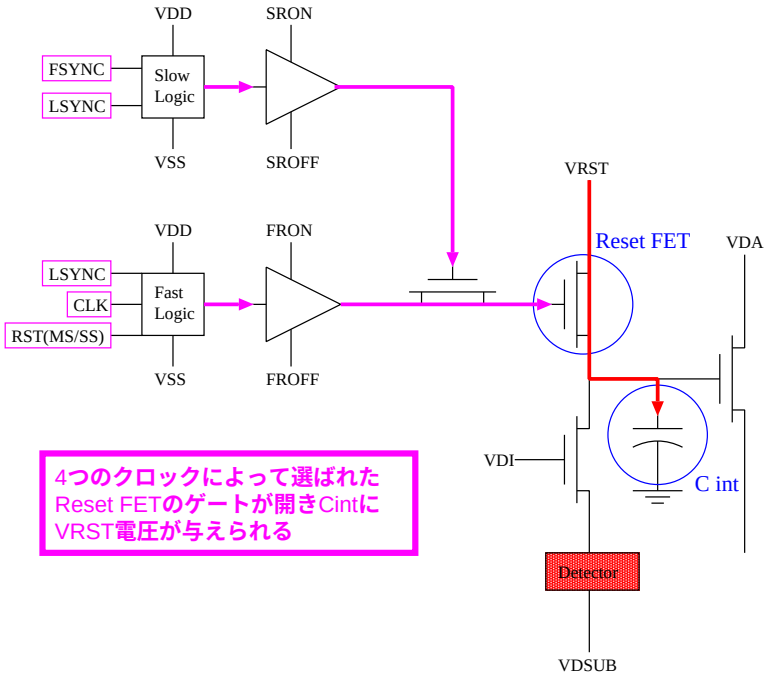

図 18　リセットが起きる仕組み

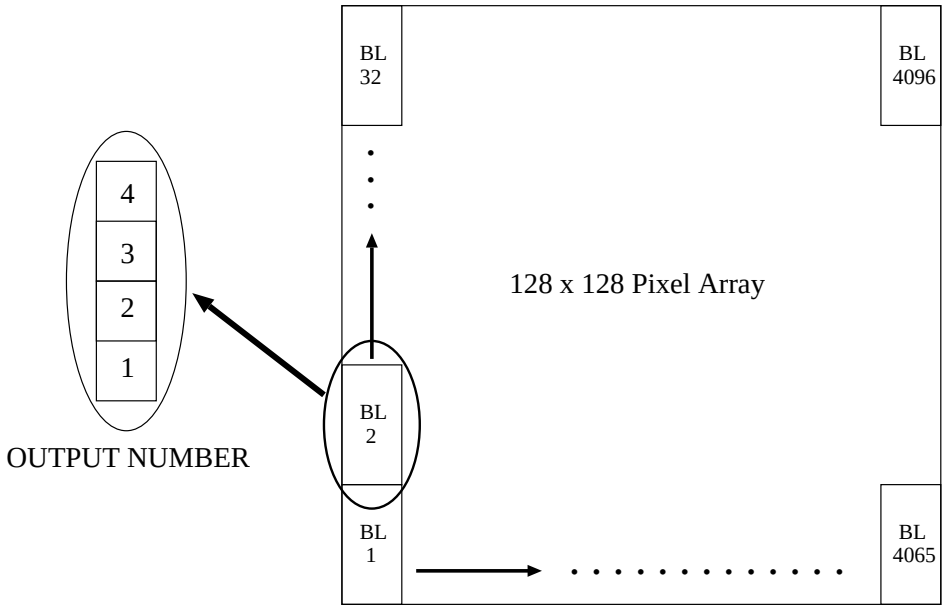

図 19　Si:As 128×128 array の output と pixel の関係: Si:As 128×128 array は 4 つの output を持っている。そこで、この output によって一度に読み出せる 4pixels を 1Block とし、各 pixel を 1∼4096 Block まで分割している

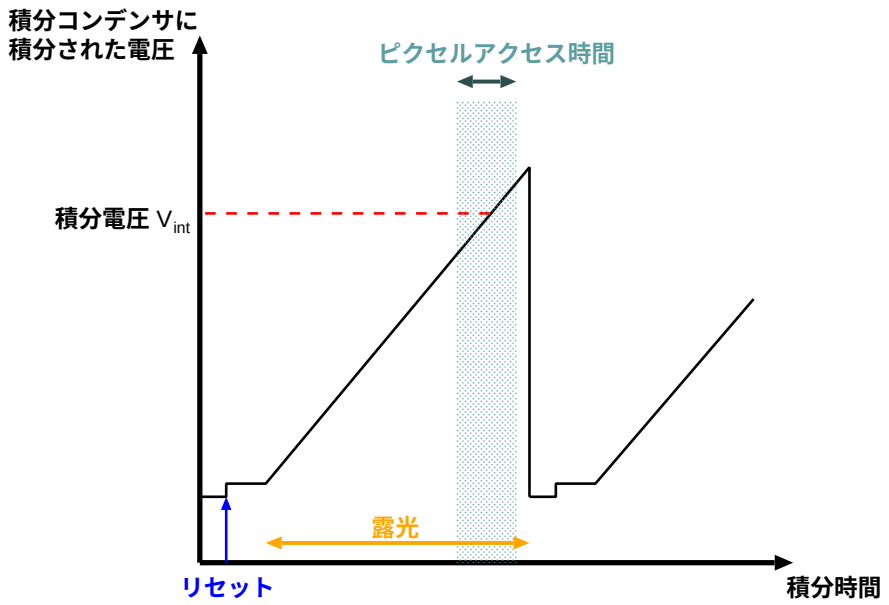

図 20　single-sampling mode で読み出す電圧

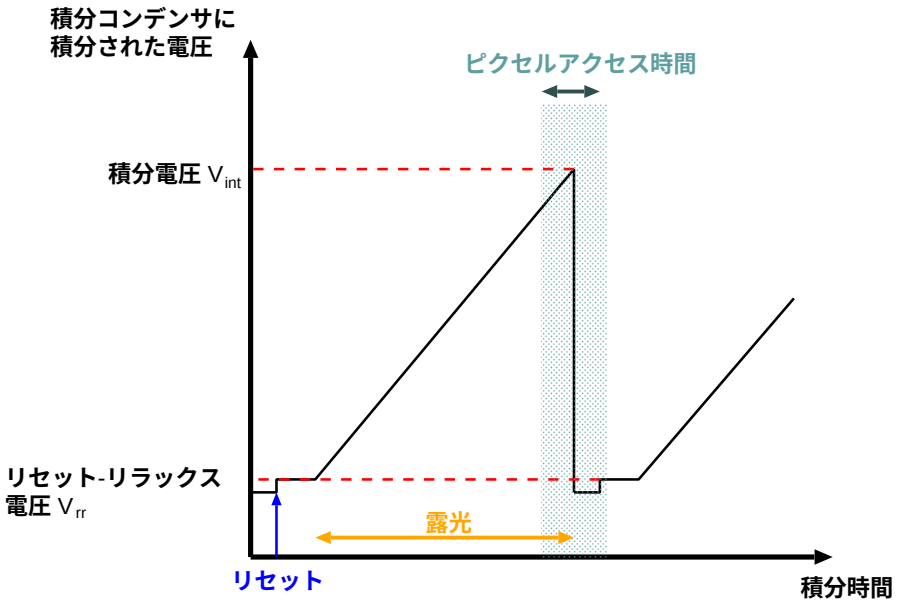

図 21　double-sampling mode で読み出す電圧

2.2.2 クロック

クロックとはマルチプレクサなどの検出器のシステムを制御するために与える矩形波の電圧で、MIRSIS 撮像検出器で使用するマルチプレクサの場合、low 状態で 3V、high 状態で 7V の矩形波を用いる。同検出器のマルチプレクサ内で使用している Slow Logic 及び Fast Logic は、FSYNC、LSYNC、CLK、RST クロックの 4 つのクロックによって動作している。各クロックの役割は以下の通りである。

- FSYNC クロック
  積分時間を決めるクロックで、1frame の画像を取得する際、FSYNC クロックが low 状態になっている時間を制御することで積分時間を変化させることができる。128×128 pixel へのアクセスを行う際 FSYNC クロックは全体を 64 分割されており、64 分割されたうちの low 状態の割合が 1pixel 当たりの露光時間の割合となり、high 状態の割合が 1pixel 当たりのリセット時間の割合となる。

- LSYNC クロック
  Column の切替えタイミングを決めるクロックで、LSYNC クロックの low/high 状態が変化するタイミングで Column が切り替わる。FSYNC クロックが 1 周期する間 LSYNC クロックは 64 周期のクロックになっていて、1 周期で 2 回 low/high 状態が変化するので、64 周期で 128 回 low/high 状態が変化することで 128 column へのアクセスを行う。

- DCLK クロック
  Row の切替えタイミングとピクセルアクセス時間を決めるクロックで、DCLK クロックの low/high 状態が変化するタイミングで Row が切り替わり、また low/high 状態が続いている時間がピクセルアクセス時間となる。LSYNC クロックが low/high 状態の間、DCLK クロックは 16 周期のクロックになっていて、1 周期で 2 回 low/high 状態が変化するので、16 周期で 32 回 low/high 状態が変化する。また、マルチプレクサは 4 つの出力端子を持っている。

  $$(\text{DLK の low/high 状態の変化数}) \times (4 \text{ つの出力端子}) = 32 \times 4 = 128\text{pixel} = 1\text{column}$$

  上式より、DCLK クロックは 128 pixel = 1 column へのアクセスを行っている。

- RST クロック
  DCLK クロックによって決まるピクセルアクセス時間内において、リセット電圧を与えるタイミングを決める。DCLK が low/high 状態の間、RST は 1 周期のクロックになっていて、RST が low から high に替わるタイミングでアクセスが行われている pixel(UNIT CELL) の積分コンデンサへリセット電圧が与えられる。また、このリセット電圧が与えられている時間の長さは FSYNC によって決められている。

以上の条件を踏まえた 128×128 pixel へのアクセスを行うクロックの例を図 22 に示す。

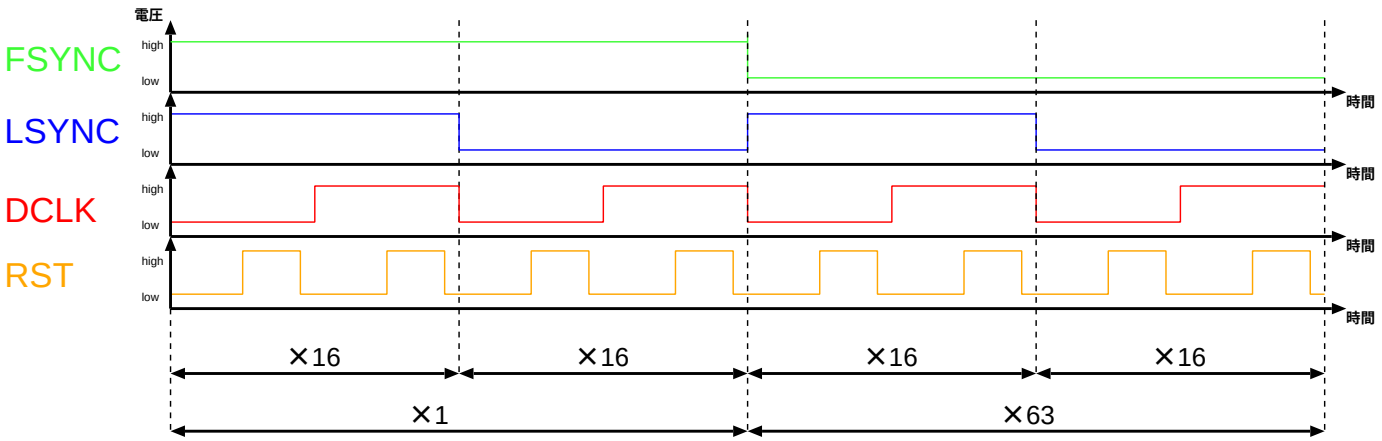

図 22　128×128 pixel へアクセスを行うクロックの例 (FSYNC の high/low の割合は可変で、128×128 pixel へアクセスを行う時間に対する FSYNC が high の割合を変化させることで、リセット電圧を与える時間を変化させる)

ここまで述べてきたクロックはマルチプレクサ内の Slow Logic、Fast Logic によって使用され、主に pixel へのアクセス、リセットタイミングと積分時間の制御を行う。しかし MIRSIS 撮像検出器にはこれらのクロック以外にも MIRSIS 撮像検出器のシステムを制御するために、以下で説明する 4 つのクロックが用意されている。

- CHP クロック
  望遠鏡のチョッピングのビーム切替えタイミングを決める。
- FRAME クロック
  画像取得の ON/OFF を決める。
- ADEN クロック
  AD 変換の ON/OFF を決める。
- ADTRG クロック
  ADC ボードの AD 変換器で AD 変換をするタイミングを決める。

上記のクロックを含めた 8 つのクロックを操作することで、MIRSIS 撮像検出器のシステムを制御している。

2.2.3 バイアス電圧

バイアス電圧とは、各検出器のマルチプレクサ内で使用している各電子部品 (MOSFET やオペアンプなど) を駆動するための電圧である。与えるバイアス電圧の数、値は使用するマルチプレクサや検出器感光部の種類によって違うので、バイアス電圧を設定する際は注意する必要がある。表 3、4 に MIRSIS 撮像検出器、分光検出器で使用するバイアス電圧を載せる。

撮像検出器用バイアス電圧の設定をする際、いくつかの注意がある。バイアス電圧 VDI はマルチプレクサの MOSFET のゲート電圧 (図 16 参照) として使用される際、4.0V である必要がある。しかし、洩れ電流が多いため、DRV ボード上では 4.5V に設定する。このような設定を行うことで、マルチプレクサで使用される際は VDI=4.0V になる。また、撮像検出器の検出器感光部に与えられる電圧 V_{Det}^{IM} は (VDI-1-VDSUB) で決まる。Si:As 128×128 array 検出器マニュアルでは、V_{Det}^{IM}=0.0∼1.5V の範囲で使用することを推奨している。この範囲外で検出器を使用すると、検出器を破壊する恐れがある。故に VDI=4.0V で使用する場合、VDSUB は 1.5∼3.0V の範囲で使用する必要がある。

分光検出器用バイアス電圧の設定をする際も注意がある。分光検出器の検出器感光部に与えられる電圧 V_{Det}^{SP} は (VGATE-1.3-VDETGRV) で決まる。Si:As 320×240 array 検出器マニュアルでは、V_{Det}^{IM}=0.0∼1.8V の範囲で使用することを推奨している。この範囲外で検出器を使用すると、検出器を破損する恐れがある。故に VGATE=-3.8V で使用する場合、VDETGRV は-6.9∼-5.1V の範囲で使用する必要がある。

バイアス電圧名	電圧値 (V)
S_RON	7.7
VNWELL	7.7
AON	5.5
VRST	5.0
VLG	3.0
VDI	4.5
VDSUB	2.0
VREF	2.0
AOFF	3.0
S_ROFF	4.0
VDA	4.5
F_ROFF	3.0
VOD	3.6
VLS	1.2
F_RON	6.7
VDD	7.3
VSS	3.0
CLKH	7.0
CLKL	3.0

表 3　MIRSIS 撮像検出器で使用するバイアス電圧

バイアス電圧名	電圧値 (V)
VDETGRV	-6.9
VNEG	-7.0
VDETGRDRG	-5.1
VPW	-6.0
VCLCOL	-5.5
VCLOUT	-4.1
VGATE	-3.8
VCASUC	-3.5
VSHPCHG	-3.0
VOFFSET	-2.8
VRSTUC	-2.0
IMUX	-1.7
VGG2	-1.9
VGG3	-1.8
VGG4	-1.6
CLKL	-5.0
VSS	-5.5

表 4　MIRSIS 撮像検出器で使用するバイアス電圧

2.3　検出器の駆動およびデータ読み出し回路群

本節では検出器の駆動およびデータ読み出しを行うための Fanout Board(FOB)、ドライバ (DRV) ボード、Analog/Digital コンバータ (ADC) ボード、Low Voltage Differential Signaling(LVDS) ボード、Digital In Out(DIO) ボードについて、それぞれの持っている役割と回路の概要、扱う際の注意などについて説明する。

2.3.1　FOB

FOB は検出器の固定と、静電気などの突発的な大電流から検出器の感光部やマルチプレクサに使用している電気部品を守る役割を持つ。検出器からの端子を FOB で 10MΩ の抵抗をはさんでフレームグランドに落としてある。また、検出器の駆動に必要な電圧を供給するバイアス電圧の線は、コンデンサをフレームグランドとの間にはさんでローパスフィルタ回路を形成している。このフィルタ回路によって高周波のノイズを除去している。図 23 に FOB の写真をのせる。

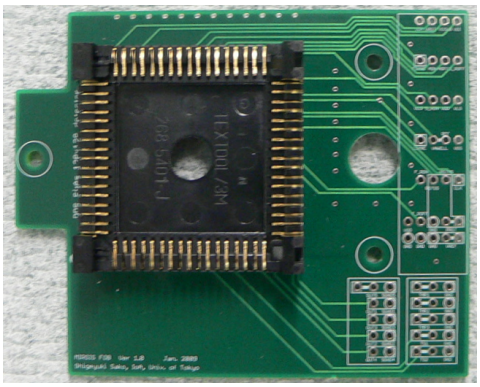

図 23　Si:As 128×128 array 検出器用 FOB の写真

2.3.2 DRV ボード

DRV ボードは、マルチプレクサで使用するバイアス電圧の生成を行う。この電圧はバイアス電圧と呼ばれ、MIRSIS 撮像検出器のマルチプレクサでは 19 個のバイアス電圧を与える必要がある。図 24 は、東京大学天文学教育センター (IoA) によって開発された DRV ボードを MIRSIS 撮像検出器用に移植したものである。ここで、1 個のバイアス電圧に注目して DRV ボードの動作を見ていく (図 25)。検出器制御用 PC で指定したパラメータが DRV ボードの Digital Analog Converter(DAC) に与えられる。DAC は Reference 電圧と指定したパラメータに応じた電圧を出力する。次に、出力された電圧はオペアンプを用いた増幅回路で電圧が増幅される。この増幅された電圧の出力先には switch が付いており、検出器制御用 PC で switch の on/off を切替えることができる。故に、検出器制御用 PC で指定したバイアス電圧のみを FOB を通じてマルチプレクサに出力することが可能である。DRV ボードはジャンパピンによってバイアス電圧の出力を制御している部分がある (図 26)。ジャンパピンを付け変えることで、VSENSE 電圧を使用するかどうか決めることができる。この VSENSE 電圧の設定は FOB を使用するかどうかで決まり、FOB を使用する場合はジャンパピンを ENA 側に接続し、FOB を使用しない場合はジャンパピンを DIS 側に接続して使う。

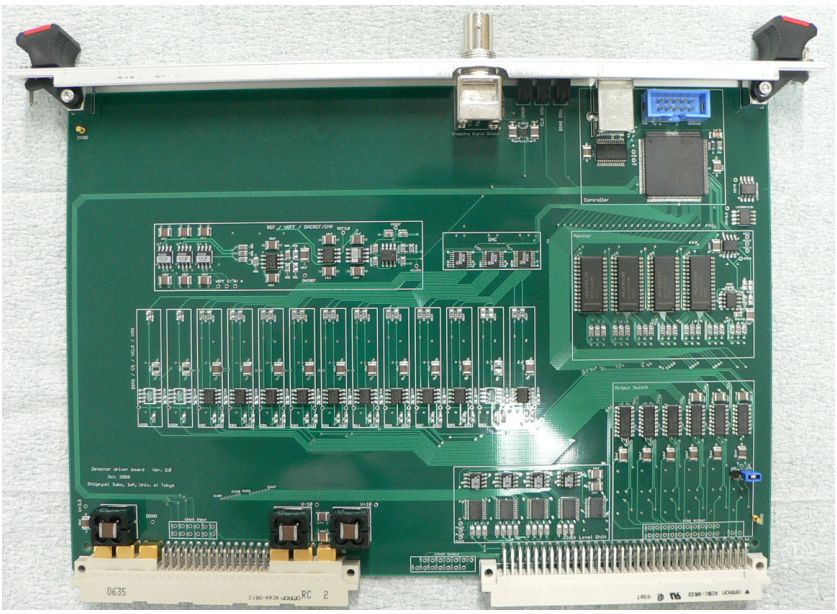

図 24　Si:As 128×128 array 検出器および Si:Sb 128×128 array 検出器用 DRV ボードの写真

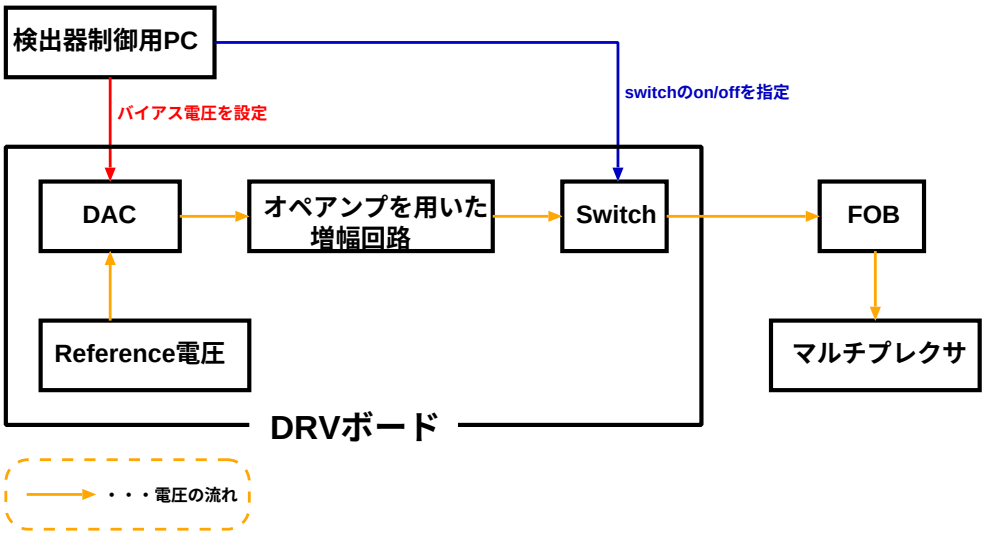

図 25　1 個のバイアス電圧に注目した際の DRV ボードの動作を表現したブロック図

図 26　ジャンパピンの写真 (左:FOB を使用する場合、右:FOB を使用しない場合)

2.3.3　ADC ボード

ADC ボードは検出器で得られた電圧を検出器制御用 PC で読み取れる形にするための Analog Digital(AD) 変換を行う。図 27 は、東京大学天文学教育センターによって開発された ADC ボードを MIRSIS 撮像検出器用に移植したものである。ここで、マルチプレクサから送られてくる電圧を ADC ボードで AD 変換する過程を見ていく (図 28)。128×128 array マルチプレクサの場合、検出器で得られた電圧を出力するための 4 本の出力端子、検出器の状態を確認するための参照信号を出力する 1 本の出力端子を合わせた、5 本の出力端子を持っている。この出力端子の数に合わせて ADC ボードでは 5 つの AD 変換器を搭載している。また、検出器で得られる電圧幅と AD 変換器で AD 変換を行うことができる電圧幅には違いがある。そこで、ADC ボードでは検出器から転送されてくる電圧を AD 変換器への適正入力レベルの電圧に変換するための増幅回路やフィルター回路が搭載されている。ADC ボードで AD 変換された電圧は、検出器制御用 PC へと転送され、データ処理が行われる。

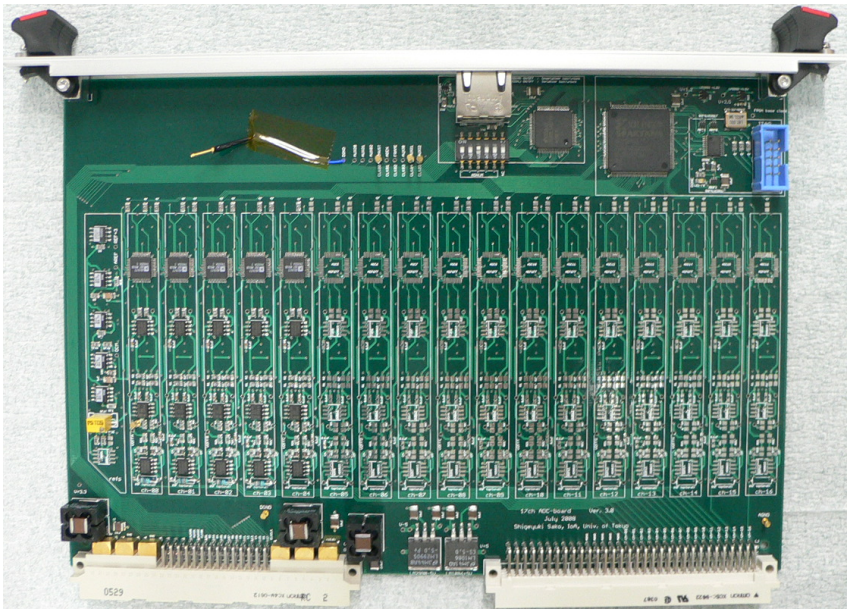

図 27　Si:As 128×128 array 検出器および Si:Sb 128×128 array 検出器用 ADC ボードの写真

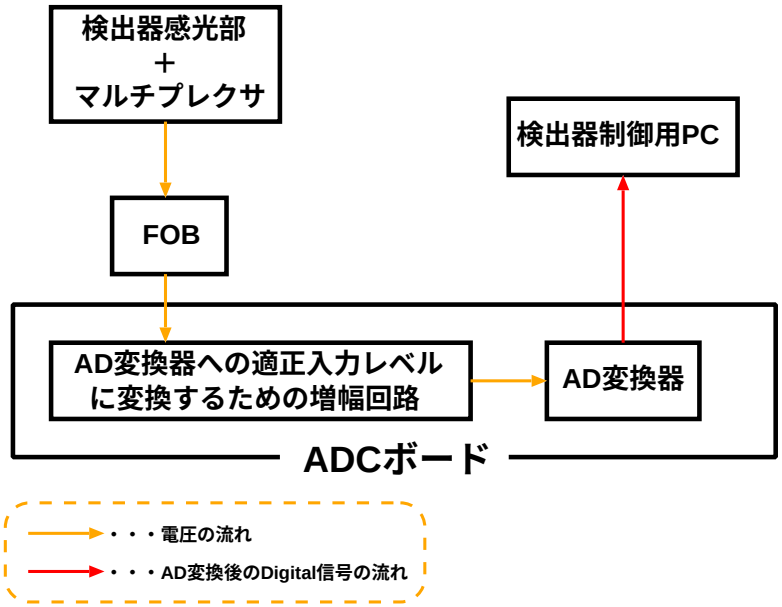

図 28　ADC ボードの動作を表現したブロック図

ADC ボードはディップスイッチを用いて AD 変換を行うチャンネル数を決めている (図 29)。ADC ボードには最大で 17 チャンネルの AD 変換を行うための AD 変換器を実装することが可能である。ディップスイッチの数字はそれぞれ 2 進数の桁を表しており、AD 変換を行うチャンネル数を 2 進数で表現する。ディップスイッチが ON 側にあると 2 進数の 1 を表し、反対側にあると 2 進数で 0 を表す。図 29 では 3 番のスイッチのみが ON になっているので、2 進数で「000100」を表している。これを 10 進数に変換すると

$$32 \times 0 + 16 \times 0 + 8 \times 0 + 4 \times 1 + 2 \times 0 + 1 \times 0 = 4$$

上式より、4 チャンネル分の AD 変換器を使用することを表している。このようにして使用する AD 変換器の数を変更する場合は、その数に合わせてディップスイッチを操作する必要がある。また、図 27 には 5 チャンネル分の AD 変換器が実装されているが、基本的には 4 チャンネル分のみ使用するので、ディップスイッチは 3 番のみ ON にしておけば良い。

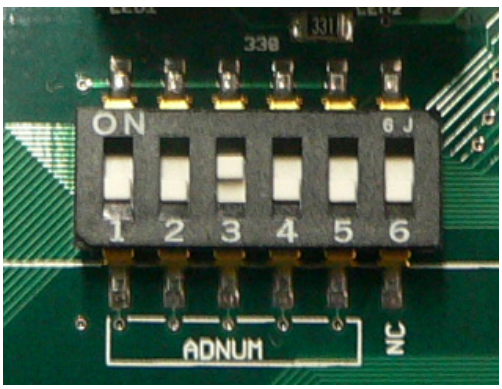

図 29　ADC ボードのディップスイッチ (写真は 4 チャンネル使用する場合)

2.3.4　LVDS ボード

LVDS ボードは DIO ボードから送られてくるパラレルな Digital 信号をシリアルな Digital 信号に変換して、ADC ボードへ転送する (図 30)。

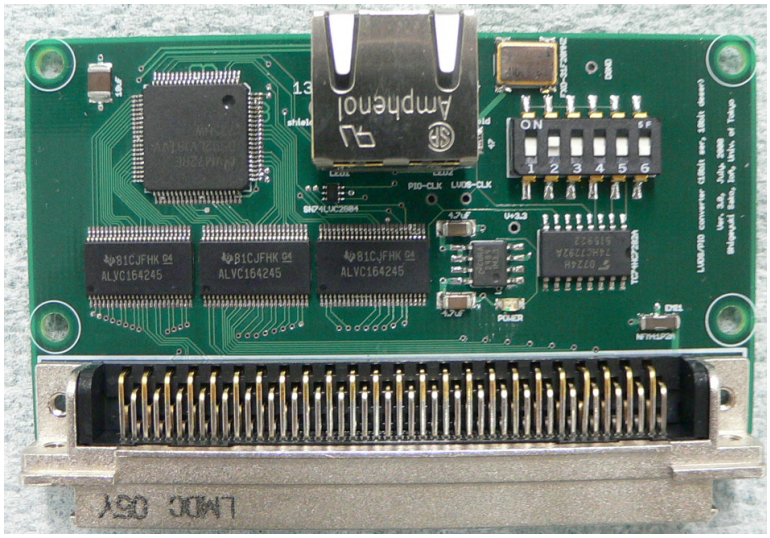

図 30　LVDS ボードの写真

2.3.5　DIO ボード

DIO ボードは検出器制御用 PC で指定したバイアス電圧やクロックを ADC ボードや DRV ボードに転送する。また、ADC ボードから転送される AD 変換後の検出器で得られた電圧を検出器制御用 PC へ転送する。MIRSIS では Interface 社製 2772c ボードを使用する (図 31)。DIO ボードには 4 つのジャンパピン接続端子がある (図 32)。ジャンパピンを接続することで、PCI バスの +5V 電源を DIO ボードから出力できるようになる。LVDS ボードは、+5V 電源を必要とする。故に、MIRSIS 検出器システムの場合は全ての端子にジャンパピンを接続して使用することで、DIO ボードから LVDS ボードへ電源を供給している。

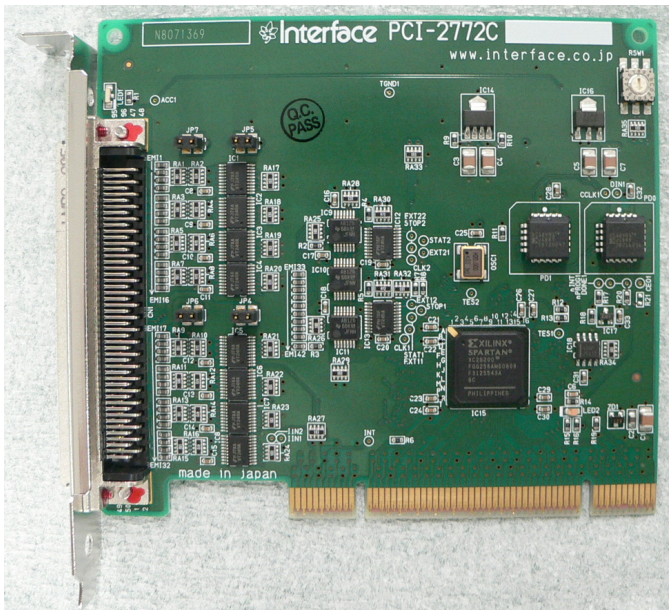

図 31　DIO ボードの写真

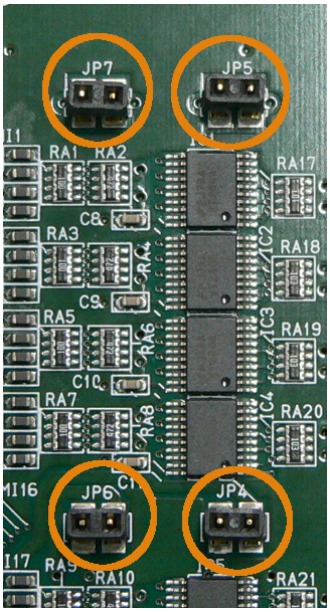

図 32　DIO ボードの 4 つのジャンパ端子

2.3.6 検出器の駆動およびデータ読み出し回路群の接続

ここでは、ここまで述べてきた FOB、DRV ボード、ADC ボード、LVDS ボード、DIO ボードの接続方法を説明する。DIO ボードは検出器制御用 PC の PCI バスに挿して使用する (図 33)。また ADC ボードと DRV ボードはエレキボックス (タカチ電機、FNS266-22-25KC[幅 42HP、高さ 6U]) に接続して使用する (図 34)。FOB は検出器箱に固定され、FOB 上には検出器とマルチプレクサが乗ったチップキャリアを固定する。(図 35)。これらの回路の接続方法を図 36 に示す。検出器制御用 PC の PCI バスに挿入された DIO ボードと ADC ボードは PCR96 ピンハーフピッチコネクタハーネス、LVDS ボード、CAT5 ケーブルを介して接続する。また、検出器制御用 PC と DRV ボードは USB ケーブルを介して接続する。エレキボックスの背面からはデータ通信用のケーブルと電源用のケーブルが出ており、データ通信用のケーブルは MIRSIS 真空容器に接続し、電源用のケーブルは直流電源に接続する。MIRSIS 真空容器内には FOB、検出器とマルチプレクサが一体となったチップキャリアを収めた検出器箱が固定されていて、真空容器に接続されたデータ通信用ケーブルからいくつかの配線を介して FOB に接続する。

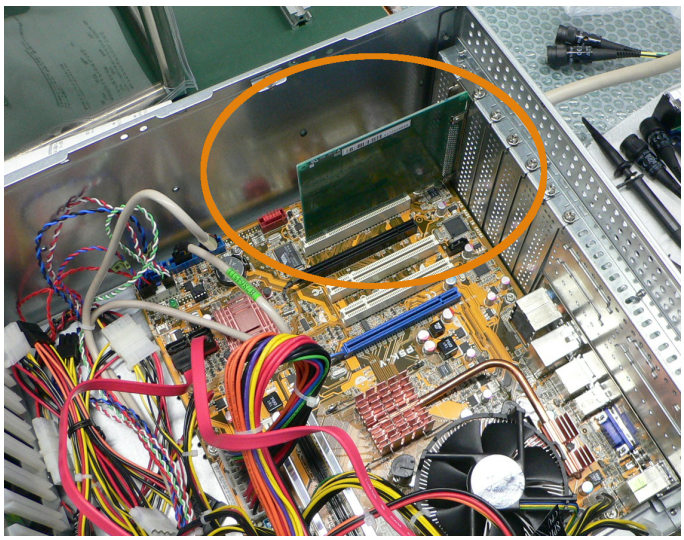

図 33　検出器制御用 PC の PCI バスに接続された DIO ボード

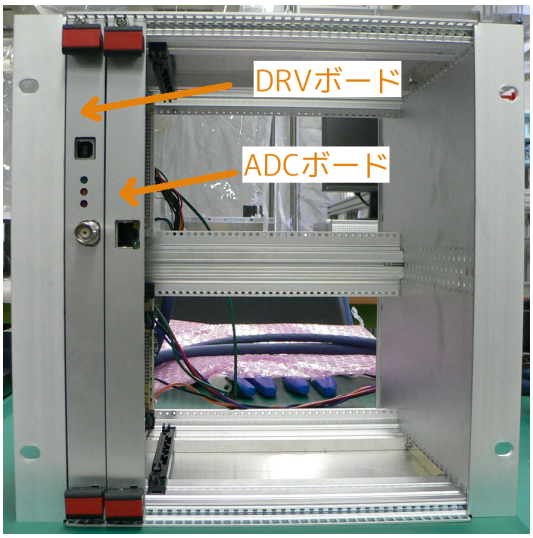

図 34　エレキボックスに接続された DRV ボードと ADC ボード

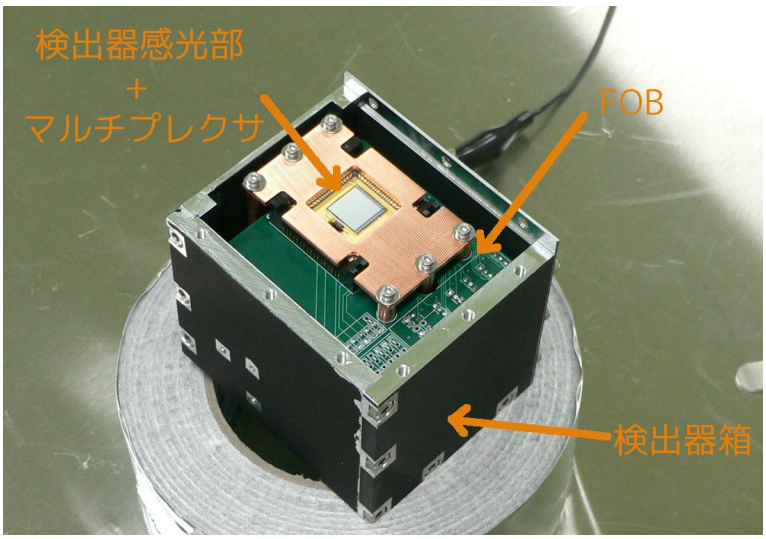

図 35　検出器箱内で検出器を固定している FOB

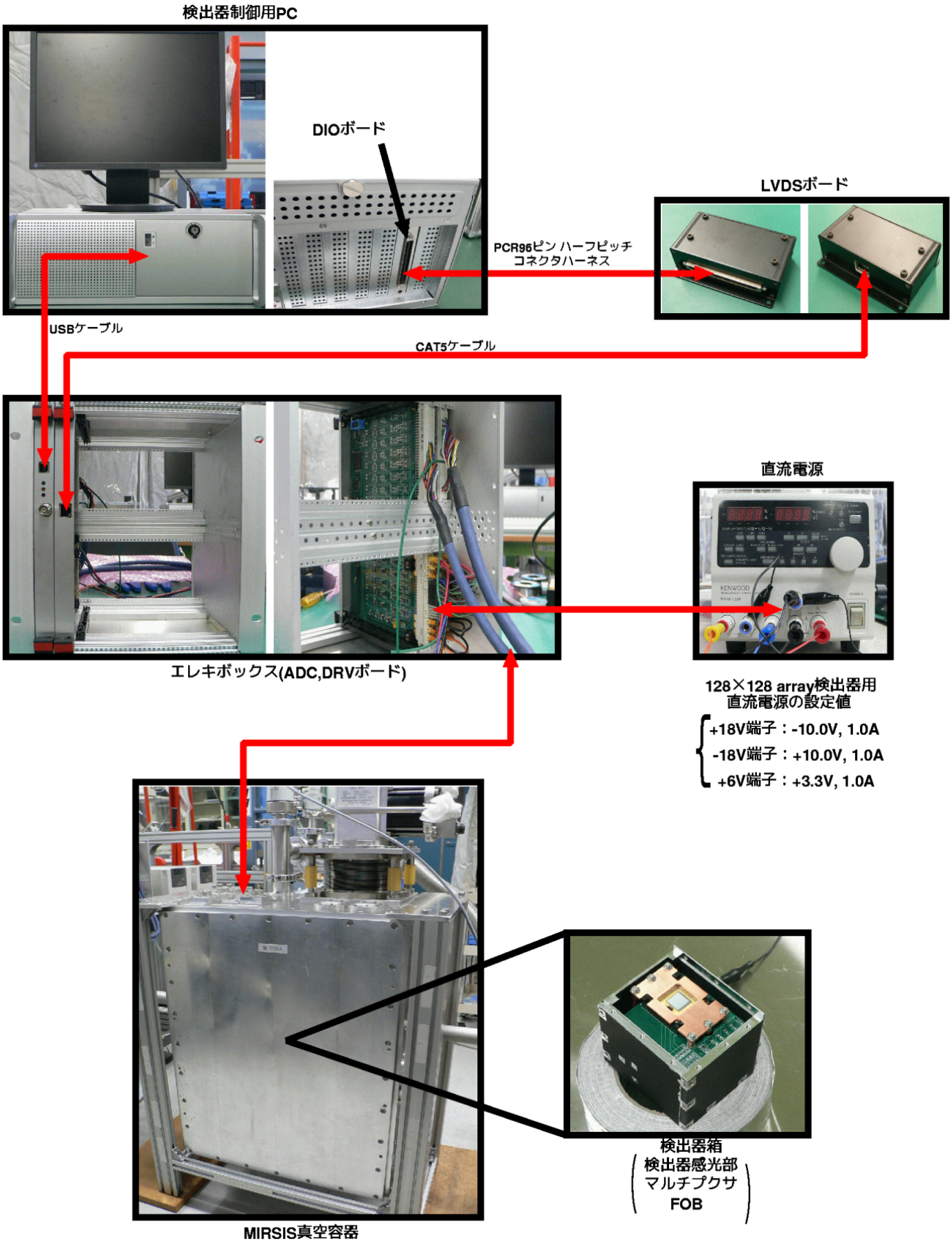

図 36　検出器の駆動およびデータ読み出し回路群の接続

2.4 検出器の駆動およびデータ取得システム

検出器の駆動およびデータ取得は FOB ボード, ドライバ (DRV) ボード, Analog/Digital コンバータ (ADC) ボード、LVDS ボード、DIO ボード、及び制御用計算機にて行う (図 37)。ここではこれらを用いた検出器の駆動・データ取得システムについて述べていく。

2.4.1 検出器の駆動システム

検出器の駆動システムは、バイアス電圧とクロック電圧のそれぞれを制御する。既に説明したように、バイアス電圧とはマルチプレクサに使用されている各電気部品を駆動するための電圧である。またクロック電圧とは、検出器の各 pixel へアクセスするタイミングを指定する電圧である。これらの電圧を制御する検出器制御用 PC には、Linux kernel ソースに RTAI パッチをあて、設定を変更しコンパイルした kernel と入れ換えることで Real time OS を実現したコンピュータを使用する。Real time OS とは時間に制約のある処理のみに機能を限定した OS で、通常の Windows OS などと比べて機能は少ないが応答速度が速いのが特徴である。検出器の操作には μsec 単位での制御が必要なので、この Real time OS を使用している。また、制御ソフトウェアには東京大学天文学教育センターによって開発された TAO array controller(TAC) システムを用いる。

この制御用 PC で指定したバイアス電圧のパラメータは USB ケーブルを介し、DRV ボードへ Digital 信号として転送される。この信号は DRV ボードの Digital Analog コンバータ (DAC) により Analog 信号へと変換され、MIRSIS 撮像検出器の場合、12 個のバイアス電圧と 7 個のカレントソース (MOSFET のドレイン・ソース間電流を流すためのバイアス電圧) をパラメータに応じて生成する。また DRV ボード上で、これらのバイアス電圧とカレントソースの出力端子に switch がついており、個別に ON/OFF の切替えが可能となっている。DRV ボードで作られたバイアス電圧とカレントソースは、FOB を通してマルチプレクサに転送される。このようにして、検出器が駆動するためのバイアス電圧は供給される。

撮像検出器の動作を制御するクロックはマルチプレクサの Fast Logic と Slow Logic で使用される 4 つのクロック (FSYNC、LSYNC、CLK、RST) と ADC ボードや望遠鏡で使用する 4 つのクロック (CHP、FRAME、ADEN、ADTRG) の 8 つのクロックから成り立つ (表 5)。制御用 PC で指定したクロックパターンは CPU を介さず、制御用 PC のメインメモリに直接書き込まれる。この動作を Direct Memory Access(DMA) と呼ぶ。その後、デジタルの入出力を行うための Digital In Out(DIO) ボードへ書き込まれ、Digital 信号として LVDS ボードへと転送される。LVDS ボードでは DIO ボードからパラレルに転送されてくる Digital 信号をシリアルデータへ変換し、ADC ボードを介して DRV ボードへ信号を転送する。クロック電圧は、制御用 PC の Clock control プログラムでパラメータを指定できる。DRV ボードへ転送された Digital 信号は、DRV ボード上の Clock level shifter によって検出器に適切な Analog 信号へ変換される。その後、Analog 信号のクロックパターンが FOB を通じてマルチプレクサにある Fast Logic と Slow Logic へと送られる。このようにして、マルチプレクサへクロックが与えられる。

2.4.2 検出器のデータ取得システム

クロックで指定したタイミングで、任意の UNIT CELL 中の積分コンデンサに積分された電圧がマルチプレクサから 4 本の出力端子を介して FOB を通り ADC ボードへと転送される。ADC ボードではプリアンプによって電圧が増幅され、Analog Digital コンバータ (ADC) によって Digital 信号へ変換される。この信号は LVDS ボードを経て DIO ボードに転送され、DMA により直接メインメモリへ書き込まれる。その後 TAC システムによって 128×128 の pixel データをまとめて 1 つの fits 画像へ変換する。この fits 画像として保存されたデータは ds9 プログラムなどで確認することができる。

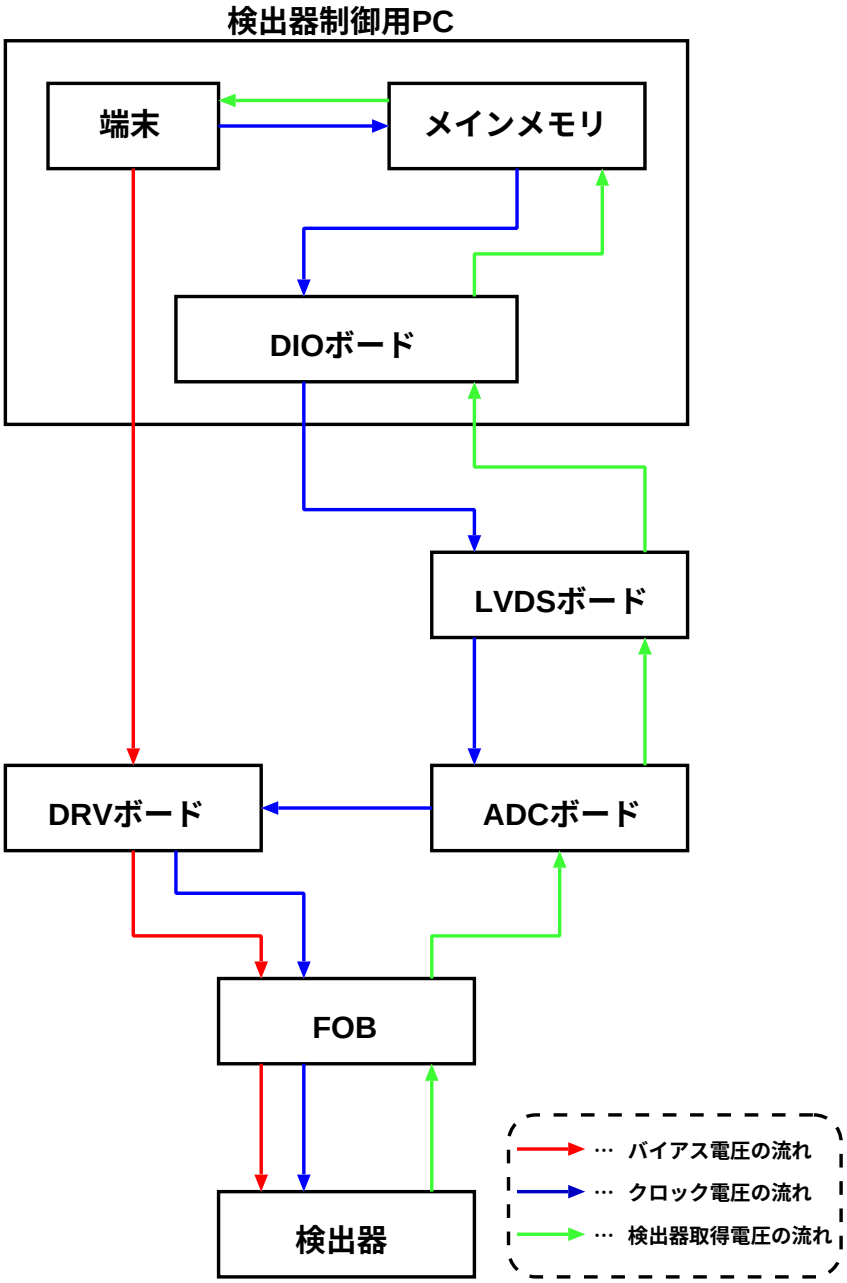

図 37　MIRSIS 検出器システムの概観

クロック名	内容
RST	リセットタイミングを与える
FSYNC	1 フレームの積分時間を決める
CLK	pixel アクセス時間を決める
LSYNC	行の切替えタイミングを与える
CHP	チョッピングのビーム切替えタイミングを決める
FRAME	画像取得の ON/OFF を決める
ADEN	AD 変換の ON/OFF を決める
ADTRG	AD 変換のタイミングを与える

表 5　クロックの種類と内容

3　検出器制御ソフトウェア

3.1　検出器制御ソフトウェア TAC システムの概要

検出器の制御ソフトウェアには東京大学天文学教育センターによって開発された TAO array coltroller(TAC) システムを用いる。一般的な検出器の読み出しにはデジタル信号処理に特化したマイクロプロセッサである Digital Signal Processor(DSP) を使用するが、TAC システムでは Real time OS と Direct Memory Access Digital In Out(DMA DIO) ボードを使用する (図 38)。この Real time OS は linux karnel ソースにフリーの RTAI パッチをあて、コンパイルした kernel と入れ換えることで実現する。このシステムのメリットとしてメインメモリ上で複雑な演算ができ、安価な DMA DIO ボードはバックアップを多く購入できることが挙げられる。MIRSIS 検出器制御用 PC では CPU に Intel Core 2 Duo Processor 3GHz、OS に linux 2.6.24 + RTAI3.6.1、DIO ボードに Interface 社の 2772c ボードを使用している。また、この DIO ボードでは、ある場所に格納したデータを古く格納した順に取り出すようにする First In First Out(FIFO) 方式を使用している。この方式では、一番新しく格納されたデータが一番最後に取り出される。次に、TAC システムのディレクトリ構成を図 39 に示す。システムディレクトリ以下は 3 つのディレクトリに分けることができる。TAC ディレクトリはデバイスドライバ関係、クロック関係、DIO ボードの制御、各ソフトウェアのライブラリの 4 つのディレクトリで構成されている。さらにバイアス電圧の制御を行う bis ディレクトリ、ステータスファイルを管理する stat ディレクトリがセットとなって TAC システムを構築している。また、クロックファイルや fits 画像データファイルはデータディレクトリに保存される。本章では各ディレクトリのソフトウェアの詳細について各ディレクトリの内容とプログラムの動作を説明していく。

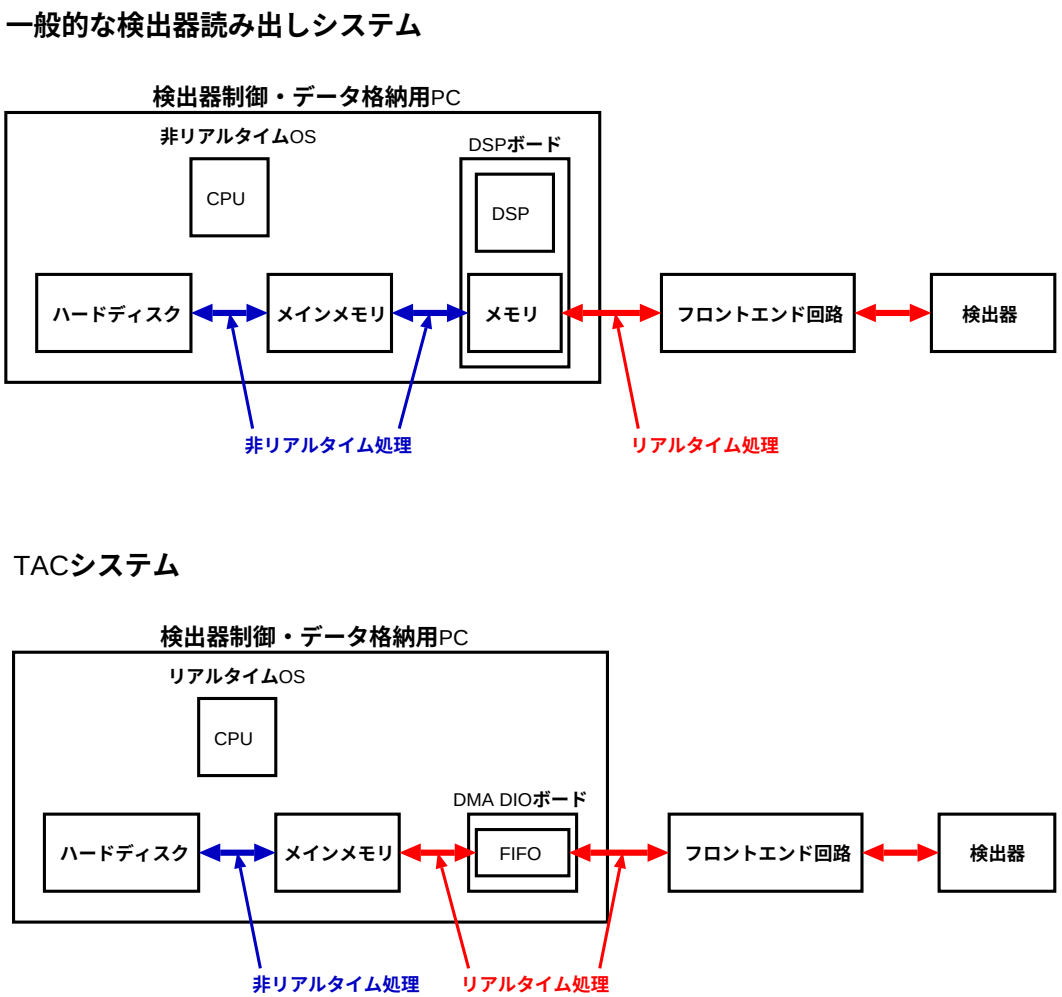

図 38　TAC システムのコンセプト

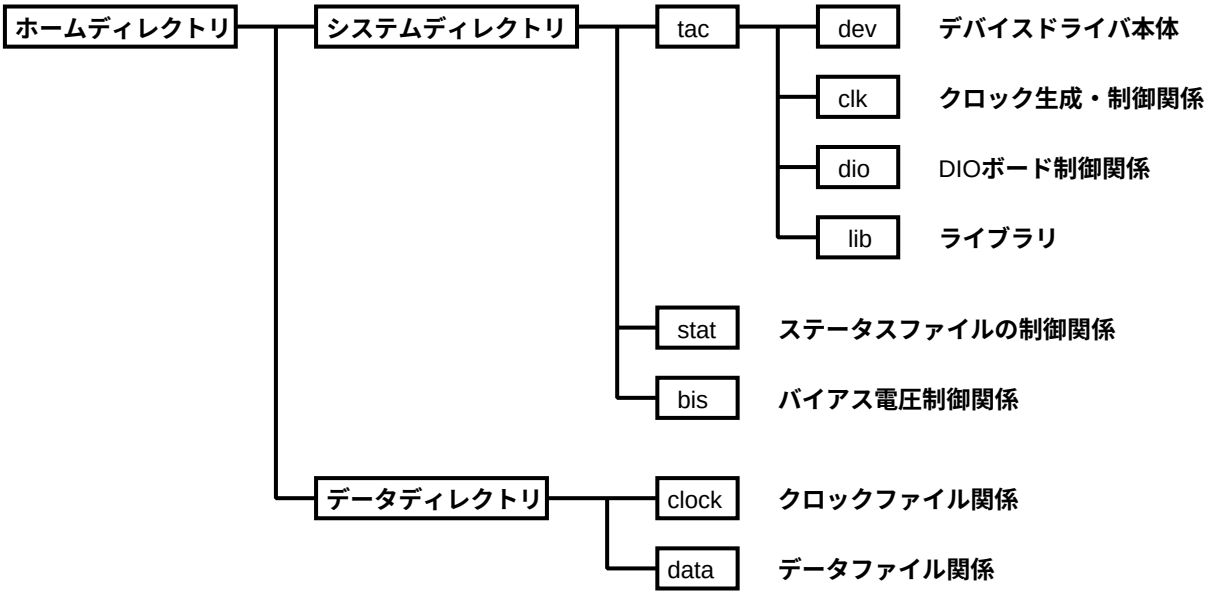

図 39　TAC システムのディレクトリ構造

3.2　デバイスドライバ (dev ディレクトリ)

デバイスドライバ関係のディレクトリの中には表 6 に示すファイルが存在する。TAC システムを使用する際、初めにこれらのプログラムを用いて RTAI モジュールと各デバイスドライバをインストールする必要がある。また、デバイスドライバ関係の操作にはスーパーユーザー権限が必要となる。そこで、これらの操作をユーザが使いやすくするために sudo コマンドの設定を行っている (図 40)。図 41 に sudo コマンドの設定を行った上での RTAI モジュール、i2772c.ko、tac_cor.ko モジュールのインストール/リムーブの端末での流れをのせる。図 41 のコマンドの順序は厳密に守る必要があり、各モジュールのインストールやリムーブを行う際は注意しなければならない。

ファイル名	内容
i2772c.ko	Interface 2772c ボードのデバイスドライバ
tac_core.ko	検出器制御用リアルタイムドライバ
rtai	RTAI モジュールのインストール/リムーブ実行プログラム
devload	i2772c.ko、tac_core.ko モジュールのインストール/リムーブ実行プログラム

表 6　dev ディレクトリ内に存在する主なファイルとその内容

```
$ su
パスワード:
# /usr/sbin/visudo
----------------------以下の内容を書き込んで保存----------------------
##Allow root to run any commands anywhere
root    ALL=(ALL)
mirsis  ALL=NOPASSWD: /home/mirsis/mirsis/tac/dev/rtai start
mirsis  ALL=NOPASSWD: /home/mirsis/mirsis/tac/dev/rtai stop
mirsis  ALL=NOPASSWD: /home/mirsis/mirsis/tac/dev/devload load
mirsis  ALL=NOPASSWD: /home/mirsis/mirsis/tac/dev/devload unload
```

図 40　sudo コマンドの設定

```
$ pwd
/home/mirsis/mirsis/tac/dev
$ sudo rtai start      <--RTAI モジュールのインストール
$ sudo devload load  <--i2772c.ko、tac_core.ko モジュールのインストール

$ pwd
/home/mirsis/mirsis/tac/dev
$ sudo devload unload   <--i2772c.ko、tac_core.ko モジュールのリムーブ
$ sudo rtai stop          <--RTAI モジュールのリムーブ
```

図 41　各モジュールのインストール/リムーブの端末での流れ

3.3　クロック関係 (clk ディレクトリ)

クロック関係のディレクトリの中には表 7 に示すファイルが存在する。これらのプログラムを用いて、マルチプレクサと ADC ボードに与えるクロックパターンの記述とバイナリクロックファイルの生成を行う。本節ではクロックパターンの記述方法について述べていく。

ファイル名	内容
cpc	バイナリクロックファイル生成プログラム
clkgen	クロックソースファイルを生成し、cpc プログラムを呼出してバイナリクロックファイルも生成するプログラム

表 7　clk ディレクトリ内に存在する主なファイルとその内容

3.3.1　cpc

cpc プログラムの文法に沿って記述されたクロックソースファイル (テキスト形式) をコンパイルし、実行可能なバイナリクロックファイルを生成するプログラムである。クロックソースファイルは観測をスムーズに行うため、2 バンク構造で記述する。2 バンク構造とはクロックプロセッサ (CLP) メモリ (メインメモリのクロックに関する情報を書き込むメモリ空間) のクロックコマンド領域を Bank-A/B の 2 領域に分ける構造である。Bank-A 領域に初期化 (INIT) ループ、待機 (STBY) ループ、データ取得 (ACQ) ループを記述する。Bank-B 領域に退避 (EVA) ループを記述する。この 2 バンク構造によって Bank-B 領域で EVA ループを実行すると同時に Bank-A 領域に別のバイナリクロックファイルをロードできるので、クロックプロセッサを停止することなくバイナリクロックファイルの変更が可能になる。各ループの動作内容は図 42 の通りである (各コマンドの詳細は後述参照)。図 43~49 はクロックソースファイルの例である。図 43 ではクロックソースファイル内で使用するパラメータや定数の定義を行っている。図 44~48 の 19 行目までは Bank-A の内容が記述されている。Bank-A の内容として図 44 の 6 行目 ~10 行目は INIT ループ、図 44 の 13 行目 ~ 図 45 の 26 行目は STBY ループ、図 45 の 28 行目 ~ 図 46 の 26 行目は ACQ ループが記述されている。図 46 の 29 行目 ~ 図 48 の 19 行目は Bank-A の各ループで使用するクロックの出力パターンを記述したビットデータ行 (詳細は後述参照) が記述されている。また、図 48 の 21 行目 ~ 図 49 にかけて Bank-B の内容である EVA ループと EVA ループで使用するビットデータ行が記述されている。ここからは図 43~49 をもとに、クロックソースファイルの記述方法について述べていく。

- 記号定数定義行とコメント
  クロックソースファイルは行を単位にして、基本的には 1 行に 1 つの命令を書く。また、図 43 の 1、2 行目のように#文字から行末まではコメント行とする。図 43 ではパラメータの設定を行っている。$文字から始まる文字列は変数として用いられ、この変数には整数値を格納する。変数に整数値を格納する行を記号定数定義行と呼ぶ。図 43 の 3 行目を例にみてみる。

  $ONEFRM 40960 # one frame size

  以上の記述で、$ONEFRM に整数値 40960 が代入される (# one frame size はコメント)。図 43 では同様の方法で、様々なパラメータの設定を行っている。

- outp コマンド
  図 44 の 4 行目を例にみてみる。

  outp &TAKEOFF_BA # Take off

  以上の記述で&TAKEOFF_BA のラベルがついたビットデータ行に記述されたクロックパターンを出力する。ビットデータ行については後ほど説明する。

- setv コマンドとコマンドデータ行のラベル
  図 44 の 7 行目を例にみてみる。

  @INIT_LOOP setv $IREG_MODE $MODE_INIT # mode -> Init

  以上の記述で$IREG_MODE を内部レジスタにし、$MODE_INIT の値 (図 43 の 31 行目より 1) をセットする。この内部レジスタは TAC システム内で使われる情報を保存するためのレジスタで、OS 上でアプリケーションがドライバを制御したり、ドライバと通常のデータの読み書きの流れの外で通信するために用意されたシステムコールである ioctrl の TAC_CLP_GET_IREG を使って読み出すことができる。また、outp や setv などのコマンドを使用している行をコマンドデータ行と呼び、コマンドデータ行にラベルを張る場合は@INIT_LOOP のように@文字を用いる。

- pjmp コマンド
  図 43 の 9 行目を例にみてみる。

  pjmp $JPORT_STBY @STBY_HEAD # to Init mode

  以上の記述でポート番号 (端末を用いて制御する値) が$JPORT_STBY (図 43 の 37 行目より 2) だった場合、@STBY_HEAD のコマンドデータ行にジャンプする。

- jump コマンド
  図 44 の 10 行目を例にみてみる。

  jump @INIT_LOOP # Return

  以上の記述で@INIT_LOOP のコマンドデータ行にジャンプする。故にこのクロックソースファイルでは、ポート番号 2 が与えられるまで@INIT_LOOP 行からこの jump 行をループする。

- load コマンド
  図 44 の 33 行目を例にみてみる。

  load $LREG_CHPLOOP　0　　# ChopN Loop [START]

  以上の記述で$LREG_CHPLOOP をループレジスタにし、値 0 をセットする。このループレジスタは後ほど説明する cjmp コマンドで使用される。

- subj コマンド、retn コマンド
  図 44 の 34 行目を例にみてみる。

  @STBY_CHP　subj $STACK_GIMG @STBY_LMF_T_BA_CS　# Get Image (Through, Stare)

  以上の記述でこのコマンドデータ行の 1 行下のコマンドデータ行を$STACK_GIMG に保存し、@STBY_LMF_T_BA_CS のコマンドデータ行にジャンプする。また図 45 の 26 行目を例にみてみる。

  retn $STACK_GIMG　　# Return

  上記の記述で subj コマンドによって保存されていたコマンドデータ行にジャンプする。

- cjmp コマンド
  図 44 の 35 行目を例にみてみる。

  cjmp $LREG_CHPLOOP @STBY_CHP　　# ChopN Loop [END]

  上記の記述で load コマンドによってループレジスタになった$LREG_CHPLOOP に記述された値が 0 でなければ値を-1 してから@STBY_LOOP にジャンプする。今回は図 44 の 33 行目より$LREG_CHPLOOP には値 0 が代入されているので、この行は飛ばされて次のコマンドデータ行に移る。

- halt
  今回のクロックソースファイルでは使用していないが、halt と記述するとクロックの出力を停止する。

- ビットデータ行
  図 46 の 49~61 行目を例にみてみる。

```
&L_SH_BA_FL_CS_PH　　bit　40 0000000000010010
                     bit　 1 0000000000010011
                     bit　 9 0000000000010010
                     bit　40 0000000010010010
                     bit　 1 0000000000010011
                     bit　 9 0000000000010010
                     bit　40 0000000000110010
                     bit　 1 0000000000110011
                     bit　 9 0000000000110010
                     bit　40 0000000010110010
                     bit　 1 0000000000110011
                     bit　 9 0000000000110010
                     endb
```

ビットデータ行のラベルには&文字を使用する。ビットデータ行の先頭には「bit」と記述し、その次に 0.2μsec を何回繰り返すかを記入する。故に 40 と記述した場合 40×0.2μsec=20μsec の間、同じビットデータを出力することになる。さらに各クロック電圧の low/high を 16 ビットで表現する。128×128 array 検出器の場合、16 ビットで表現した各桁は図 46 の 26 行目 ~38 行目で示しているクロック電圧の low/high 状態を表している。故に右端からそれぞれの桁は ADTRG、ADEN、FRAME CHP、LSYNC、DCLK、FSYNC、PRST クロックの low/high 状態を表している。ここまで述べてきたことを踏まえ、上記のビットデータ行によって出力されるクロックパターンを図 50 に示す。また「endb」は&L_SH_BA_FL_CS_PH でラベル付けされたビットデータ出力の終了を表す。

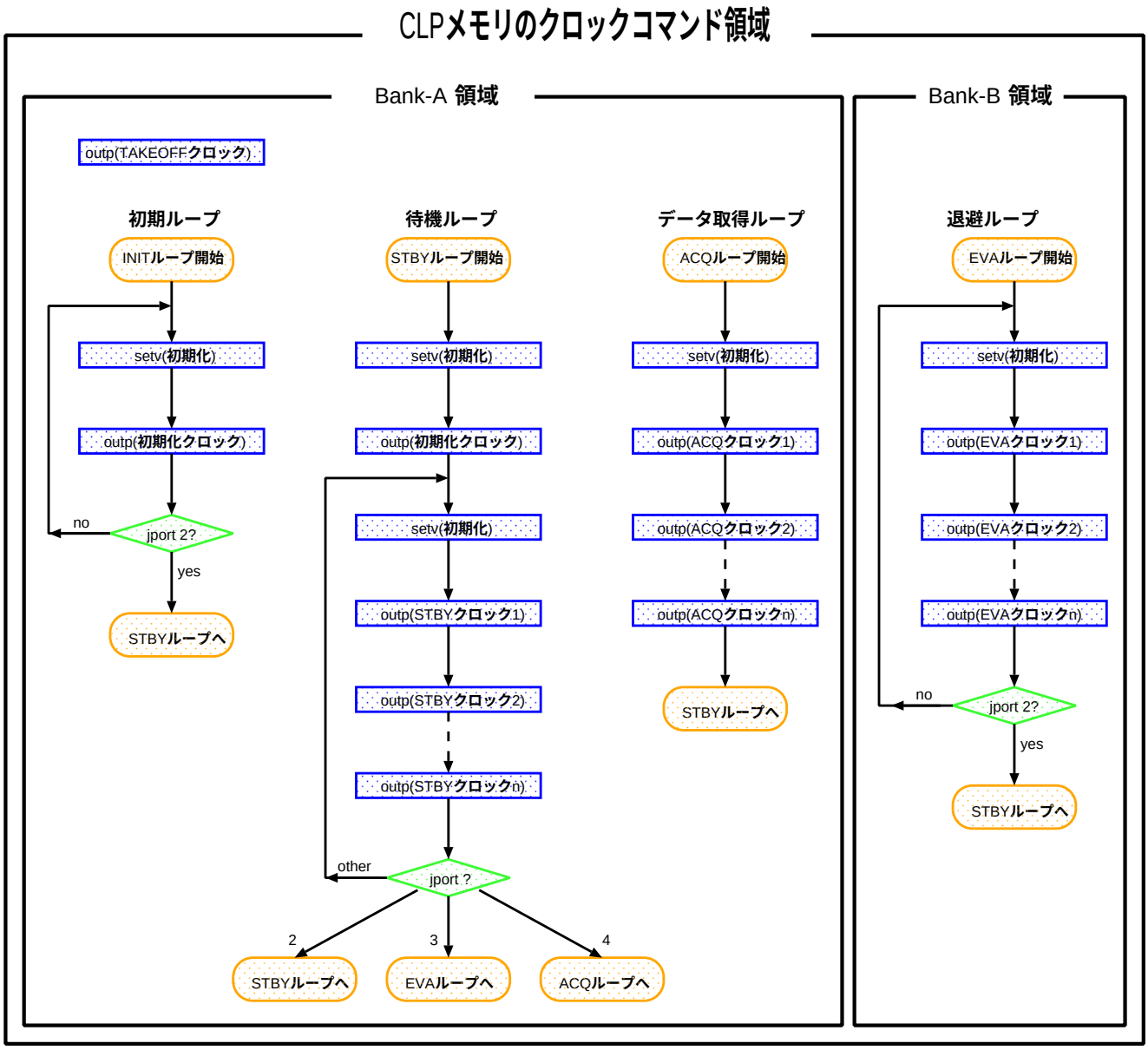

図 42　CLP クロックメモリ領域の 2 バンク構造

```
#=========================  Parameters  ================================================#
#--- FRP clock parameters ---
$ONEFRM              40960      # one frame size
$NEXP                10         # NExp
$NBEAM               1          # ChopL * ChopB
$COADD               0          # 0:Raw, 1:CoAdd
$CLKTYP              0          # 0,1,2,3,4,5
#--- Local clock parameters ---#
$BPP                 100        # BPP in 0.2usec
$PP                  100        # PP  in 0.2usec
$ND                  1          # ND
$WIPE                0          # Wipe
$CHOPL               1          # ChopL
$CHOPB               1          # ChopB
$CHOP                0          # 0:Stare, 1:Chop
$SROW                1          # Start Row
$SCOL                1          # Start Col
$WROW                128        # Width Row
$WCOL                128        # Width Col
#--- the Number of Loops ---#
$NCHOP               0          # ChopN-1
$NNEXP               8          # NExp-1-1
$NROWLOOP            15         # Nrloop-1
$NCOL                62         # (Ncloop-2)/2 -(ND-1)-1
#=========================  Definition  ================================================#
#--- Common ---#
$NULL                0
$OFF                 0
$ON                  1
#--- MODE (see cpc.h) ---#
$MODE_INIT           1
$MODE_STBY           2
$MODE_EVA            3
$MODE_ACQ            4
#--- JPORT (see cpc.h) ---#
$JPORT_INIT          1
$JPORT_STBY          2
$JPORT_EVA           3
$JPORT_ACQ           4
#--- IREG (see cpc.h) ---#
$IREG_MODE           0
$IREG_PINIT          1
$IREG_ONEFRM         2
$IREG_NEXP           3
$IREG_NBEAM          4
$IREG_COADD          5
$IREG_CLKTYP         6
$IREG_NULL           7
$IREG_BPP            8
$IREG_PP             9
$IREG_ND             10
$IREG_WIPE           11
$IREG_CHOPL          12
$IREG_CHOPB          13
$IREG_CHOP           14
$IREG_SROW           16
$IREG_SCOL           17
$IREG_WROW           18
$IREG_WCOL           19
#--- STACK ---#
$STACK_GIMG          1
#--- LREG ---#
$LREG_CHPLOOP        1
$LREG_NEXPLOOP       2
$LREG_WIPELOOP       3
$LREG_PPXLOOP        4
$LREG_PPWLOOP        5
$LREG_COLLOOP_H      6
$LREG_COLLOOP_L      7
$LREG_ROWLOOP1       8
$LREG_ROWLOOP2       9
$LREG_ROWLOOP_ODD_H  10
$LREG_ROWLOOP_EVEN_H 11
$LREG_ROWLOOP_ODD_L  12
$LREG_ROWLOOP_EVEN_L 13
$LREG_COLLOOP SK     14
$LREG_WIPELOOP       3
$LREG_PPXLOOP        4
$LREG_PPWLOOP        5
$LREG_COLLOOP_H      6
$LREG_COLLOOP_L      7
$LREG_ROWLOOP1       8
$LREG_ROWLOOP2       9
$LREG_ROWLOOP_ODD_H  10
$LREG_ROWLOOP_EVEN_H 11
$LREG_ROWLOOP_ODD_L  12
$LREG_ROWLOOP_EVEN_L 13
$LREG_COLLOOP_SK     14
```

図 43　クロックソースファイルの例 (ページ 1)

```
#=========================  Bank A  ==================================================#

#------------------------  Take off  ---------------------------------------------------#
                      outp &TAKEOFF_BA                          # Take off

#------------------------  Initialization ----------------------------------------------#
@INIT_LOOP            setv $IREG_MODE      $MODE_INIT           # mode -> Init
                      outp &CBINIT_BA                           # Init clock bits
                      pjmp $JPORT_STBY     @STBY_HEAD           # to Init mode
                      jump @INIT_LOOP                           # Return


#------------------------  Standby mode Main-routin  -----------------------------------#
@STBY_HEAD            setv $IREG_MODE      $MODE_STBY           # mode -> Standby
                      setv $IREG_ONEFRM    $ONEFRM              # set one frame size
                      setv $IREG_NEXP      $NEXP                # set NExp
                      setv $IREG_NBEAM     $NBEAM               # set NBeam
                      setv $IREG_COADD     $COADD               # set CoAdd
                      setv $IREG_CLKTYP    $CLKTYP              # set ClkTyp
                      setv $IREG_BPP       $BPP                 # set BPP
                      setv $IREG_PP        $PP                  # set PP
                      setv $IREG_ND        $ND                  # set ND
                      setv $IREG_WIPE      $WIPE                # set Wipe
                      setv $IREG_CHOPL     $CHOPL               # set ChopL
                      setv $IREG_CHOPB     $CHOPB               # set ChopB
                      setv $IREG_CHOP      $CHOP                # set Chop
                      setv $IREG_SROW      $SROW                # set Start Row
                      setv $IREG_SCOL      $SCOL                # set Start Col
                      setv $IREG_WROW      $WROW                # set Width Row
                      setv $IREG_WCOL      $WCOL                # set Width Col
                      outp &CBINIT_BA                           # Reset clock bits
@STBY_LOOP            setv $IREG_NULL      $NULL                # Standby loop header
                      load $LREG_CHPLOOP   0                    # ChopN Loop [START]
@STBY_CHP             subj $STACK_GIMG     @STBY_LMF_T_BA_CS    # Get Image (Through, Stare)
                      cjmp $LREG_CHPLOOP   @STBY_CHP            # ChopN Loop [END]
                      pjmp $JPORT_ACQ      @DATACQ              # to Data Acq mode
                      pjmp $JPORT_EVA      @PART_B              # to Eva mode
                      pjmp $JPORT_STBY     @STBY_HEAD           # to Stby mode
                      pjmp $JPORT_INIT     @INIT_LOOP           # to Init mode
                      jump @STBY_LOOP                           # Return

#------------------------  Standby loop ------------------------------------------------#
@STBY_LMF_T_BA_CS     setv $IREG_NULL      $NULL                # Header of Multiple frame loop

#-+-+-+-+-+-+-+-+-+- NExp 1st loop -+-+-+-+-+-+-+-+-+
                      setv $IREG_NULL      $NULL                # 1st Exp part Header
#-- PRST = ON --
                      outp &FH_SL_BA_FL_CS_PH
                      load $LREG_ROWLOOP1 $NROWLOOP             # col=1 loop [START]
@STBY_LR1_TANSPJ      outp &LF_SL_BA_FL_CS_PH
                      cjmp $LREG_ROWLOOP1 @STBY_LR1_TANSPJ      # col=1 loop [END]
                      load $LREG_ROWLOOP2 $NROWLOOP             # col=2 loop [START]
@STBY_LR2_TANSPJ      outp &F_SL_BA_FL_CS_PH
                      cjmp $LREG_ROWLOOP2 @STBY_LR2_TANSPJ      # col=2 loop [END]
                      load $LREG_COLLOOP_L $NCOL                # Column loop [START]
@STBY_LC1_TANSPJ      setv $IREG_NULL      $NULL                # Column loop Header
                      load $LREG_ROWLOOP_ODD_L $NROWLOOP        # col=2n-1 loop [START]
@STBY_LR3_TANSPJ      outp &L_SL_BA_FL_CS_PH
                      cjmp $LREG_ROWLOOP_ODD_L @STBY_LR3_TANSPJ  # col=2n-1 loop [END]
                      load $LREG_ROWLOOP_EVEN_L $NROWLOOP       # col=2n loop [START]
@STBY_LR4_TANSPJ      outp &N_SL_BA_FL_CS_PH
                      cjmp $LREG_ROWLOOP_EVEN_L @STBY_LR4_TANSPJ # col=2n loop [END]
                      cjmp $LREG_COLLOOP_L @STBY_LC1_TANSPJ     # Column loop [END]
                      outp &FT_SL_BA_FL_CS_PH

                     load $LREG_COLLOOP_L $NCOL                 # Column loop [START]
@STBY_LC1_TANSPJ      setv $IREG_NULL      $NULL                # Column loop Header
                      load $LREG_ROWLOOP_ODD_L $NROWLOOP        # col=2n-1 loop [START]
@STBY_LR3_TANSPJ      outp &L_SL_BA_FL_CS_PH
                      cjmp $LREG_ROWLOOP_ODD_L @STBY_LR3_TANSPJ  # col=2n-1 loop [END]
                      load $LREG_ROWLOOP_EVEN_L $NROWLOOP       # col=2n loop [START]
@STBY_LR4_TANSPJ      outp &N_SL_BA_FL_CS_PH
                      cjmp $LREG_ROWLOOP_EVEN_L @STBY_LR4_TANSPJ # col=2n loop [END]
                      cjmp $LREG_COLLOOP_L @STBY_LC1_TANSPJ     # Column loop [END]
                      outp &FT_SL_BA_FL_CS_PH
```

図 44　クロックソースファイルの例 (ページ 2)

```
#-+-+-+-+-+-+-+-+-+- NExp loop -+-+-+-+-+-+-+-+-+-+
                     load $LREG_NEXPLOOP  $NNEXP               # NExp Loop [START]
@STBY_LNX_T_BA_CS    setv $IREG_NULL      $NULL                # Exp part Header

#-- PRST = ON --
                     outp &FH_SL_BA_FL_CS_PH
                     load $LREG_ROWLOOP1 $NROWLOOP             # col=1 loop [START]
@STBY_LR1_TANSPC     outp &LF_SL_BA_FL_CS_PH
                     cjmp $LREG_ROWLOOP1 @STBY_LR1_TANSPC      # col=1 loop [END]
                     load $LREG_ROWLOOP2 $NROWLOOP             # col=2 loop [START]
@STBY_LR2_TANSPC     outp &F_SL_BA_FL_CS_PH
                     cjmp $LREG_ROWLOOP2 @STBY_LR2_TANSPC      # col=2 loop [END]
                     load $LREG_COLLOOP_L $NCOL                # Column loop [START]
@STBY_LC1_TANSPC     setv $IREG_NULL      $NULL                # Column loop Header
                     load $LREG_ROWLOOP_ODD_L $NROWLOOP        # col=2n-1 loop [START]
@STBY_LR3_TANSPC     outp &L_SL_BA_FL_CS_PH
                     cjmp $LREG_ROWLOOP_ODD_L @STBY_LR3_TANSPC  # col=2n-1 loop [END]
                     load $LREG_ROWLOOP_EVEN_L $NROWLOOP       # col=2n loop [START]
@STBY_LR4_TANSPC     outp &N_SL_BA_FL_CS_PH
                     cjmp $LREG_ROWLOOP_EVEN_L @STBY_LR4_TANSPC # col=2n loop [END]
                     cjmp $LREG_COLLOOP_L @STBY_LC1_TANSPC     # Column loop [END]
                     outp &FT_SL_BA_FL_CS_PH

                     cjmp $LREG_NEXPLOOP  @STBY_LNX_T_BA_CS    # NExp Loop [END]

                     retn $STACK_GIMG                          # Return

#--------------------  Data acquisition mode Main-routin  ---------------------------------#
@DATACQ              setv $IREG_MODE      $MODE_ACQ            # mode -> Acq
                     setv $IREG_PINIT     $ON                  # Frame Pointer Reset
                     subj $STACK_GIMG     @STBY_LMF_T_BA_CS    # Get Image (Through, Stare)
                     load $LREG_CHPLOOP   $NCHOP               # ChopN Loop [START]
@DATACQ_CHP          subj $STACK_GIMG     @DACQ_LMF_S_BA_CS    #   Get Image (Stored, Stare)
                     pjmp $JPORT_STBY     @STBY_HEAD           #   to Stby mode
                     cjmp $LREG_CHPLOOP   @DATACQ_CHP          # ChopN Loop [END]
                     outp &ACQEND_CS_BA                        # ACQ mode END (Stare)
                     jump @STBY_HEAD                           # Return

#------------------------  Data acquisition loop  ----------------------------------------#
@DACQ_LMF_S_BA_CS    setv $IREG_NULL      $NULL                # Header of Multiple frame loop

#-+-+-+-+-+-+-+-+-+- NExp 1st loop -+-+-+-+-+-+-+-+-+-+
                     setv $IREG_NULL      $NULL                # 1st Exp part Header
#-- PRST = ON --
                     outp &FH_SH_BA_FH_CS_PH
                     load $LREG_ROWLOOP1 $NROWLOOP             # col=1 loop [START]
@DACQ_LR1_SAFSPJ     outp &LF_SH_BA_FH_CS_PH
                     cjmp $LREG_ROWLOOP1 @DACQ_LR1_SAFSPJ      # col=1 loop [END]
                     load $LREG_ROWLOOP2 $NROWLOOP             # col=2 loop [START]
@DACQ_LR2_SAFSPJ     outp &F_SH_BA_FL_CS_PH
                     cjmp $LREG_ROWLOOP2 @DACQ_LR2_SAFSPJ      # col=2 loop [END]
                     load $LREG_COLLOOP_L $NCOL                # Column loop [START]
@DACQ_LC1_SAFSPJ     setv $IREG_NULL      $NULL                # Column loop Header
                     load $LREG_ROWLOOP_ODD_L $NROWLOOP        # col=2n-1 loop [START]
@DACQ_LR3_SAFSPJ     outp &L_SH_BA_FL_CS_PH
                     cjmp $LREG_ROWLOOP_ODD_L @DACQ_LR3_SAFSPJ  # col=2n-1 loop [END]
                     load $LREG_ROWLOOP_EVEN_L $NROWLOOP       # col=2n loop [START]
@DACQ_LR4_SAFSPJ     outp &N_SH_BA_FL_CS_PH
                     cjmp $LREG_ROWLOOP_EVEN_L @DACQ_LR4_SAFSPJ # col=2n loop [END]
                     cjmp $LREG_COLLOOP_L @DACQ_LC1_SAFSPJ     # Column loop [END]
                     outp &FT_SH_BA_FH_CS_PH
```

図 45　クロックソースファイルの例 (ページ 3)

```
#-+-+-+-+-+-+-+-+-+- NExp loop -+-+-+-+-+-+-+-+-+-+-+
                      load $LREG_NEXPLOOP  $NNEXP               # NExp Loop [START]
@DACQ_LNX_S_BA_CS     setv $IREG_NULL      $NULL                # Exp part Header

#-- PRST = ON --
                      outp &FH_SH_BA_FH_CS_PH
                      load $LREG_ROWLOOP1 $NROWLOOP             # col=1 loop [START]
@DACQ_LR1_SAFSPC      outp &LF_SH_BA_FH_CS_PH
                      cjmp $LREG_ROWLOOP1 @DACQ_LR1_SAFSPC      # col=1 loop [END]
                      load $LREG_ROWLOOP2 $NROWLOOP             # col=2 loop [START]
@DACQ_LR2_SAFSPC      outp &F_SH_BA_FL_CS_PH
                      cjmp $LREG_ROWLOOP2 @DACQ_LR2_SAFSPC      # col=2 loop [END]
                      load $LREG_COLLOOP_L $NCOL                # Column loop [START]
@DACQ_LC1_SAFSPC      setv $IREG_NULL      $NULL                # Column loop Header
                      load $LREG_ROWLOOP_ODD_L $NROWLOOP        # col=2n-1 loop [START]
@DACQ_LR3_SAFSPC      outp &L_SH_BA_FL_CS_PH
                      cjmp $LREG_ROWLOOP_ODD_L @DACQ_LR3_SAFSPC  # col=2n-1 loop [END]
                      load $LREG_ROWLOOP_EVEN_L $NROWLOOP       # col=2n loop [START]
@DACQ_LR4_SAFSPC      outp &N_SH_BA_FL_CS_PH
                      cjmp $LREG_ROWLOOP_EVEN_L @DACQ_LR4_SAFSPC # col=2n loop [END]
                      cjmp $LREG_COLLOOP_L @DACQ_LC1_SAFSPC     # Column loop [END]
                      outp &FT_SH_BA_FH_CS_PH

                      cjmp $LREG_NEXPLOOP  @DACQ_LNX_S_BA_CS    # NExp Loop [END]

                      retn $STACK_GIMG                          # Return


#          +-------------------------------------------------+
#          | 0f 0e 0d 0c 0b 0a 09 08 07 06 05 04 03 02 01 00 |
#          |                                                 |
#          |  C  C  C  C  C  C  C  C  P  F  D  L  C  F  A  A |
#          |  L  L  L  L  L  L  L  L  R  S  C  S  H  R  D  D |
#          |  K  K  K  K  K  K  K  K  S  Y  L  Y  P  A  E  T |
#          |  1  1  0  0  0  0  0  0  T  N  K  N     M  N  R |
#          |  1  0  9  8  7  6  5  4     C     C     E     G |
#          |                                                 |
#          +-------------------------------------------------+
#====================  BINARY CLOCK PATTERN DATA ========================#
&TAKEOFF_BA             bit 1024 0000000000000000
                        endb

&CBINIT_BA              bit 1000 0000000000000000
                        endb

&ACQEND_CS_BA           bit  200 0000000000000100
                        endb

&L_SH_BA_FL_CS_PH       bit   40 0000000000010010
                        bit    1 0000000000010011
                        bit    9 0000000000010010
                        bit   40 0000000010010010
                        bit    1 0000000000010011
                        bit    9 0000000000010010
                        bit   40 0000000000110010
                        bit    1 0000000000110011
                        bit    9 0000000000110010
                        bit   40 0000000010110010
                        bit    1 0000000000110011
                        bit    9 0000000000110010
                        endb
```

図 46　クロックソースファイルの例 (ページ 4)

```
&F_SH_BA_FL_CS_PH       bit   40 0000000001000010
                        bit    1 0000000001000011
                        bit    9 0000000001000010
                        bit   40 0000000011000010
                        bit    1 0000000001000011
                        bit    9 0000000001000010
                        bit   40 0000000001100010
                        bit    1 0000000001100011
                        bit    9 0000000001100010
                        bit   40 0000000011100010
                        bit    1 0000000001100011
                        bit    9 0000000001100010
                        endb

&N_SH_BA_FL_CS_PH       bit   40 0000000000000010
                        bit    1 0000000000000011
                        bit    9 0000000000000010
                        bit   40 0000000010000010
                        bit    1 0000000000000011
                        bit    9 0000000000000010
                        bit   40 0000000000100010
                        bit    1 0000000000100011
                        bit    9 0000000000100010
                        bit   40 0000000010100010
                        bit    1 0000000000100011
                        bit    9 0000000000100010
                        endb

&LF_SH_BA_FH_CS_PH      bit   40 0000000001010110
                        bit    1 0000000001010111
                        bit    9 0000000001010110
                        bit   40 0000000011010110
                        bit    1 0000000001010111
                        bit    9 0000000001010110
                        bit   40 0000000001110110
                        bit    1 0000000001110111
                        bit    9 0000000001110110
                        bit   40 0000000011110110
                        bit    1 0000000001110111
                        bit    9 0000000001110110
                        endb

&FH_SH_BA_FH_CS_PH      bit  200 0000000000000000
                        endb

&FT_SH_BA_FH_CS_PH      bit  200 0000000000000010
                        endb

&LF_SL_BA_FL_CS_PH      bit   40 0000000001010000
                        bit    1 0000000001010001
                        bit    9 0000000001010000
                        bit   40 0000000011010000
                        bit    1 0000000001010001
                        bit    9 0000000001010000
                        bit   40 0000000001110000
                        bit    1 0000000001110001
                        bit    9 0000000001110000
                        bit   40 0000000011110000
                        bit    1 0000000001110001
                        bit    9 0000000001110000
                        endb

&L_SL_BA_FL_CS_PH       bit   40 0000000000010000
                        bit    1 0000000000010001
                        bit    9 0000000000010000
                        bit   40 0000000010010000
                        bit    1 0000000000010001
                        bit    9 0000000000010000
                        bit   40 0000000000110000
                        bit    1 0000000000110001
                        bit    9 0000000000110000
                        bit   40 0000000010110000
                        bit    1 0000000000110001
                        bit    9 0000000000110000
                        endb

&F_SL_BA_FL_CS_PH       bit   40 0000000001000000
                        bit    1 0000000001000001
                        bit    9 0000000001000000
                        bit   40 0000000011000000
                        bit    1 0000000001000001
                        bit    9 0000000001000000
                        bit   40 0000000001100000
                        bit    1 0000000001100001
                        bit    9 0000000001100000
                        bit   40 0000000011100000
                        bit    1 0000000001100001
                        bit    9 0000000001100000
                        endb
```

図 47　クロックソースファイルの例 (ページ 5)

```
&N_SL_BA_FL_CS_PH       bit   40 0000000000000000
                        bit    1 0000000000000001
                        bit    9 0000000000000000
                        bit   40 0000000010000000
                        bit    1 0000000000000001
                        bit    9 0000000000000000
                        bit   40 0000000000100000
                        bit    1 0000000000100001
                        bit    9 0000000000100000
                        bit   40 0000000010100000
                        bit    1 0000000000100001
                        bit    9 0000000000100000
                        endb

&FH_SL_BA_FL_CS_PH      bit  200 0000000000000000
                        endb

&FT_SL_BA_FL_CS_PH      bit  200 0000000000000000
                        endb

#=========================  Bank B  ==================================================#

#----------------------  Evacuation mode Main-routin  ------------------------------------#
@PART_B               setv $IREG_MODE      $MODE_EVA            # mode -> Eva
                      load $LREG_CHPLOOP   0                    # ChopN Loop [START]
@EVAC_CHP             subj $STACK_GIMG     @EVAC_LMF_T_BB_CS    #   Get Image (Through, Stare)
                      cjmp $LREG_CHPLOOP   @EVAC_CHP            # ChopN Loop [END]
                      pjmp $JPORT_STBY     @STBY_HEAD           # to Stby mode
                      jump @PART_B                              # Return

#-----------------------  Evacuation mode loop  ---------------------------------------#
@EVAC_LMF_T_BB_CS     setv $IREG_NULL      $NULL                # Header of Multiple frame loop

#-+-+-+-+-+-+-+-+-+- NExp 1st loop -+-+-+-+-+-+-+-+-+
                      setv $IREG_NULL      $NULL                # 1st Exp part Header

#-- PRST = ON --
                      outp &FH_SL_BB_FL_CS_PH
                      load $LREG_ROWLOOP1 $NROWLOOP             # col=1 loop [START]
@EVAC_LR1_TBNSPJ      outp &LF_SL_BB_FL_CS_PH
                      cjmp $LREG_ROWLOOP1 @EVAC_LR1_TBNSPJ      # col=1 loop [END]
                      load $LREG_ROWLOOP2 $NROWLOOP             # col=2 loop [START]
@EVAC_LR2_TBNSPJ      outp &F_SL_BB_FL_CS_PH
                      cjmp $LREG_ROWLOOP2 @EVAC_LR2_TBNSPJ      # col=2 loop [END]
                      load $LREG_COLLOOP_L $NCOL                # Column loop [START]
@EVAC_LC1_TBNSPJ      setv $IREG_NULL      $NULL                # Column loop Header
                      load $LREG_ROWLOOP_ODD_L $NROWLOOP        # col=2n-1 loop [START]
@EVAC_LR3_TBNSPJ      outp &L_SL_BB_FL_CS_PH
                      cjmp $LREG_ROWLOOP_ODD_L @EVAC_LR3_TBNSPJ  # col=2n-1 loop [END]
                      load $LREG_ROWLOOP_EVEN_L $NROWLOOP       # col=2n loop [START]
@EVAC_LR4_TBNSPJ      outp &N_SL_BB_FL_CS_PH
                      cjmp $LREG_ROWLOOP_EVEN_L @EVAC_LR4_TBNSPJ # col=2n loop [END]
                      cjmp $LREG_COLLOOP_L @EVAC_LC1_TBNSPJ      # Column loop [END]
                      outp &FT_SL_BB_FL_CS_PH

#-+-+-+-+-+-+-+-+-+- NExp loop -+-+-+-+-+-+-+-+-+-+-+
                      load $LREG_NEXPLOOP  $NNEXP               # NExp Loop [START]
@EVAC_LNX_T_BB_CS     setv $IREG_NULL      $NULL                # Exp part Header

#-- PRST = ON --
                      outp &FH_SL_BB_FL_CS_PH
                      load $LREG_ROWLOOP1 $NROWLOOP             # col=1 loop [START]
@EVAC_LR1_TBNSPC      outp &LF_SL_BB_FL_CS_PH
                      cjmp $LREG_ROWLOOP1 @EVAC_LR1_TBNSPC      # col=1 loop [END]
                      load $LREG_ROWLOOP2 $NROWLOOP             # col=2 loop [START]
@EVAC_LR2_TBNSPC      outp &F_SL_BB_FL_CS_PH
                      cjmp $LREG_ROWLOOP2 @EVAC_LR2_TBNSPC      # col=2 loop [END]
                      load $LREG_COLLOOP_L $NCOL                # Column loop [START]
@EVAC_LC1_TBNSPC      setv $IREG_NULL      $NULL                # Column loop Header
                      load $LREG_ROWLOOP_ODD_L $NROWLOOP        # col=2n-1 loop [START]
@EVAC_LR3_TBNSPC      outp &L_SL_BB_FL_CS_PH
                      cjmp $LREG_ROWLOOP_ODD_L @EVAC_LR3_TBNSPC  # col=2n-1 loop [END]
                      load $LREG_ROWLOOP_EVEN_L $NROWLOOP       # col=2n loop [START]
@EVAC_LR4_TBNSPC      outp &N_SL_BB_FL_CS_PH
                      cjmp $LREG_ROWLOOP_EVEN_L @EVAC_LR4_TBNSPC # col=2n loop [END]
                      cjmp $LREG_COLLOOP_L @EVAC_LC1_TBNSPC      # Column loop [END]
                      outp &FT_SL_BB_FL_CS_PH

                      cjmp $LREG_NEXPLOOP  @EVAC_LNX_T_BB_CS     # NExp Loop [END]

                      retn $STACK_GIMG                          # Return
```

図 48　クロックソースファイルの例 (ページ 6)

```
#          +-------------------------------------------------+
#          | 0f 0e 0d 0c 0b 0a 09 08 07 06 05 04 03 02 01 00 |
#          |                                                 |
#          |  C  C  C  C  C  C  C  C  P  F  D  L  C  F  A  A |
#          |  L  L  L  L  L  L  L  L  R  S  C  S  H  R  D  D |
#          |  K  K  K  K  K  K  K  K  S  Y  L  Y  P  A  E  T |
#          |  1  1  0  0  0  0  0  0  T  N  K  N     M  N  R |
#          |  1  0  9  8  7  6  5  4     C     C     E     G |
#          |                                                 |
#          +-------------------------------------------------+
#====================  BINARY CLOCK PATTERN DATA ========================#
&LF_SL_BB_FL_CS_PH      bit   40 0000000001010000
                        bit    1 0000000001010001
                        bit    9 0000000001010000
                        bit   40 0000000011010000
                        bit    1 0000000001010001
                        bit    9 0000000001010000
                        bit   40 0000000001110000
                        bit    1 0000000001110001
                        bit    9 0000000001110000
                        bit   40 0000000011110000
                        bit    1 0000000001110001
                        bit    9 0000000001110000
                        endb

&L_SL_BB_FL_CS_PH       bit   40 0000000000010000
                        bit    1 0000000000010001
                        bit    9 0000000000010000
                        bit   40 0000000010010000
                        bit    1 0000000000010001
                        bit    9 0000000000010000
                        bit   40 0000000000110000
                        bit    1 0000000000110001
                        bit    9 0000000000110000
                        bit   40 0000000010110000
                        bit    1 0000000000110001
                        bit    9 0000000000110000
                        endb

&F_SL_BB_FL_CS_PH       bit   40 0000000001000000
                        bit    1 0000000001000001
                        bit    9 0000000001000000
                        bit   40 0000000011000000
                        bit    1 0000000001000001
                        bit    9 0000000001000000
                        bit   40 0000000001100000
                        bit    1 0000000001100001
                        bit    9 0000000001100000
                        bit   40 0000000011100000
                        bit    1 0000000001100001
                        bit    9 0000000001100000
                        endb

&N_SL_BB_FL_CS_PH       bit   40 0000000000000000
                        bit    1 0000000000000001
                        bit    9 0000000000000000
                        bit   40 0000000010000000
                        bit    1 0000000000000001
                        bit    9 0000000000000000
                        bit   40 0000000000100000
                        bit    1 0000000000100001
                        bit    9 0000000000100000
                        bit   40 0000000010100000
                        bit    1 0000000000100001
                        bit    9 0000000000100000
                        endb

&FH_SL_BB_FL_CS_PH      bit  200 0000000000000000
                        endb

&FT_SL_BB_FL_CS_PH      bit  200 0000000000000000
                        endb

#========================  Command data END  ============================================#
```

図 49　クロックソースファイルの例 (ページ 7)

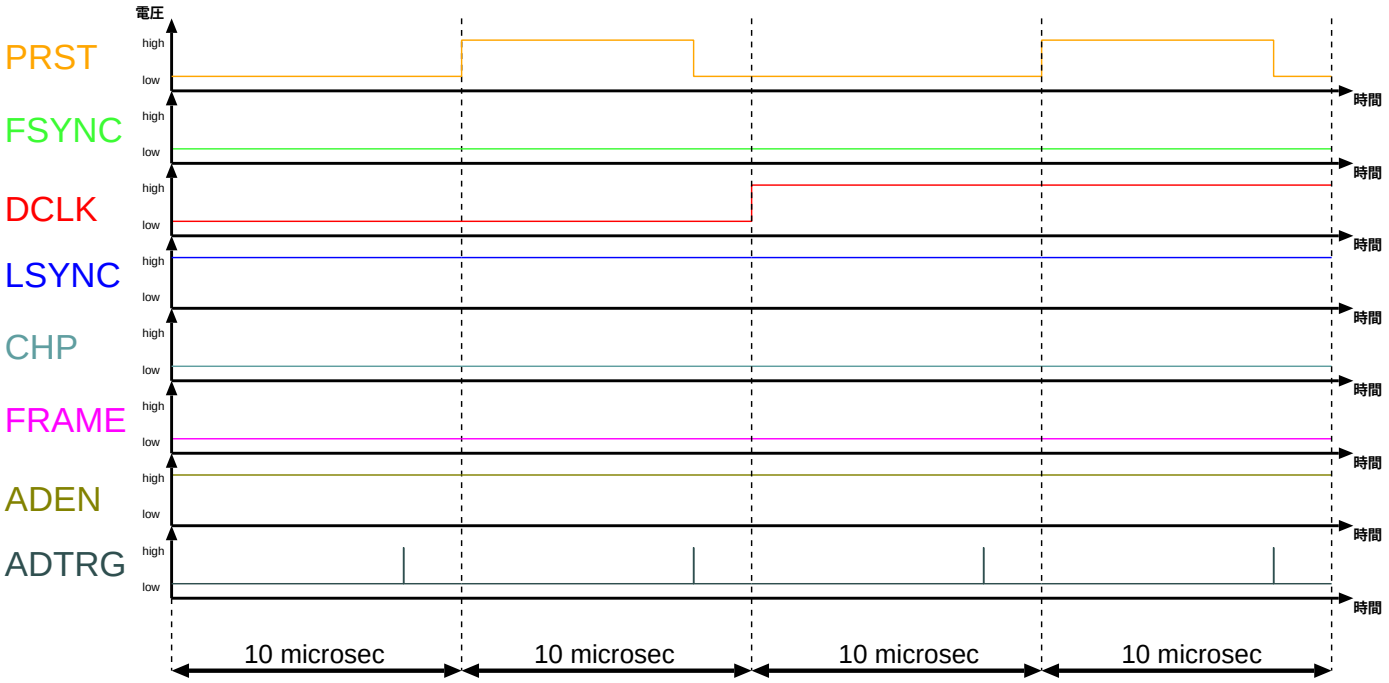

図 50　ビットデータとクロックパターンの関係 (図 46 49~61 行目の例の場合)

3.3.2　clkgen

clkgen プログラムは 14 個のパラメータを指定することでクロックソースファイルを生成し、cpc プログラムを用いてクロックバイナリファイルも生成する。表 8 に clkgen プログラムで指定するパラメータとその内容を示す。また BPP、WRow、WCol、PP、ND パラメータによって積分時間は以下の式で決まる。

- 1 Row Readout Time:1 行にアクセスするのにかかる時間 (μsec)

$$\left(0.2 \times \mathrm{BPP} \times \frac{\mathrm{WRow}}{4}\right) + \left(0.8 \times \frac{(128 - \mathrm{WRow})}{4}\right) \tag{8}$$

- 1 Cycle Time:全 pixel にアクセスするのにかかる時間 (μsec)

$$((1\mathrm{RowReadoutTime}) \times \mathrm{WCol}) + \left(0.8 \times 128 \times \frac{(128 - \mathrm{WCol})}{4}\right) + (0.8 \times \mathrm{BPP}) \tag{9}$$

- 1 Exposure Integration Time:1 回の積分時間 (μsec)

$$\frac{\mathrm{PP}}{\mathrm{BPP}} \times \left\{((1\mathrm{CycleTime}) - 0.8 \times \mathrm{BPP}) \times \frac{(65 - \mathrm{ND})}{64} + 0.8 \times \mathrm{BPP}\right\} \tag{10}$$

clkgen プログラムは各パラメータと生成されるクロックソースファイル、クロックバイナリファイルの保存先を入力して使用する。図 51 に clkgen コマンドの使用例を示す。

パラメータ	内容
BPP	ピクセルアクセス時間 [MIRSIS 撮像検出器では DCLK の high 又は low の継続時間 (2.2.2 参照)、1BPP あたり 0.2μsec、20 の倍数]
PP	BPP と組み合わせて積分時間を決める (式 (10) 参照) [BPP の整数倍]
ND	積分時間を操作するパラメータ (1 ピクセルをリセットしている時間の割合)[1∼64] ※ (65 - ND)/64 で ND=1 の場合の割合を 100% と定義し、ND を操作することで ND=1 のときの積分時間に対する割合から積分時間を操作する
NExp	1 チョップループの中でチョップビームの片ビームごとに撮影する画像の枚数
Wipe	CoAdd モードを使用する際に 1 チョップループの中でチョップビームの片ビームごとに画像を捨てる枚数
ChopL	チョップのループ数 (ON/OFF を何サイクル行うか決める)
ChopB	チョップビーム数 (通常は ON 点/OFF 点の 2 点を観測するので 2 を設定する)
CoAdd	CoAdd モードの有無を選択 ※ CoAdd モード:撮影した画像を NExp で指定した分の枚数だけ 1 枚に足し合わせた画像にして出力する
Chop_Mode	チョップモードを選択 (0:チョッピングオフ、1:チョッピングオン)
Clk_Typ	クロックタイプの設定 (クロックタイプは 0∼5 まで用意されている) クロックタイプ 0: クロックタイプ 2 で得られるデータを検出器の pixel と同じ配置に並べ直して画像を出力 (天体観測用) クロックタイプ 1: クロックタイプ 0 から ADC ボードで得られた電圧をもとにノイズを引いた画像を出力 クロックタイプ 2: (クロックタイプ 3-クロックタイプ 4) の画像を出力 クロックタイプ 3: 4 つの出力端子から読み出される検出器で積分された電圧 (図 21 V_{int}) をもとに 128×128pixel の fits 画像を縦に 4 分割し出力端子ごとの情報を fits 画像として出力 クロックタイプ 4: 4 つの出力端子から読み出される検出器でリセットが起きた後の電圧 (図 21 V_{rr}) をもとに 128×128pixel の fits 画像を縦に 4 分割し出力端子ごとの情報を fits 画像として出力 クロックタイプ 5: リファレンス電圧 V_{REF} から得られる情報を fits 画像として出力
SRow	Row の読み出し開始 pixel を設定 [1∼128]
SCol	Column の読み出し開始 pixel を設定 [1∼128]
WRow	Row の読み出し幅を設定 [<(128-SRow)]
WCol	Column の読み出し幅を設定 [<(128-SCol)]

表 8　clkgen プログラムで指定するパラメータと内容

```
$ pwd
/home/mirsis/mirsis/tac/clk
$ clkgen 100 100 1 10 0 1 1 0 0 0 1 1 128 128 /home/mirsis/data/clock

            --- bank A ---          --- bank B ---
 bit:   006478/500000(0.5%)  006478/500000(0.5%)
 com:  000079/040000(0.0%)  00002E/040000(0.0%)
 binary file size = 24.00 Mbyte.
```

図 51　クロックパラメータが BPP:100 PP:100 ND:1 NEx:10 Wip:0 ChL:1 ChB:1 CoAdd:0 Chop_Mode:0 Clk_Typ:0 SRow:1 SCol:1 WRow:128 WCol:128 で/home/mirsis/data/clock にクロックソースファイル、クロックバイナリファイルを保存する場合の clkgen コマンドの使用例

3.4 DIO ボードの制御 (dio ディレクトリ)

dio ディレクトリには、DIO ボード (Interface 2772c ボード) を制御するプログラム (表 9) が存在する。これらを用いて DIO ボードを使用する際のボード初期化、クロックの操作を行うクロックプロセッサ (CLP) やデータ取得操作を行うフレームプロセッサ (FRP) の制御、積分の開始/終了の制御を行う。

ファイル名	内容
init_dio	2772c ボードの初期化
meclear	ワーク領域メモリの初期化
load_cp	バイナリクロックデータを CLP メモリにロード、ロードされたクロック情報のステータスファイルへの書き込み
jport	jport の設定
go	CLP/FRP プロセッサの開始
abort	CLP/FRP プロセッサの強制終了
break	積分の強制終了
acqdat	現在ロードしているクロックを使用して積分を開始、フレームデータを取得

表 9　dio ディレクトリ内に存在する主なファイルとその内容

3.5 BIAS 制御 (bis ディレクトリ)

マルチプレクサに与えるバイアス電圧値の設定と出力 switch の制御を行う。バイアス値はハードウェアによる制限とソフトウェアによる設定で決まる。ここではハードウェアとソフトウェアによるバイアス電圧値の設定方法、バイアス電圧出力の制御方法について述べていく。

3.5.1 ハードウェアによるバイアス値の制限

検出器に与えるバイアス電圧はハードウェアによって制限している。ソフトウェアによってコントロールする DDAC 値 (12bit) をデジタルアナログコンバータ (DAC) に入力し、この値とリファレンス電圧により決まる電圧が DAC から出力される。その後、attenuator による減衰とオペアンプによる増幅で任意の範囲でバイアス電圧が出力できる (図 52)。DDAC、DACREF 電圧、抵抗 R_1、抵抗 R_2、抵抗 R_f、抵抗 R_s によって出力電圧 VOUT は以下の式で決まる。

- DAC

$$\mathrm{VDAC}' = \frac{\mathrm{DACREF} \times \mathrm{DDAC}}{4096}$$

- Attenuator

$$\mathrm{VDAC} = \frac{R_2}{R_1 + R_2} \times \mathrm{VDAC}'$$

- Amplifier

$$\mathrm{VOUT} = \frac{R_f + R_s}{R_s} \times \mathrm{VDAC} - \frac{R_f}{R_s} \times \mathrm{VOFF}$$

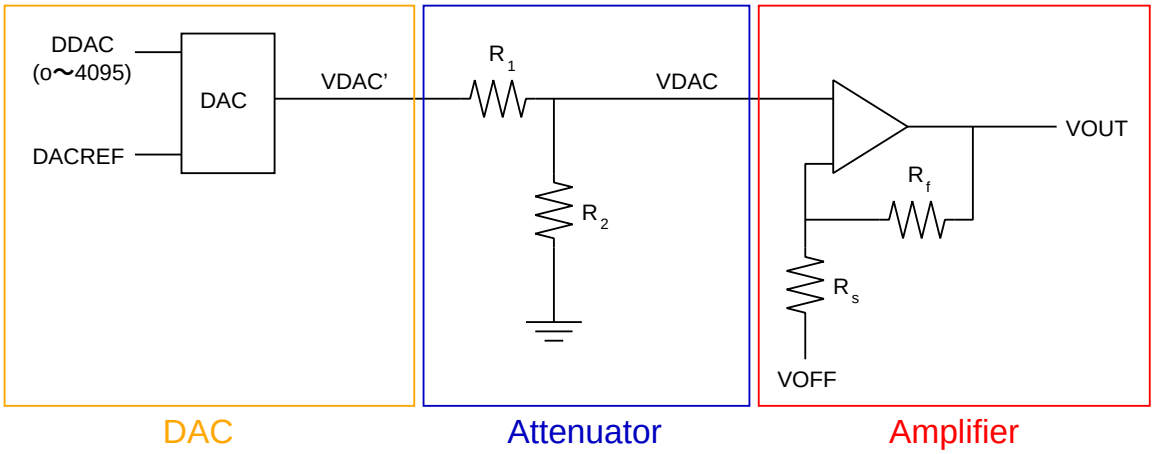

図 52　バイアス電圧生成回路

3.5.2　ソフトウェアによる設定

bis/src にある header ファイル (bias.h.SiAs) の内容を変更することで、DAC へ与える DDAC 値を設定する。DDAC 値は 0〜4095 の範囲で指定が可能で、ハードウェアと合わせて任意のバイアス電圧になるように設定する必要がある。header ファイル (図 53) を編集した後に make することで、設定した DDAC 値が DAC へ出力されるようになる。

```
/* 内部バイアス電圧(=内部生成電圧) */
/*  TYPE, NAME, MUXCH, DACH, OSWCH, V, I, DDAC, OSWORD, ALIAS */
static struct struct_bias btbl_data[TBL_NUM] = {
  { "BIAS", "BIAS0" ,14, 0 , 0 , 7.7,    0, 3000,  2, "S_RON" },
  { "BIAS", "BIAS1" ,0 , 1 , 1 , 7.7,    0, 3050,  1, "VNWELL"},
  { "BIAS", "BIAS2" ,9 , 2 , 2 , 0.0,    0,    0,  0, "--"    },
  { "BIAS", "BIAS3" ,15, 3 , 3 , 0.0,    0,    0,  0, "--"    },
  { "BIAS", "BIAS4" ,1 , 4 , 4 , 5.5,    0,  350,  6, "AON"   },
  { "BIAS", "BIAS5" ,7 , 5 , 5 , 5.0,    0, 3490, 10, "VRST"  },
  { "BIAS", "BIAS6" ,6 , 6 , 6 , 3.0,    0,  700, 13, "VLG"   },
  { "BIAS", "BIAS7" ,8 , 7 , 7 , 4.0,    0, 2150, 11, "VDI"   },
//  { "BIAS", "BIAS8" ,30, 8 , 8 , 2.0,       ,  72 , 12, "VDSUB" },  /* VDSUB = 2.0V */
  { "BIAS", "BIAS8" ,30, 8 , 8 , 2.0,    0, 720,  2, "VDSUB" },  /* VDSUB = 2.0V */
//{ "BIAS", "BIAS8" ,30, 8 , 8 , 3.0,    0, 2100, 12, "VDSUB" },  /* VDSUB = 3.0V */
  { "BIAS", "BIAS9" ,24, 9 , 9 , 2.0,    0,  700, 17, "VREF"  },
  { "BIAS", "BIAS10",25, 10, 10, 3.0,    0, 2100,  7, "AOFF"  },
  { "BIAS", "BIAS11",31, 11, 11, 4.0,    0, 3470,  3, "S_ROFF"},
  { "CS"  , "CSA0"  ,46, 12, 12, 4.5,  200, 2820, 14, "VDA"   },
  { "CS"  , "CSA1"  ,36, 13, 13, 3.0,   -3,  720,  5, "F_ROFF"},
  { "CS"  , "CSA2"  ,34, 14, 14, 3.6, 1120, 2900, 16, "VOD"   },
  { "CS"  , "CSA3"  ,47, 15, 15, 1.2, -200,  980, 15, "VLS"   },
  { "CS"  , "CSB0"  ,62, 16, 16, 6.7,    3, 2000,  4, "F_RON" },
  { "CS"  , "CSB1"  ,52, 17, 17, 7.3,   10, 2840,  8, "VDD"   },
  { "CS"  , "CSB2"  ,50, 18, 18, 3.0,  -10, 2080,  9, "VSS"   },
  { "CS"  , "CSB3"  ,63, 19, 19, 0.0,    0,    0,  0, "--"    },
  { "VCLK", "VCLKNO",22, 20, 22, 7.0,    0, 2050, 18, "CLKH"  },
  { "VCLK", "VCLKNC",23, 21, 22, 3.0,    0, 2040, 19, "CLKL"  },
  { "VSS" , "VSSA"  ,37, 22, 20, 0.0,    0,    0,  0, "--"    },
  { "VSS" , "VSSB"  ,53, 23, 21, 0.0,    0,    0,  0, "--"    }
};
```

各バイアスのDDAC値

図 53　バイアス電圧の DDAC 値を記述する header ファイルの一部

3.5.3 バイアス出力の制御と管理

bis ディレクトリ内の bias プログラムでバイアス出力の制御と管理を行う。このプログラムには DAC 値の設定、出力 switch の ON/OFF、バイアス値のモニタの 3 つの機能がある。DAC 値はデフォルトで 128 が設定されており、0∼255 の間で設定が可能である。DAC 値を変更することでおよそ 0.01V 単位でバイアス電圧の設定が可能である。出力されるバイアス値をモニターし、状況によっては DAC 値を 128 から変更することで目標とするバイアス電圧への微調が可能となる。

3.6 ステータスファイル管理

検出器へ与える各バイアス電圧、現在使用しているクロック、取得した画像の情報などを記述したテキストファイルをステータスファイルと呼ぶ。このステータスファイルは検出器を駆動する前に雛型を MySQL のデータベースへアップロードすることで、TAC システム内のプログラムを使用する度に内容が自動更新されていく。このアップロードは db_upfile プログラムで行う。ステータスファイルは特に bias プログラムを使用する際に重要で、bias プログラムはステータスファイルに書いてある DAC 値を読み込んで出力するバイアス値を決める。また、ステータスファイルはウェブブラウザ (http://localhost/mirsis/mirsis_status.php) を用いて確認することができる。

3.7 オペレーション

この節では、ここまで述べてきた TAC システム内の各プログラムを用いた検出器の具体的な起動手順と終了手順について説明する (図 54∼57)。検出器の起動手順として、まず初めに stat ディレクトリに移動しステータスファイルを db_upfile プログラムで MySQL へアップする。次に dev ディレクトリに移動し、RTAI、i2772c.ko、tac_core.ko モジュールをインストールする。その後 dio ディレクトリに移動し、ワークメモリ、Interface 2772c ボードの初期化を実行後、クロックバイナリファイルをロードする。ここで、ADC、DRV ボードへ電源を供給する直流電源の出力端子からそれぞれ +10V、-10V、+3.3V 出力するように設定を行った後に直流電源の output switch を ON にする。ここから bis ディレクトリに移動後、バイアス電圧の設定を行う。bias プログラムでステータスファイルに記述した DAC 値をそれぞれのバイアス電圧に設定、出力バイアス電圧値の確認を行う。次に dio ディレクトリに移動して CLP/FRP を開始し INIT_Mode へ移行、bis ディレクトリに戻ってマルチプレクサにバイアス電圧の供給を開始し、もう一度出力電圧の確認をする。最後に dio ディレクトリに戻って STBY_Mode に移行し、待機状態になって検出器の起動は終了する。ここまでの起動作業を行った後、acqdat プログラムを用いてフレームデータを取得することが可能となる。acqdat プログラムによって取得したフレームデータは、ステータスファイルに記述したディレクトリに保存される。

検出器の終了手順として、まず初めに bis ディレクトリに移動後 bias プログラムでマルチプレクサへバイアス電圧の供給を停止する。次に ADC、DRV ボードへ電源を供給していた直流電源の output switch を OFF にする。その後 dio ディレクトリに移動後 CLP/FRP を停止し、dev ディレクトリに移動して i2772c.ko、tac_core.ko、RTAI モジュールをリムーブして検出器駆動を終了する。

```
$ cd /home/mirsis/mirsis/stat
$ db_upfile /home/mirsis/status_file/status_IMDet  <--ステータスファイルを MySQL へアップ
$ cd /home/mirsis/mirsis/tac/dev
$ sudo rtai start  <--RTAI モジュールのインストール
$ sudo devload load  <--i2772c.ko、tac_core.ko モジュールのインストール
$ cd /home/mirsis/mirsis/tac/dio
$ memclear ALL  <--ワーク領域メモリの初期化
clear CLP memory.
clear FRP(ADC) memory.
clear FRP(COOK1/2) memory.
$ init_dio  <--Interface 2772c ボードの初期化
Initialize ALL functions.
$ load_cp 0 /home/mirsis/data/clock/clock.bin  <--クロックバイナリファイルをロード
Loading Clock Pattern -  done.
---ADC、DRV ボードへ電源を供給する直流電源の output switch を ON---
$ cd /home/mirsis/mirsis/bis
$ bias DAC default <--ステータスデータに記述した DAC 値をセット
$ bias MON all <--設定した電圧を確認
  ch    Name    Alias   Vset    DAC     Vint    OSW     Vout    Iset  Idet
                        (V)             (V)             (V)     (uA)  (uA)
 ------------------------------------------------------------------------
  00    BIAS0   S_RON   7.70    128 V   7.70 V  OFF     0.00 F     0     0
  01    BIAS1   VNWELL  7.70    128 V   7.70 V  OFF     0.00 F     0     0
  04    BIAS4   AON     5.50    128 V   5.50 V  OFF     0.00 F     0     0
  05    BIAS5   VRST    5.00    128 V   5.00 V  OFF     0.00 F     0     0
  06    BIAS6   VLG     3.00    128 V   3.00 V  OFF     0.00 F     0     0
  07    BIAS7   VDI     4.50    128 V   4.50 V  OFF     0.00 F     0     0
  08    BIAS8   VDSUB   2.00    128 V   2.00 V  OFF     0.00 F     0     0
  09    BIAS9   VREF    2.00    128 V   2.00 V  OFF     0.57 F     0     0
  10    BIAS10  AOFF    3.00    128 V   3.00 V  OFF     0.00 F     0     0
  11    BIAS11  S_ROFF  4.00    128 V   4.00 V  OFF     0.00 F     0     0
  12    CSA0    VDA     4.50    128 V   4.50 V  OFF     0.00 F   200     0
  13    CSA1    F_ROFF  3.00    128 V   3.00 V  OFF     0.00 F    -3     0
  14    CSA2    VOD     3.60    128 V   3.60 V  OFF     0.00 F  1120     0
  15    CSA3    VLS     1.20    128 V   1.20 V  OFF     0.00 F  -200     0
  16    CSB0    F_RON   6.70    128 V   6.70 V  OFF     0.00 F     3     0
  17    CSB1    VDD     7.30    128 V   7.30 V  OFF     0.00 F    10     0
  18    CSB2    VSS     3.00    128 V   3.00 V  OFF     0.00 F   -10     0
  20    VCLKNO  CLKH    7.00    128 V   7.00 V  OFF6    7.00 F
  21    VCLKNC  CLKL    3.00    128 V   3.00 V  OFF6    3.00 F
```

図 54　検出器の起動手順 1

```
$ cd /home/mirsis/mirsis/tac/dio
$ go <--CLP/FRP の開始
$ jport 1 <--INIT_Mode へ移行
$ cd /home/mirsis/mirsis/bis
$ bias OSW on <--マルチプレクサにバイアス電圧の供給を開始
$ bias MON all <--マルチプレクサに供給を開始したバイアス電圧を確認
  ch    Name   Alias   Vset     DAC      Vint    OSW     Vout     Iset   Idet
                          (V)              (V)             (V)     (uA)   (uA)
 ------------------------------------------------------------------------------
  00    BIAS0  S_RON   7.70    128 V   7.70 V  ON      7.71 V      0    -21
  01    BIAS1  VNWELL  7.70    128 V   7.71 V  ON      7.70 V      0     21
  04    BIAS4  AON     5.50    128 V   5.50 V  ON      5.50 V      0    -21
  05    BIAS5  VRST    5.00    128 V   5.00 V  ON      5.00 V      0      0
  06    BIAS6  VLG     3.00    128 V   3.00 V  ON      3.00 V      0      0
  07    BIAS7  VDI     4.50    128 V   4.50 V  ON      4.50 V      0      0
  08    BIAS8  VDSUB   2.00    128 V   2.00 V  ON      2.00 V      0      0
  09    BIAS9  VREF    2.00    128 V   2.00 V  ON      2.00 V      0      0
  10    BIAS10 AOFF    3.00    128 V   3.00 V  ON      3.00 V      0      0
  11    BIAS11 S_ROFF  4.00    128 V   4.00 V  ON      4.00 V      0      0
  12    CSA0   VDA     4.50    128 V   4.50 V  ON      4.49 V    200    138
  13    CSA1   F_ROFF  3.00    128 V   3.00 V  ON      3.00 V     -3      0
  14    CSA2   VOD     3.60    128 V   3.60 V  ON      3.48 V   1120   1102
  15    CSA3   VLS     1.20    128 V   1.20 V  ON      1.22 V   -200   -161
  16    CSB0   F_RON   6.70    128 V   6.70 V  ON      6.70 V      3      0
  17    CSB1   VDD     7.30    128 V   7.30 V  ON      7.30 V     10     46
  18    CSB2   VSS     3.00    128 V   3.00 V  ON      3.00 V    -10    -23
  20    VCLKNO CLKH    7.00    128 V   7.00 V  ON      7.00 V
  21    VCLKNC CLKL    3.00    128 V   3.00 V  ON      3.00 V
$ cd /home/mirsis/mirsis/tac/dio
$ jport 2 <--STBY_Mode へ移行
```

図 55　検出器の起動手順 2

```
$ acqdat
RAW|STARE, chop-frequency:0.00Hz
integ-time:4.10sec x 1beams
exec-time:4.26sec
Now getting data...   reading.. done.
```

図 56　フレームデータの取得

```
$ cd /home/mirsis/mirsis/tac/bis
$ bias OSW off  <--マルチプレクサへバイアス電圧の供給を停止
---ADC、DRV ボードへ電源を供給する直流電源の output switch を OFF---
$ cd /home/mirsis/mirsis/tac/dio
$ abort <--CLP/FRP の停止
$ cd /home/mirsis/mirsis/tac/dev
$ sudo devload load  <--i2772c.ko、tac_core.ko モジュールのリムーブ
$ sudo rtai stop  <--RTAI モジュールのリムーブ
```

図 57　検出器の終了手順

4 MIRSIS 撮像検出器 Si:As 128×128 array 検出器駆動試験

検出器系の試験は、以下に示す手順で行う。

1. ADC ボード、DRV ボードの駆動、特性測定試験
2. TAC システムを用いたクロック出力試験
3. Function Generator、TAC システムを用いた画像取得試験
4. マルチプレクサの駆動試験
5. 検出器の駆動試験

また各試験を行う際には毎回、ケーブルを全て接続した状態で導通チェックを行う。MIRSIS 撮像検出器を立上げるにあたって、1 の試験は済んだボードを使用した。本章では上記のうち 4、5 の試験 (それぞれ 4.1、4.2 参照) について述べていく (1∼3 の試験については後章参照)。

4.1 Si:Sb 128×128 マルチプレクサ駆動試験

4.1.1 試験目的

検出器は低温下 (<10K) でのみ駆動が可能だが、マルチプレクサは常温 (∼300K) でも駆動が可能なので、検出器に比べ試験が容易である。また、マルチプレクサは可視光に感受性があるので、光への反応も確認できる。故に、マルチプレクサ駆動試験は検出器駆動試験を行う前の重要な試験である。

前章までに述べた、MIRSIS 検出器読み出し用 TAC システム (ADC、DRV 等の各ボード、計算機) の駆動試験として、このシステムを用いた MIRSIS 撮像検出器で使用する Si:As 128×128 array 検出器と同じ pixel 数の Si:Sb 128×128 array 検出器のマルチプレクサの駆動実験を行った。また、あるクロック下でのマルチプレクサから ADC ボードへ送られてくる Analog データを、クロック出力と合わせて確認する。この試験を通して、実際のボードの接続、バイアス電圧の制御、クロック出力の確認、使用ソフトウェアの使い方を学び、MIRSIS 検出器の駆動に役立てた。

4.1.2 試験環境

検出器試験用真空容器を用いて常温 (T∼300K) および低温 (T∼4K) で試験を行う。マルチプレクサの制御にはリアルタイム Linux OS 計算機と TAC システムを用いる。検出器へのクロック電圧とバイアス電圧を送る配線を図 58 に示す。フロントエンド回路 (ADC、DRV ボード) から真空容器までは DIN96 コネクタ (京セラエルコ、60-8455-396-410-000) と KPT06F20-41S コネクタを結線した特注のハーネスを用いる。容器内部では常温の熱を遮断するため熱伝導率の低いマンガニン線を用いた後、銅プリント基板ケーブル (図 59 左) に変換する。このケーブルを、ラディエーションシールド (∼30K まで冷却) に固定した銅コアのサーマルアンカ基板 (図 60) に接続する。熱伝導率の良い銅を用いたサーマルアンカ基板は、ケーブルを介して外部から侵入した熱流入をラディエーションシールドへ逃す役割を持つ。サーマルアンカ基板からラディエーションシールドと検出器箱の間の熱流を遮断するためにマンガニンプリント基板ケーブル (図 59 右) を使用し、光学定盤に固定されたマルチプレクサと FOB を内蔵した検出器箱 (∼4K まで冷却) へ接続する。この試験では、MAX38 で使用していた検出器箱を転用して使用した。

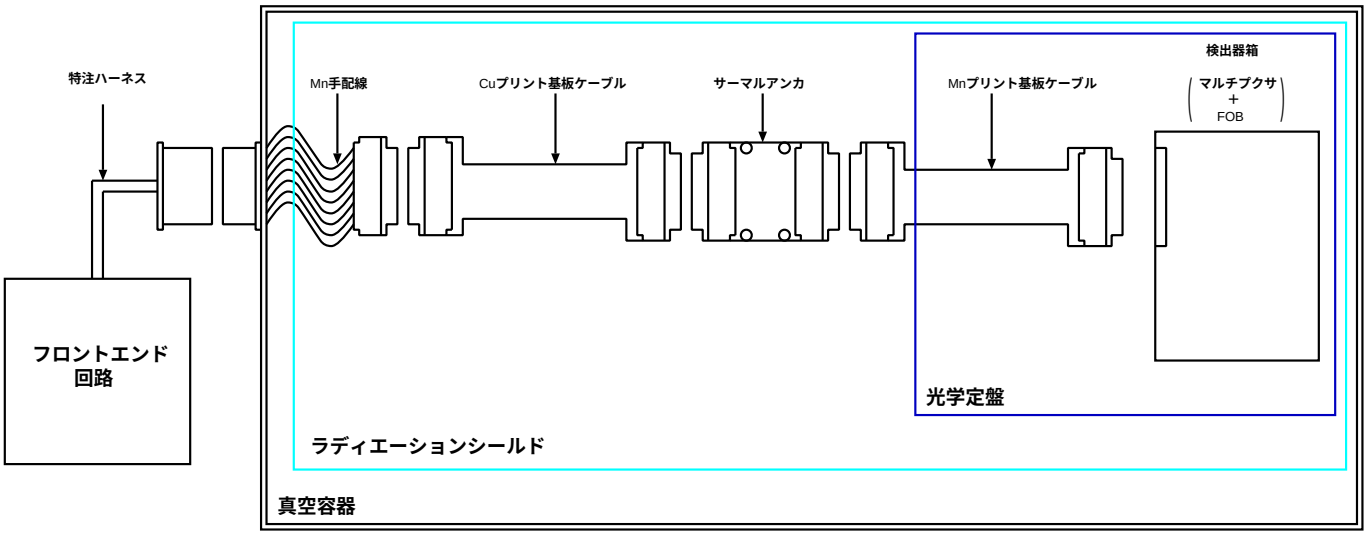

図 58　マルチプレクサ駆動試験時の配線概略図

図 59　プリント基板ケーブル (左:銅、右:マンガニン)

図 60　サーマルアンカ基板

4.1.3 試験方法

初めに常温で検出器箱の窓を閉じ、マルチプクサの駆動を確認するための画像を取得する。次に検出器箱の蓋を開閉することで、マルチプレクサの可視光への反応を確認するための画像を取得する。このとき、マルチプレクサから ADC ボードに送られてくる 4 本の出力端子の中で出力端子 1 からの Analog データ (ADC ボードの TPJA2 点) と 3 つのクロック (FSYNC、LSYNC、DCLK) の関係をオシロスコープを用いて確認する。この確認を行う際、図 61 で示した点にオシロスコープのプローブをあてる。可視光への反応を確認した後、検出器箱と真空容器の蓋を全て閉じ、真空引きと冷却を行う。マルチプレクサが極低温 (∼4K) まで冷却され温度が一定になった後、極低温下におけるマルチプクレサの駆動を確認するための画像を取得し冷却を終了する。

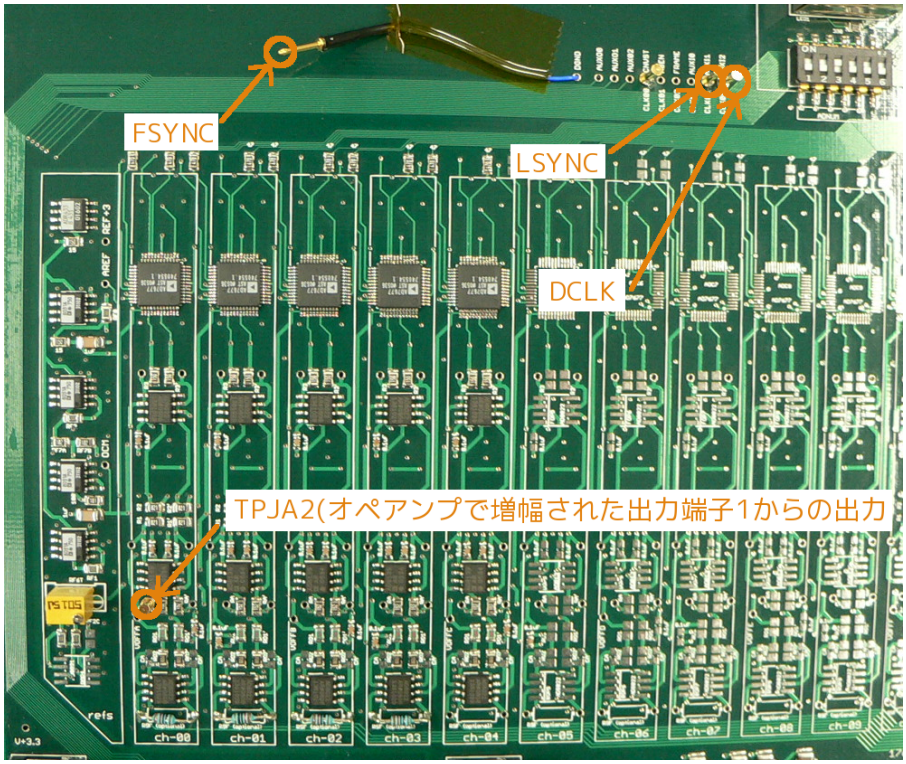

図 61　ADC ボード上でクロックと Analog データを確認するための測定点

4.1.4 実験結果

1. 駆動確認のための TAC システムを用いた画像取得試験

   マルチプレクサの駆動を確認するために行った画像取得試験の結果を見ていく。図 62~65 は全てクロックタイプ 3 を用いて撮影した fits 画像である。図 62 は常温で検出器箱の蓋を閉じ真空容器の蓋は開けた状態で撮影した fits 画像である。過去の試験で 4 本ある出力端子のうちの 1 本 (出力端子 2) が破損したので、この出力端子からは出力がない。しかし他の出力端子からは、マルチプレクサの熱雑音による出力が確認できる。図 63 は図 62 の状態から検出器箱の蓋を開け、マルチプレクサに直接可視光を当てた状態で撮影した fits 画像である。カウントが図 62 のカウントよりかなり上昇しており、可視光によってマルチプクサ内に発生した電荷が UNIT CELL の積分コンデンサに蓄積され、出力端子 2 を除く端子から出力が発生したのが確認できる。図 64 は冷却を行う前に、常温で検出器箱と真空容器の蓋を閉めた状態で撮影した fits 画像である。図 62 と同様に、熱雑音による出力が確認できる。図 65 は図 64 の状態から約 4K まで冷却し、撮影した fits 画像である。マルチプレクサを冷却することで全体的に出力が低くなり、図 64 と比較すると pixel 間で出力の差がほとんどなく、カウント値が平らな画像になることが分かる。

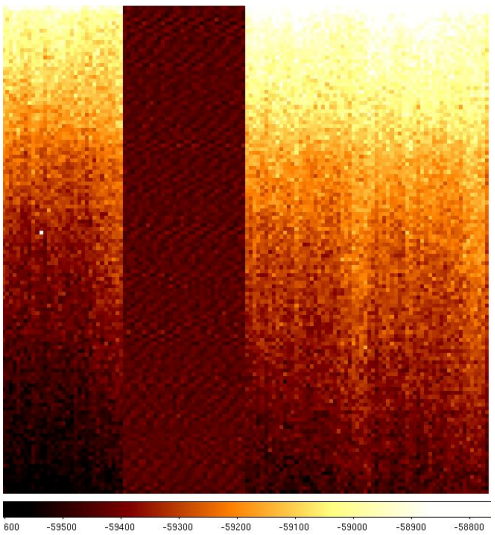

図 62　マルチプレクサの暗電流による出力 (常温、検出器箱の蓋は閉じているが、真空容器の蓋は開いている状態)

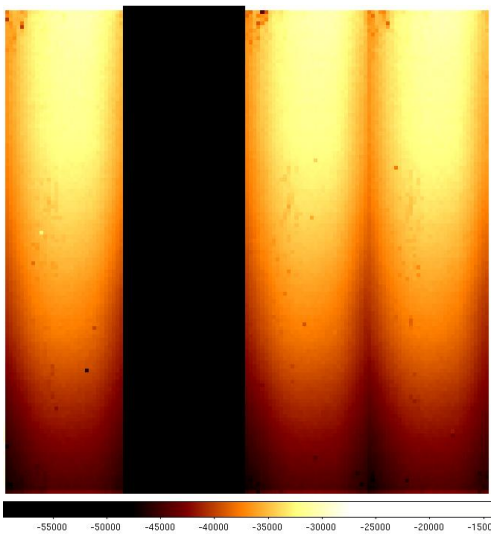

図 63　マルチプレクサに可視光を当てたときの出力 (常温、検出器箱と真空容器の蓋が開いて光が入っている状態)

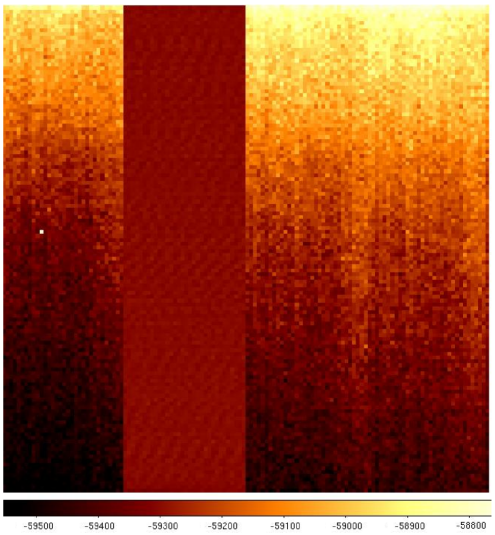

図 64　マルチプレクサの暗電流による出力 (常温、検出器箱と真空容器の蓋が閉じている状態)

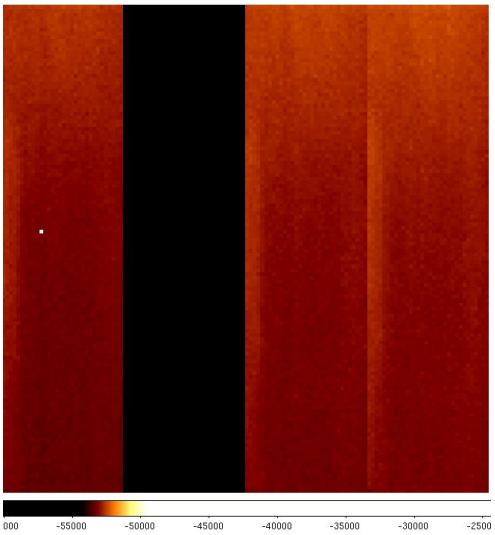

図 65　マルチプレクサの暗電流による出力 (低温、検出器箱と真空容器の蓋が閉じている状態)

2. クロックとマルチプレクサの Analog データの確認

クロックパラメータが BPP:100 PP:100 ND:1 NEx:10 Wip:0 ChL:1 ChB:1 CAd:0 CkT:3 SRw:1 SCL:1 WRw:128 WCl:128 のクロックを用いて、FSYNC、LSYNC、DCLK クロック対する出力端子 1 のオフセット付非反転増幅器によって増幅された Analog データの出力を確認する (クロックについては 2.2.2 を参照)。出力端子 1 からの出力 V_{out} は図 66 に示す回路によって増幅され、TPJA2 点から V_{out}^{amp} が出力される。Si:Sb 128×128 array マルチプクサと Si:As 128×128 array マルチプクサは極低温 (<10K) まで冷却すると、各出力端子からリセット状態で 2.7±0.1V、積分コンデンサのコンデンサ容量の最大まで電荷を積分した状態で 1.3V の電圧を出力する (Si:Sb 128×128 array 検出器マニュアルより)。これらの電圧がオフセット付非反転増幅器回路によって増幅されると、TPJA2 点では以下の出力が予想される。

$$\text{リセット時：}\ 2.7\times(1.00+3.65)/1.00-2.4\times3.65/1.00=3.8(\text{V})$$
$$\text{最大積分時：}\ 1.3\times(1.00+3.65)/1.00-2.4\times3.65/1.00=-2.7(\text{V})$$

上記の予想値と Analog データの実測値を 1Frame、1Line、1Pixel にそれぞれ焦点を当てて確認していく。

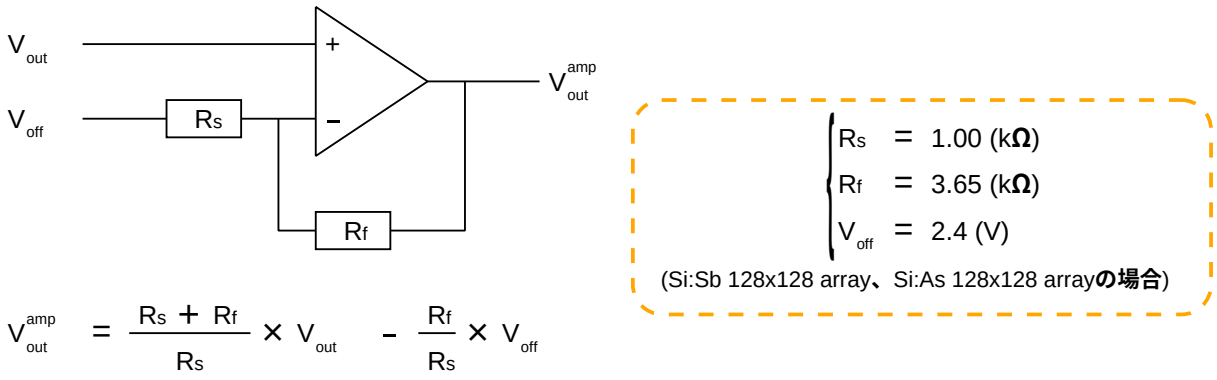

図 66　オフセット付き非反転増幅器を用いた出力端子 1 の出力増幅回路

(a) 1Frame の確認

図 67、図 68 は以下の設定でオシロスコープ出力を確認したときの様子である。

- ch1(黄色)…FSYNC [5V/div,position:+3div]
- ch2(緑色)…LSYNC [5V/div,position:+2div]
- ch4(青色)…マルチプレクサの出力端子 1 の Analog データ [2V/div,position:-2div]
- time:10 [msec/div]
- delay:41.00 [msec]

(ch…オシロスコープの出力チャンネル)

1frame の中で LSYNC は 64 周期の矩形波を持ち、1 周期で 2line へのアクセスを行うことで 2 × 64 = 128line = 1frame へのアクセスを行う (図 22 参照)。また ND が 1 なので、FSYNC が 1Frame の中で LSYNC1 周期分のみ high になる。図 67 は検出器箱の蓋を閉じた状態で、マルチプレクサを駆動したときの各端子の電圧をモニタしたオシロスコープの画像である。ここではマルチプレクサを double sampling mode で読み出しているため、積分電圧とリセット電圧 (図 21 参照) がオフセット付き非反転増幅器で増幅されたものが出力端子 1 から交互に出力される。図 67 はダーク状態なので、積分電圧 (ここで用いた積分時間は 81msec) がリセット電圧とほとんど変わらないため、出力端子 1 の電圧は全体的に平坦である。出力電圧は約 4.0V と、先ほどの予想である 3.8V より約 0.2V 高い。しかし、先ほどの計算結果は低温での出力であり、低温と常温では抵抗の抵抗値などが変わってくるので UNIT CELL の増幅率が変わってくる。故に、今回はマルチプレクサを常温で駆動しているため、出力に違いが出たと考えられる。また、一部で hotpixel のような出力が見られた。この pixel は何らかの形で壊れていると考えられる。また、Frame と Frame の間でクロックが休止している間に約-2.7V の出力 V_{out}^{amp} があることが分かった。故に、マルチプレクサにクロックが与え

られていない間、出力端子から以下の計算で得られる出力 V_{out} が出力されていることが判明した。

$$\begin{aligned} V_{out} &= \left(V_{out}^{amp} + \frac{R_f}{R_s} V_{off} \right) \frac{R_s}{R_s + R_f} \\ &= (-2.7 + \frac{3.65}{1.00} \cdot 2.4) \frac{1.00}{1.00 + 3.65} \\ &\sim 1.3(V) \end{aligned}$$

図 68 は検出器箱の蓋を開け、マルチプレクサに可視光を当てて駆動したときの各端子電圧をモニタしたオシロスコープの画像である。図 67 と比較すると出力端子 1 の出力に大きな変化があるのが確認できる。これはマルチプクサが可視光を受けて発生した電荷が積分コンデンサに積分され、図のような出力が各 pixel から発生した結果である。

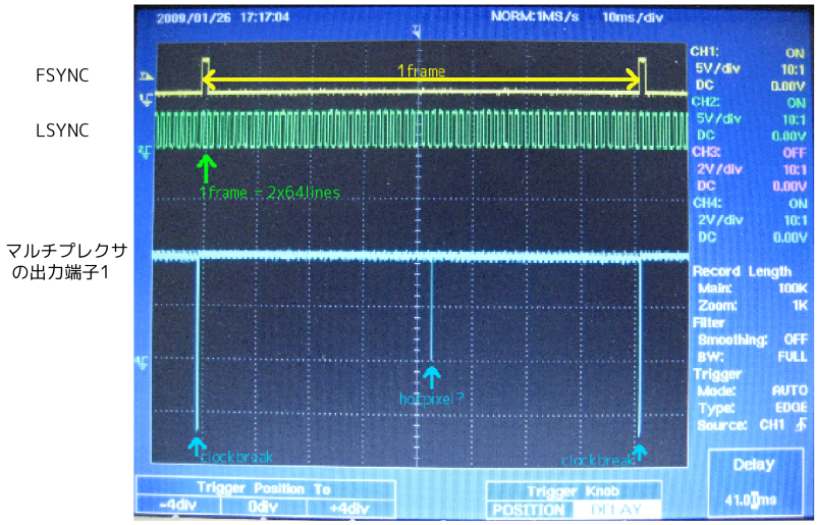

図 67　1frame の Analog データ

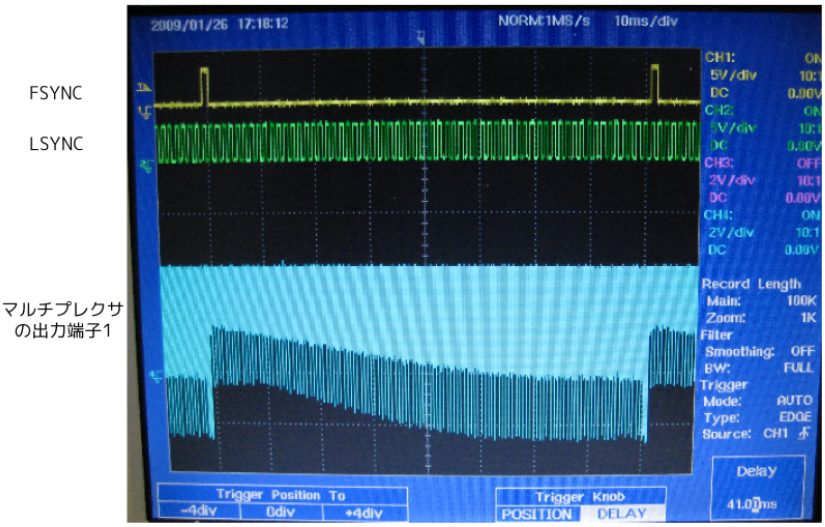

図 68　1frame の Analog データ (光を入れた時)

（b）1Line の確認

図 69,70 は以下の設定でオシロスコープ出力を確認したときの様子である。

- ch1(黄色)…FSYNC [5V/div,position:-3div]
- ch2(緑色)…LSYNC [5V/div,position-2div]
- ch3(赤色)…DCLK [5V/div,position-1div]
- ch4(青色)…マルチプレクサの出力端子 1 の Analog データ [2V/div,position:+2div]
- time:100 [usec/div]
- delay:2.200 [msec]

1line の中で DCLK は 16 周期の矩形波を持ち、1 周期で 2pixel へのアクセスを行うことで $2 \times 16 = 32$pixel へのアクセスを行う。また、マルチプレクサは出力端子を 4 本持っている。故に DLCK は 16 周期で $32 \times 4 = 128$pixel = 1line へのアクセスを行う。図 69 は検出器箱の蓋を閉じた状態でマルチプレクサを駆動したときの各端子電圧をモニタしたオシロスコープの画像である。図 67 と同様の理由で、出力端子 1 の電圧はほとんど変動していない。また、図 70 は検出器箱の蓋を開けマルチプレクサに可視光を当てて駆動したときの各端子電圧をモニタしたオシロスコープの画像である。この図で出力端子 1 の電圧は DCLK1/2 周期 (= ピクセルアクセス時間) 中に、積分電圧とリセット電圧の 2 つの電圧 (それぞれ下側、上側) を見ていることが分かる。図 70 から検出器感光部がないマルチプレクサのみの回路では、各 line の 1pixel 目だけ他の pixel より大きな積分電圧を出力することが判明した。

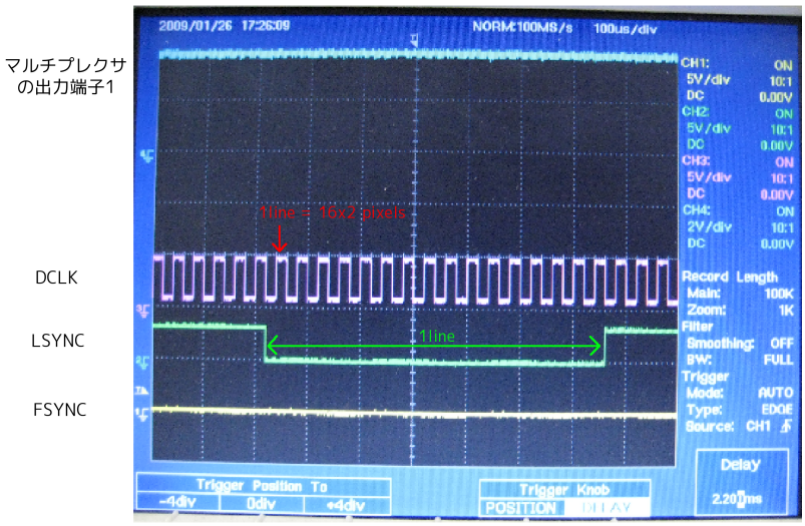

図 69　1line の Analog データ

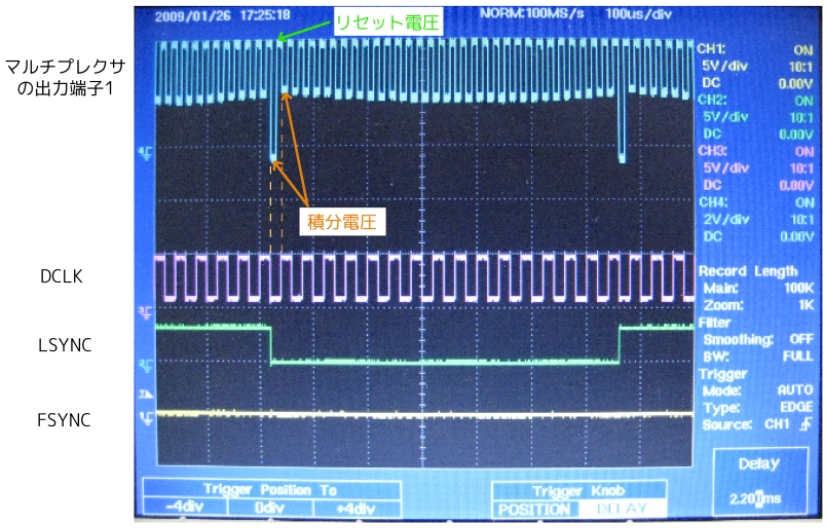

図 70　1line の Analog データ (光を入れた時)

（c）1Pixel の確認

図 71,72 は以下の設定でオシロスコープ出力を確認したときの様子である。

- ch1(黄色)…FSYNC [5V/div,position:-3div]
- ch2(緑色)…LSYNC [5V/div,position:-2div]
- ch3(赤色)…DCLK [5V/div,position:-1div]
- ch4(青色)…マルチプレクサの出力端子 1 の Analog データ [2V/div,position:+3div]
- time:10 [usec/div]
- delay:43.72 [msec]

図 71、72 は 1Frame を確認する際に発見した hot pixel のような pixel に着目し、各端子電圧をモニタしたオシロスコープの画像である。図 71 は検出器箱の蓋を閉じた状態でマルチプレクサを駆動したとき、図 72 は検出器箱の蓋を開けマルチプレクサに可視光を当てて駆動したときの各端子電圧をモニタしたオシロスコープの画像である。図 72 より、(b) でもみたように DCLK1/2 周期 (=ピクセルアクセス時間) の間に、積分電圧とリセット電圧を読み出しているのが確認できる。hot pixel 以外の pixel については、ピクセルアクセス時間内で積分電圧がリセットされ、リセット電圧が設定されている様子が図 72 からはっきり確認できる (double sampling mode でピクセルアクセス時間内に読み出す積分電圧、リセット電圧については図 21 を参照)。また、hot pixel のような pixel は検出器箱の蓋を開閉しても出力が全く変化しないことから、破損していると考えられる。この図 71、72 で見えている hot pixel は、リセット電圧の設定に問題があると考えられる。

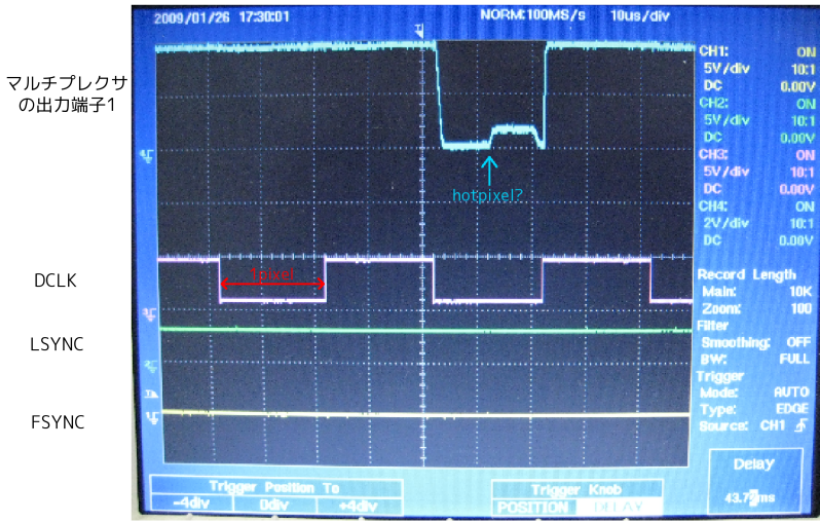

図 71　1pixel の Analog データ

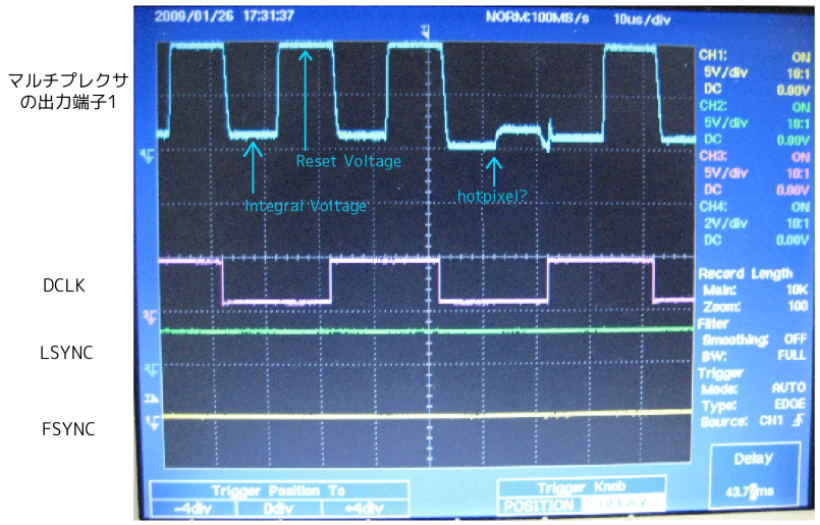

図 72　1pixel の Analog データ (光を入れた時)

4.2 Si:As 128×128 array 検出器駆動試験

4.2.1 試験目的

次に MIRSIS 撮像検出器である Boeing 社製 Si:As BIB 128×128 array 検出器の駆動試験 (p48、5 の試験) を行った。4.1 の Si:Sb BIB 128×128 array マルチプレクサ駆動実験により、TAC システムを用いて同型のマルチプクサが駆動できることを確認できている。撮像検出器は Si:Sb 検出器と同型のマルチプレクサを持つが、感光部の半導体が違うためにマルチプレクサに与えるバイアス電圧が異なる。故にバイアス電圧の設定を変更することで、Si:Sb マルチプレクサを駆動したときのシステムをこの検出器でも使用することが可能である。ここでは Si:As 128×128 array 検出器駆動実験の詳細および温度 · 積分時間を変化させた際の検出器の変化の確認について述べていく。

4.2.2 試験環境

MIRSIS 真空容器を用いて検出器を低温 (∼ 3.5K) まで冷却し試験を行う。Si:Sb の試験と同様に検出器の制御にはリアルタイム Linux OS と TAC システムを用いる。検出器へのクロック電圧とバイアス電圧を送る配線を図 73 に示す。フロントエンド回路から真空容器まではマルチプレクサの試験と同様の特注ハーネスを用いる。容器内部では常温の熱を遮断するため熱伝導率の悪いマンガニン配線を用いた後、光学定盤に固定した銅コアのサーマルアンカ基板 (∼30K まで冷却) に接続する。サーマルアンカ基板から検出器箱の前までは銅プリント基板ケーブルを用いて、検出器箱 (∼4K まで冷却) の前でマンガニンプリント基板ケーブルに変換する。検出器箱には MAX38 で使用している検出器箱の機械設計、熱設計を参考に製作した MIRSIS 撮像検出器用検出器箱を用いる。

また、検出器の駆動を確かめるために実装用抵抗を光源として使用する。実装用抵抗に電圧をかけることで発生する赤外線熱放射が、赤外線検出器にとっての光源となる。図 74 に示すように、検出器箱のふたを開けた状態で検出器箱の窓の前に実装用抵抗を固定する。固定された実装用抵抗の周りは、アルミの板とアルミテープを用いて遮光を施す。この様な方法で実際に光源を配置し、遮光を施した写真が図 75 である。

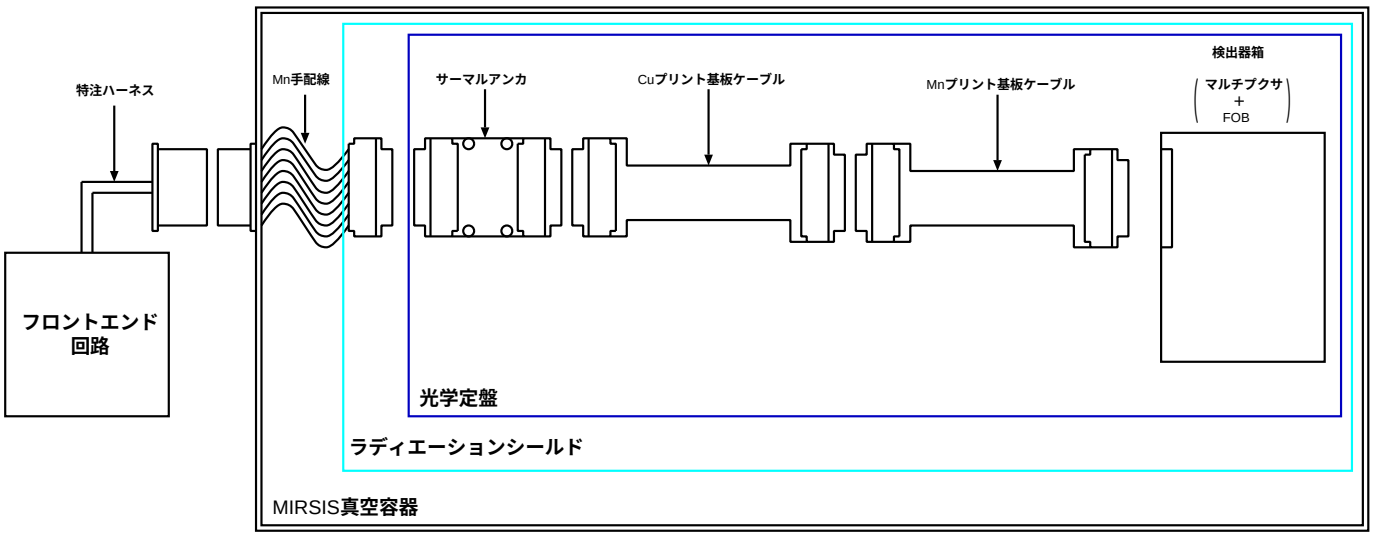

図 73　検出器駆動試験時の配線概略図

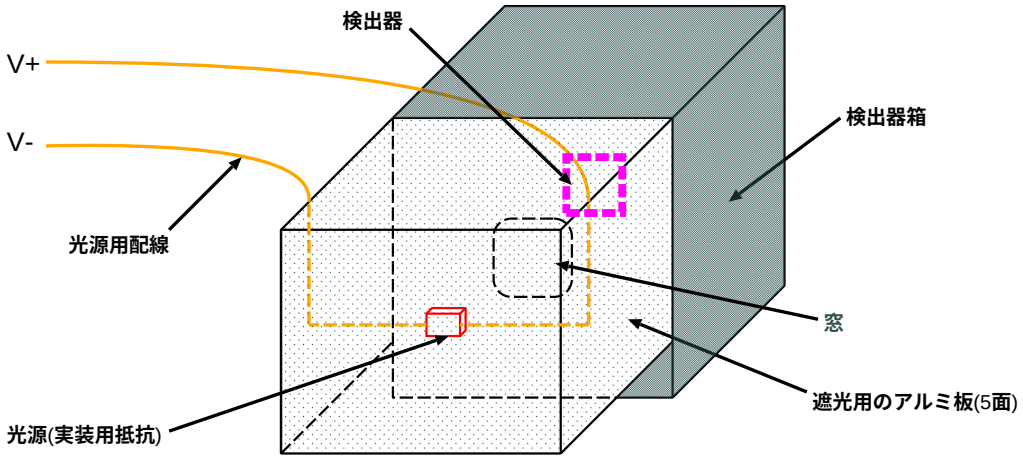

図 74　検出器箱と光源の位置関係

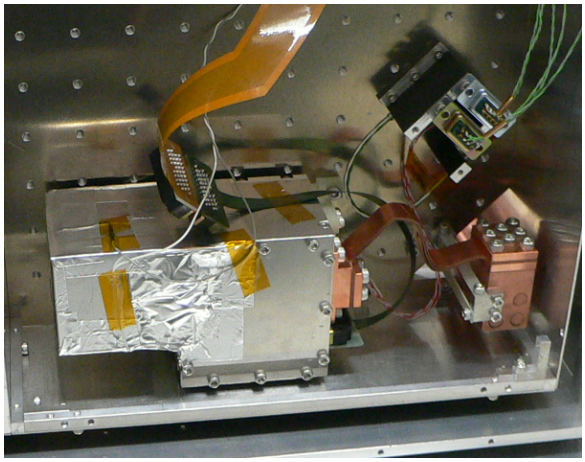

図 75　光源の周りにアルミ板とアルミテープをも用いて実際に遮光を施した撮像検出器箱の写真

4.2.3　試験方法

真空容器の蓋を全て閉じ、真空引きと冷却を行う。検出器が極低温 (∼3K) まで冷却され温度が一定になった後、検出器の駆動を確認するための画像を取得する。その後、光源への反応を確認するため光源を光らせて画像を取得する。最後に検出器の温度を 3.2K∼7.0K の間で変化させ、それぞれの温度で積分時間が 81∼1050msec の画像をそれぞれ取得し、積分時間と温度による検出器出力の変化を確認する。

4.2.4 実験結果

図 76 は今回の試験で撮像検出器を駆動した際、初めて取得した画像である。クロックタイプ 3(各出力端子からの出力を確認) を用いて、検出器温度は 3.0K、積分時間が 81msec の光源なしで撮影した fits 画像である。各 pixel 間で出力の違いが見られることから、暗電流による出力が確認できたと考えられる。故に、撮像検出器の駆動に成功した。また、検出器の光源への反応を確かめるためにクロックタイプ 0 を用いて撮影した画像が図 77 と図 78 である。図 77 は検出器箱の前に設置した光源を光らせたときの画像で、図 78 はこの光源を消して数分待った後の画像である。2 枚の画像を比較すると、各 pixel からの出力に大きな違いがあることが確認できる。故に、検出器が光源へ反応したと考えられる。最後に光源をオフにして、検出器の温度、積分時間を変化させることによって、暗電流による出力にどのような変化があるか確認したのが図 79 である。それぞれ検出器温度が 3.2K、5.0K、7.0K の各温度で積分時間が 81msec、565msec、1050msec で撮影した画像である。この画像から、温度や積分時間に依存せず、出力が常に一定の pixel (各画像の黒い点)、温度や積分時間が増加すると極端に出力が増える pixel (各画像の白い点) の 2 種類の dead pixel が確認できる。ここでは前者の pixel を Dark dead pixel、後者の pixel を Hot dead pixel と呼ぶことにする。各 dead pixel は、温度または積分時間が増加すると顕著に現れている。ただし Dark dead pixel の場合、実際には温度に依らず一定の低レベル出力を持っていて、単に低温で短積分時間の場合には周囲の pixel の出力レベルも低く、dead pixel が目立たないだけと考えられる。MIRSIS で使用する撮像検出器は、以前、他の装置で使用していた検出器を再利用して使用している。そのため検出器が製造されてから長い年月が経ったことによる検出器の劣化が、これらの dead pixel が発生した主な原因だと考えられる。

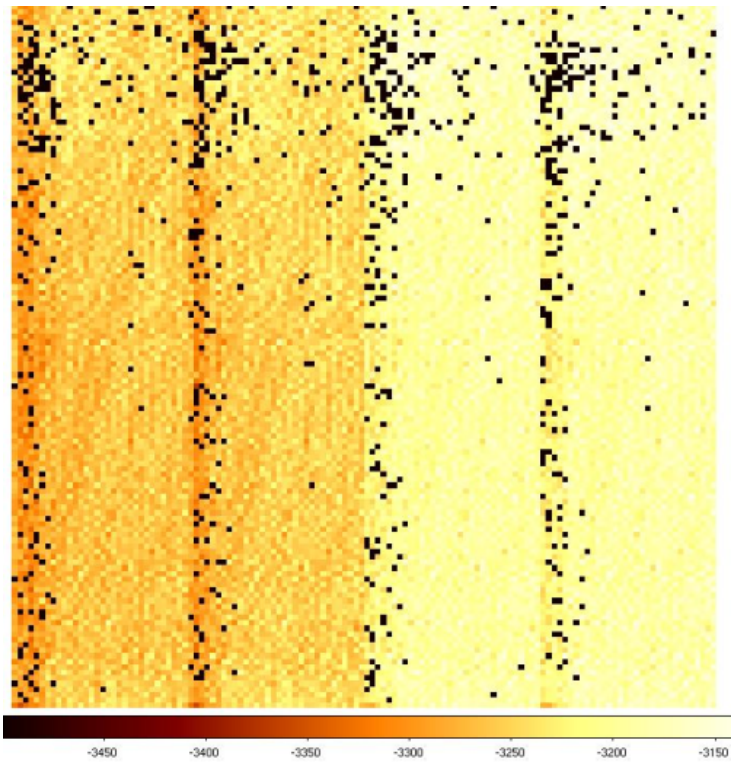

図 76　撮像検出器の駆動に成功し、初めて取得した画像 (クロックタイプ 3 を使用)

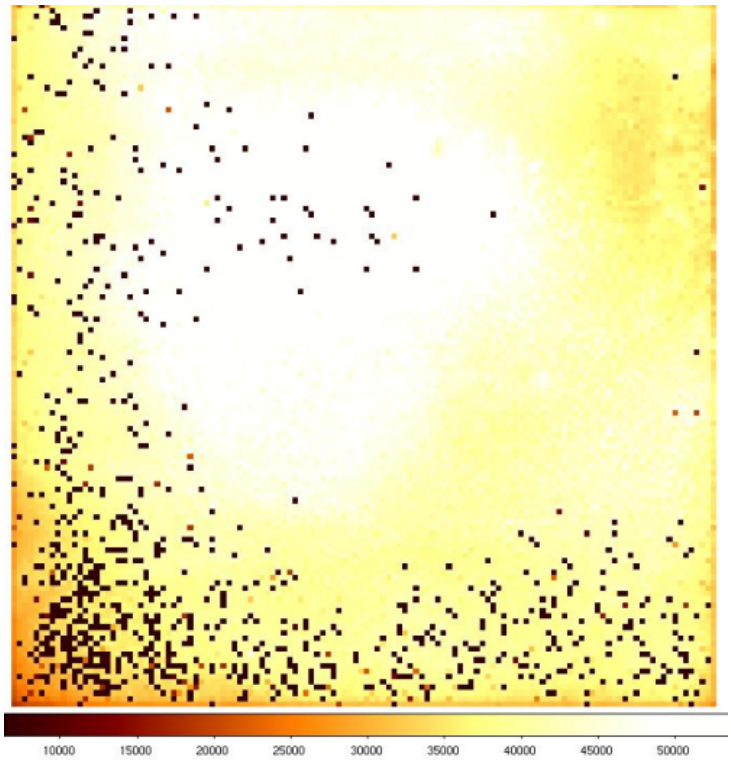

図 77　光源が ON の状態

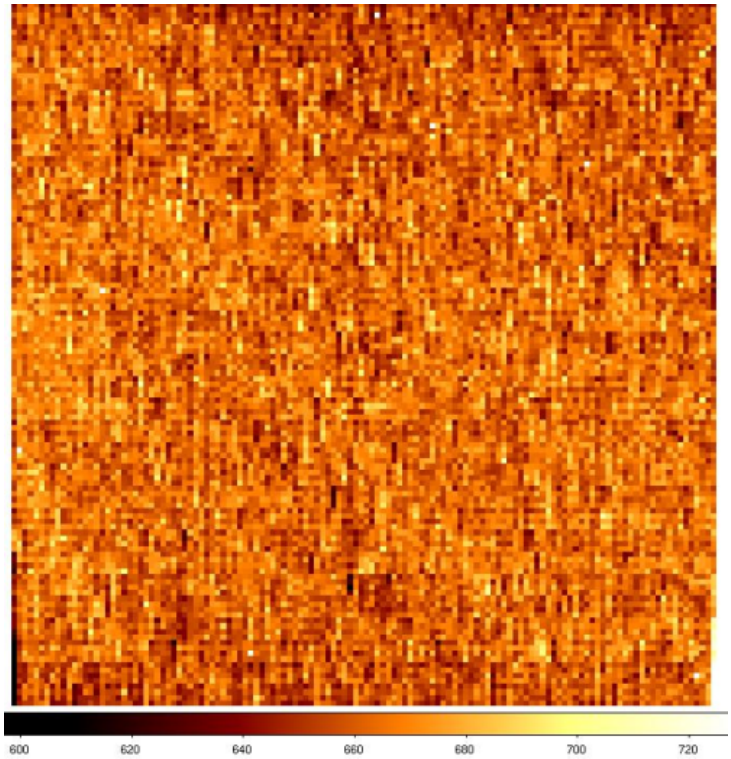

図 78　光源が OFF の状態

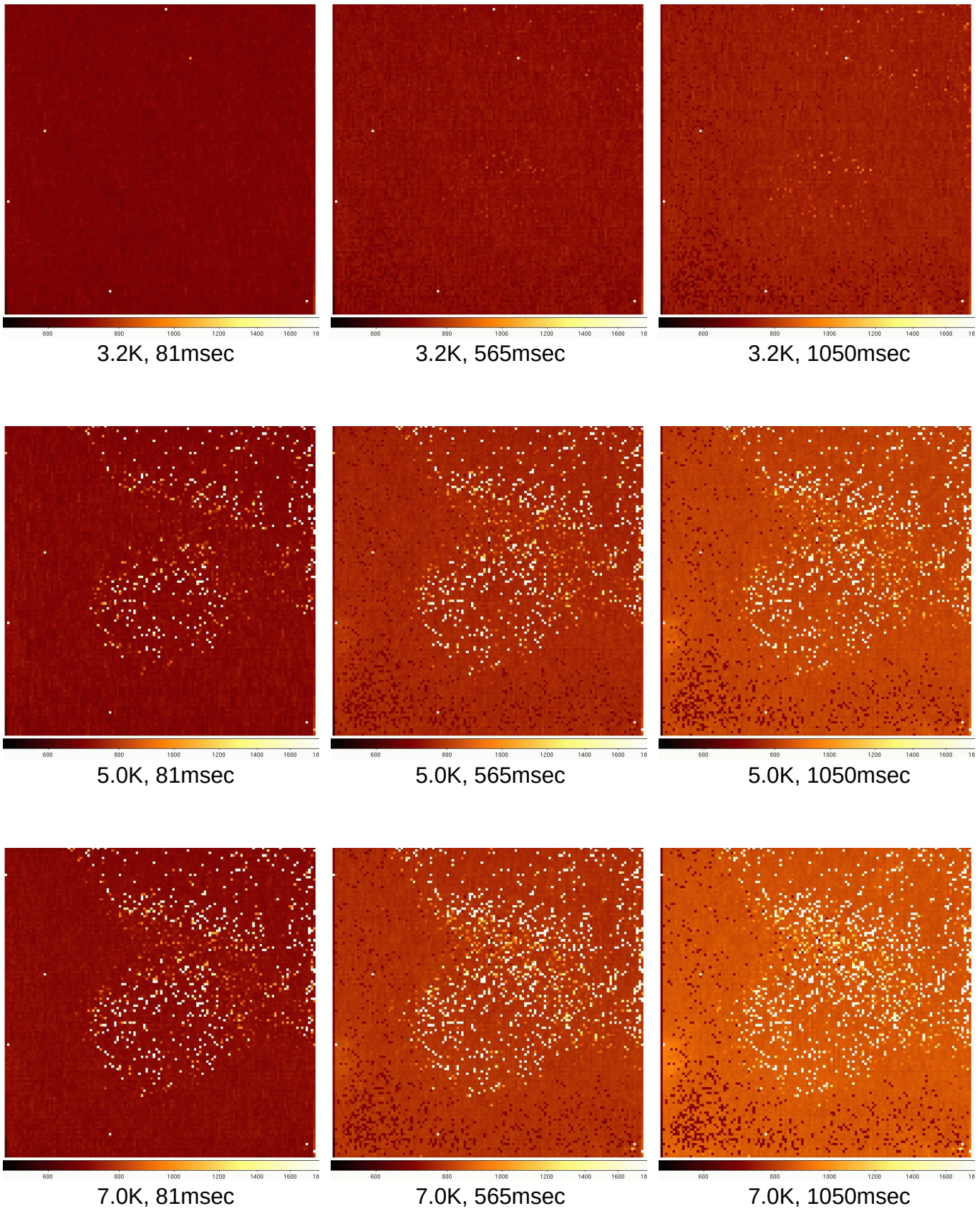

図 79　各温度、積分時間での暗電流による出力 (全て同じ表示レンジ)

5　MIRSIS 分光検出器 Si:As 320×240 array 検出器の立上げ

本章では、MIRSIS 分光検出器である Si:As 320×240 array 検出器の立上げに関し、これまで行なってきた実験について述べていく。MIRSIS 撮像検出器の駆動に成功し、新たに MIRSIS 分光検出器の立上げを行なうにあたって以下の準備が必要となった。

- 検出器制御用 PC の立上げ (ハードウェアの組立て、ソフトウェアのインストールと設定)
- 検出器箱の製作
- DRV ボードの実装と駆動試験
- ADC ボードの実装と駆動、特性測定試験
- FOB ボードの製作と実装
- TAC システムの撮像検出器仕様から分光検出器仕様への設定

上記に示した中で、ここでは ADC ボードの特性測定試験、DRV ボードの駆動試験について述べていく。

5.1　ADC ボードの特性測定試験

ADC ボードでは、FOB を通して検出器から出力された電圧をプリアンプ回路と差動アンプ回路によって AD 変換器の入力電圧幅に電圧を変換し、2 次ベッセルフィルター回路で高周波のノイズをカットした後に AD 変換器で AD 変換を行なう (図 80)。

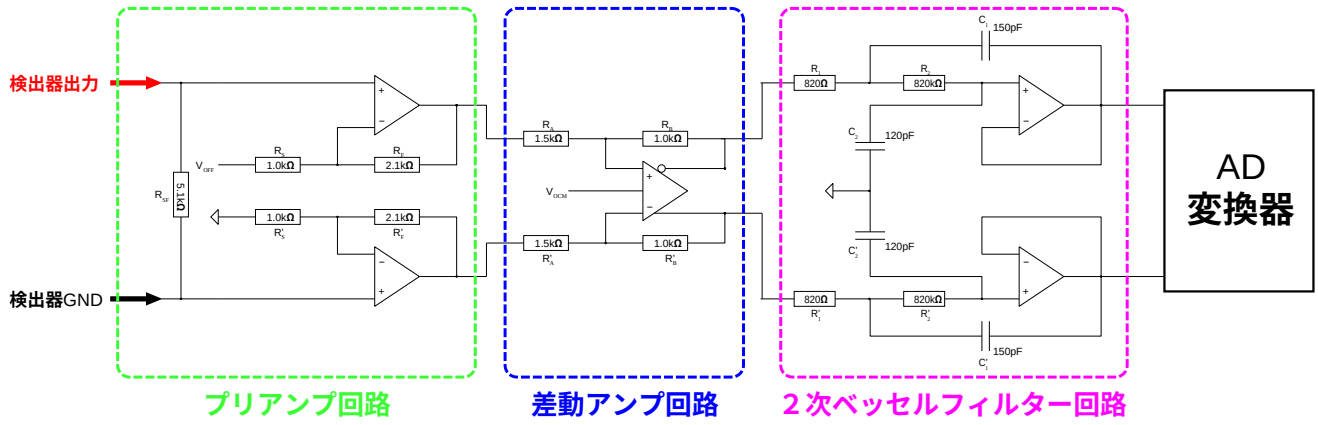

図 80　ADC ボードの検出器出力の入力から AD 変換器までの回路

Si:As 320×240 array 検出器は出力端子を 16 本持つ (表 2 参照)。故に、ADC ボードでの AD 変換回路群も 16 チャンネル (ch-00∼ch-15) 必要になる (図 81)。また分光検出器からの出力は極低温 (<10K) まで冷却すると、各出力端子からリセット状態で-2.5V、積分コンデンサのコンデンサ容量の最大まで電荷を積分した状態で-0.5V の電圧を出力する (Si:As 320×240 array 検出器のマニュアルより)。ADC ボードの実装を行なう際、この電圧幅と AD 変換器の入力電圧幅 (0∼3V) を考慮した (詳細は後述参照) 。また実装後の ADC ボードについて 2 つの特性測定試験を行なったので、その実験の詳細と結果について述べていく。

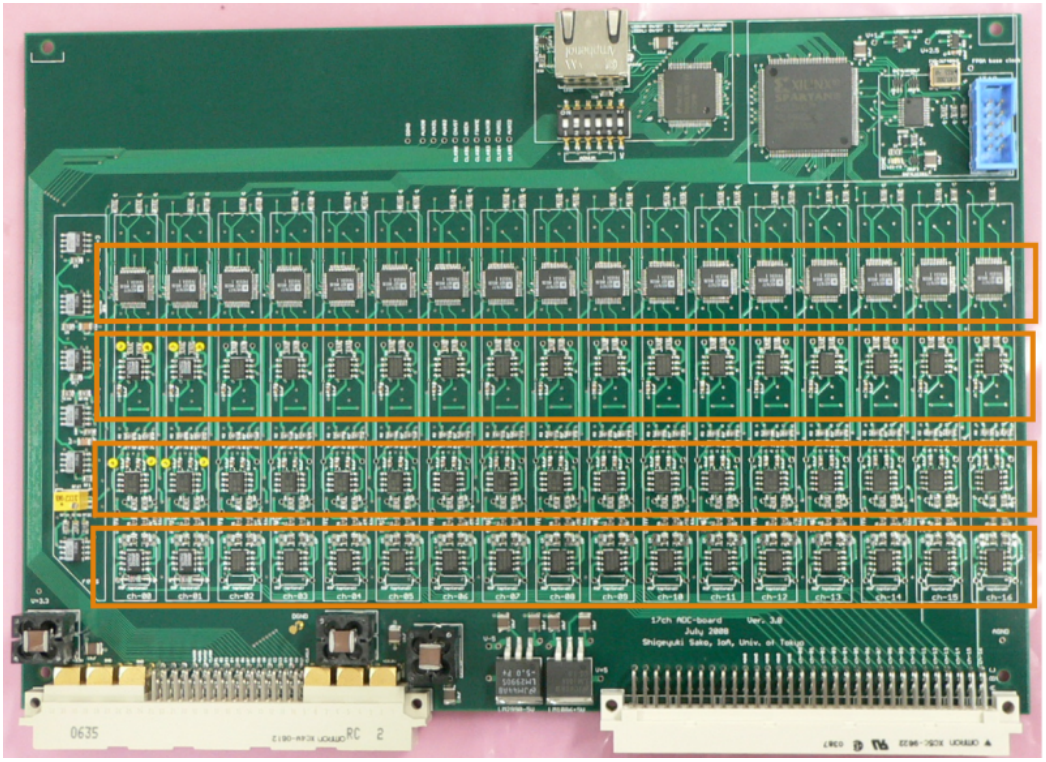

図 81　Si:As 320×240 array 検出器用 ADC ボード (17 チャンネル分の AD 変換回路群が実装されているが、実際には 16 チャンネル分のみ使用する)

5.1.1　ADC ボードの AD 変換回路群概要

- プリアンプ回路

検出器の出力電圧 V_{Out}^{Det} と検出器 GND 電圧 V_{GND}^{Det} はプリアンプ回路 (図 82) によって電圧が増幅される。増幅された後の電圧 V_{Out}^{PA}、V_{GND}^{PA} は以下の式で表される。

$$V_{Out}^{PA} = V_{Out}^{Det} \times \left(1 + \frac{R_f}{R_s}\right) - V_{OFF} \times \frac{R_f}{R_s} \tag{11}$$

$$V_{GND}^{PA} = V_{GND}^{Det} \times \left(1 + \frac{R'_f}{R'_s}\right) \tag{12}$$

$$\begin{cases} R_s、R'_s = 1.0(k\Omega) \\ R_f、R'_f = 2.1(k\Omega) \\ V_{OFF} : オフセット電圧\ (V) \end{cases}$$

このプリアンプ回路で重要なのはオフセット電圧 V_{OFF} の設定である。V_{Out}^{PA} が V_{Out}^{Det} の関数と考え $V_{Out}^{PA}(V_{Out}^{Det})$ と表したとき、以下の式の条件を満たす V_{OFF} 電圧を設定することで、差動アンプの出力電圧を V_{OCM} 電圧を中心に絶対値が同じ値だけ増減することができ、AD 変換器での AD 変換の精度が向上する (詳細は後述参照)。

$$|V_{Out}^{PA}(-2.5V)| = |V_{Out}^{PA}(-0.5V)| \tag{13}$$

式 (11)、(13) より V_{OFF} は以下の式で決まる。

$$\begin{aligned} |-2.5 \times 3.1 - V_{OFF} \times 2.1| &= |-0.5 \times 3.1 - V_{OFF} \times 2.1| \\ V_{OFF} &= -2.214(V) \end{aligned} \tag{14}$$

上式より V_{OFF} は-2.214V になるように設定する。ADC ボードでは可変抵抗 (0∼1kΩ) を調整することで、V_{OFF} を調整することが可能である (図 83)。

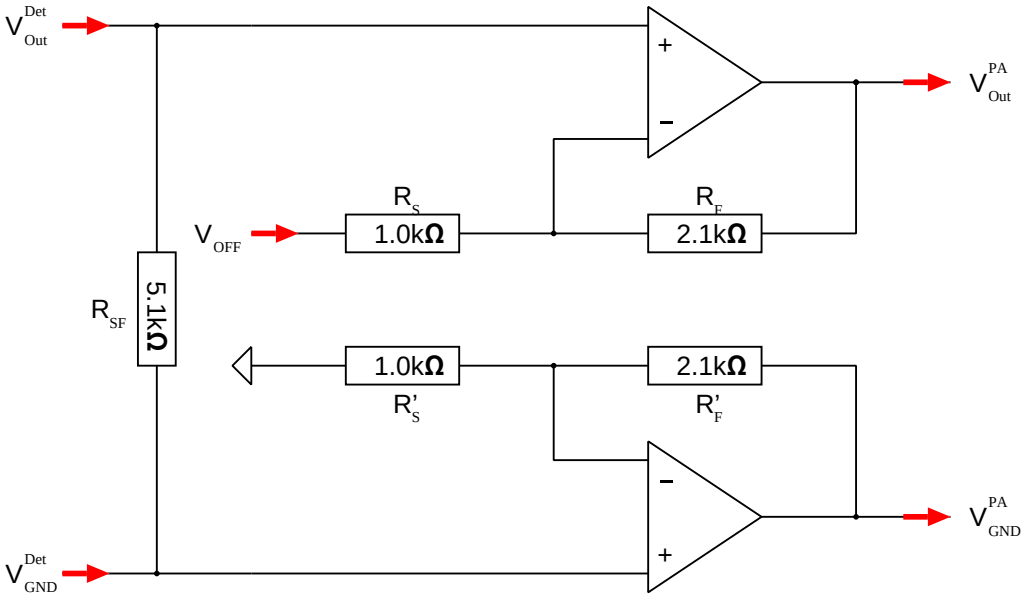

図 82　プリアンプ回路

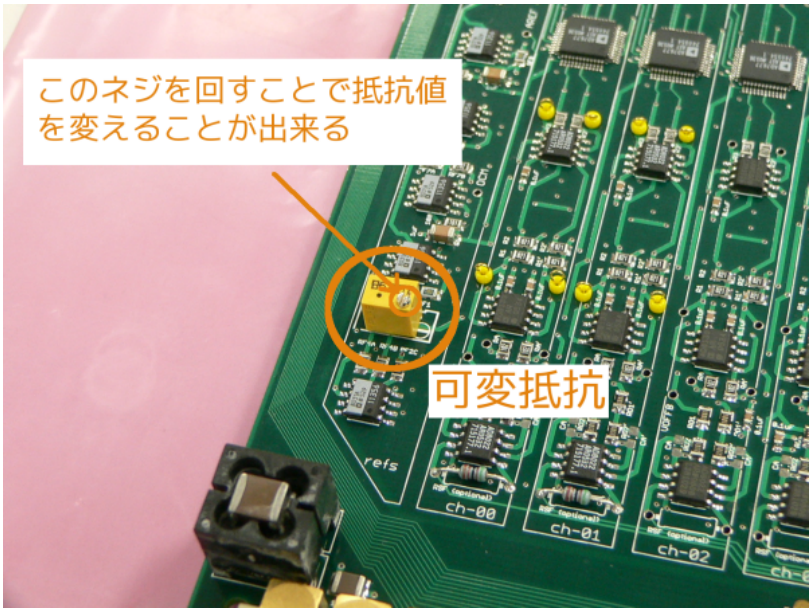

図 83　ADC ボード上の可変抵抗

- 差動アンプ回路
  プリアンプ回路で増幅された電圧は、差動アンプ回路に入力される (図 84)。差動アンプ回路は以下の式に示す性質を持つ。

$$(V_{Out}^{PA} - V_{GND}^{PA}) \times G = (V_{DF}^{+} - V_{DF}^{-}) \tag{15}$$

$$G = \frac{R_B}{R_A} \tag{16}$$

$$\frac{V_{DF}^{+} + V_{DF}^{-}}{2} = V_{OCM} \tag{17}$$

$$\begin{cases} R_A、R'_A = 1.5(k\Omega) \\ R_B、R'_B = 1.0(k\Omega) \\ V_{DF}^{-}：差動アンプの + 側出力電圧\ (V) \\ V_{DF}^{+}：差動アンプの - 側出力電圧\ (V) \\ V_{OCM}：コモンモード電圧\ (V) \end{cases}$$

アンプに与える V_{OCM} は、式 (17) より V_{DF}^{-}、V_{DF}^{+} の平均電圧を決めている。AD 変換器の入力電圧幅は 0~3V なので、V_{OCM} を (0+3)/2=1.5(V) に設定することで差動アンプの出力電圧幅と AD 変換器の入力電圧幅を合わせることができ、より精度の良い AD 変換を行なうことが可能である。また、式 (15)~(17) より差動アンプから出力される電圧 V_{DF}^{-}、V_{DF}^{+} は以下の式で表される。

$$V_{DF}^{-} = V_{OCM} - \frac{V_{Out}^{PA} - V_{GND}^{PA}}{2} \cdot \frac{R_B}{R_A} \tag{18}$$

$$V_{DF}^{+} = V_{OCM} + \frac{V_{Out}^{PA} - V_{GND}^{PA}}{2} \cdot \frac{R_B}{R_A} \tag{19}$$

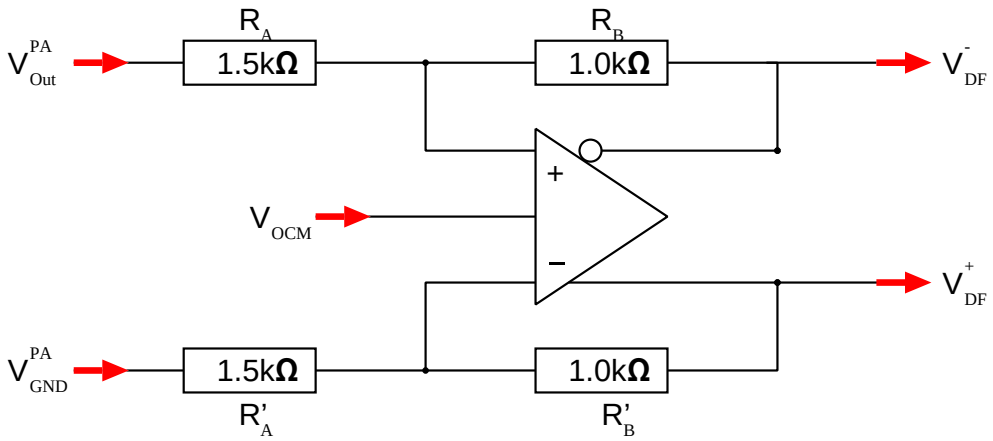

図 84　差動アンプ回路

- 2 次ベッセルフィルター回路

プリアンプ回路と差動アンプ回路によって増幅された電圧は、2 次ベッセルフィルター回路に入力される (図 85)。2 次ベッセルフィルター回路は高性能のローパスフィルターで入力電圧組を $v_{in} = (V_{DF}^{-}、V_{DF}^{+})$、出力電圧組を $v_{out} = (V_{BF}^{-}、V_{BF}^{+})$ とすると、2 つの電圧の関係は以下の式で表される。

$$v_{out} = \frac{\omega_0^2}{-\omega^2 + i\frac{\omega_0}{Q\omega} + \omega_0^2} \cdot v_{in} \tag{20}$$

$$\omega_0 = \frac{1}{\sqrt{C_1 C_2 R_1 R_2}} \tag{21}$$
$$Q = \omega_0 C_1 \frac{R_1 R_2}{R_1 + R_2}$$

$$\begin{cases} C_1 = 150(\mathrm{pF}) \\ C_2 = 120(\mathrm{pF}) \\ R_1 = 820(\Omega) \\ R_2 = 820(\Omega) \\ \omega : v_{in}\text{の角振動数 (rad/sec)} \end{cases}$$

$\alpha \equiv \omega_0/Q$、$\beta \equiv \omega_0^2$ と定義すると、式 (20) より Gain は以下の式で与えられる。

$$\begin{aligned} \mathrm{Gain} &= \left| \frac{v_{out}}{v_{in}} \right| \\ &= \frac{|\beta|}{|\beta - \omega^2 + i\alpha\omega|} \\ &= \frac{\beta}{\sqrt{(\beta - \omega^2)^2 + \alpha^2\omega^2}} \end{aligned} \tag{22}$$

式 (22) より、v_{in} の角振動数 ω が 0(直流) に近付くと Gain~1 となり、角振動数 ω が ∞ に近付くと Gain~0 となる。また、式 (21) より与えられるカットオフ角振動数 ω_0 からカットオフ周波数 f_0 を求めると $f_0 = \omega_0/2\pi = 1.45(\mathrm{MHz})$ となる。

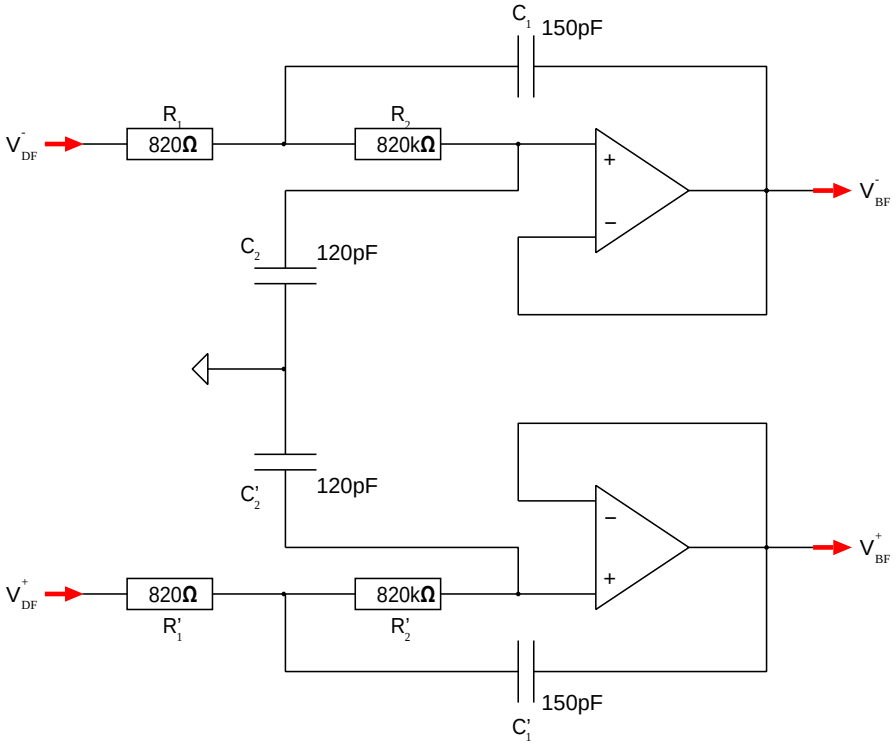

図 85　2 次ベッセルフィルター回路

- AD 変換器へ入力される電圧

  2 次ベッセルフィルターを Gain∼1 で通過し、AD 変換器へ入力される電圧 V_{BF}^{-}、V_{BF}^{+} は式 (11)、(12)、(18)、(19) より以下の式で表される。

$$V_{BF}^{-} = V_{OCM} - \frac{1}{2}\left(V_{Out}^{Det} \times \left(1 + \frac{R_f}{R_s}\right) - V_{OFF} \times \frac{R_f}{R_s} - V_{GND}^{Det} \times \left(1 + \frac{R'_f}{R'_s}\right)\right) \cdot \frac{R_B}{R_A} \tag{23}$$

$$V_{BF}^{+} = V_{OCM} + \frac{1}{2}\left(V_{Out}^{Det} \times \left(1 + \frac{R_f}{R_s}\right) - V_{OFF} \times \frac{R_f}{R_s} - V_{GND}^{Det} \times \left(1 + \frac{R'_f}{R'_s}\right)\right) \cdot \frac{R_B}{R_A} \tag{24}$$

- AD 変換器

  2 次ベッセルフィルター回路によってノイズがカットされた電圧は AD 変換器へ入力される。AD 変換器による AD 変換の様子を図 86 に示す。AD 変換器では、V_{BF}^{-} の位相を反転させた電圧と V_{BF}^{+} を加算する。加算された電圧を AD 変換器にかかる ADTRG クロックが決めるタイミングで、16bit AD 変換を行なう。このクロックは任意の時刻を指定できる。AD 変換後の Digital データは検出器制御用 PC へと転送される。

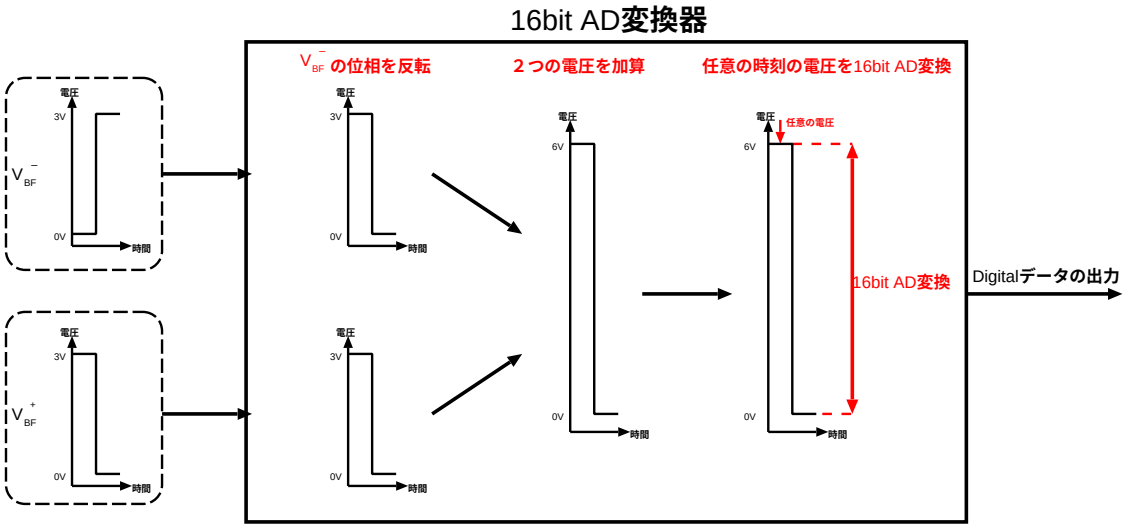

図 86　16bit AD 変換器の動作

5.1.2　ADC ボードの特性測定試験 1(アンプ回路の動作確認)

以上をふまえて、分光検出器用 ADC ボードの単体試験を行う。ここではまず、AD 変換回路群の中の ch-00、01 について、プリアンプ回路と差動アンプ回路が理論値通りの動作を行っているか実験を行った。図 87 に ADC ボードのアンプ回路の動作確認を行った際の実験状況を示す。V_{Out}^{Det} として直流電源を用いて-3.0~0.0(V) の直流電圧を入力し、差動アンプ回路から出力される電圧 V_{DF}^{-}、V_{DF}^{+} を測定する。この実験を行なう際の準備として、ADC ボードに電源を投入することで発生する V_{OFF} 電圧は図 87 の VOFF 電圧測定地点で測定し、式 (14) より-2.214(V) に調整を行なった。図 88、図 89 は横軸に V_{Out}^{Det}(V)、縦軸にそれぞれ V_{DF}^{-}、V_{DF}^{+}(V) をとったグラフである。各グラフには ch-00、01 について測定から得られた実測値と式 (11)、(12)、(18)、(19) から得られる理論値をプロットしてある。2 つのグラフから V_{Out}^{Det} が-2.5~-0.5(V) の間で、理論値と実測値がほぼ一致していることが分かる。逆にこの範囲外では、オペアンプによる電圧増幅の線形性が保たれなくなることが分かる。この線形性が保たれなくなる理由については現在、調査中である。故に今回の試験結果から、分光検出器から出力される予定の電圧範囲 (-2.5~-0.5V) でアンプ回路の線形性が確認できた。

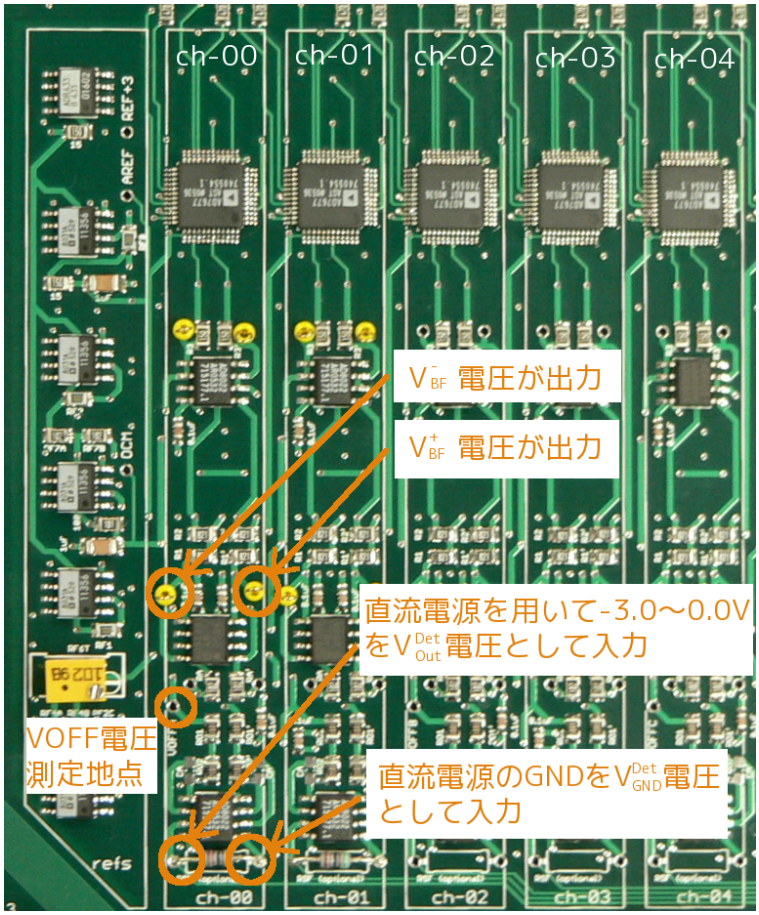

図 87　ADC ボードの特性測定試験状況 (アンプ回路の動作確認)

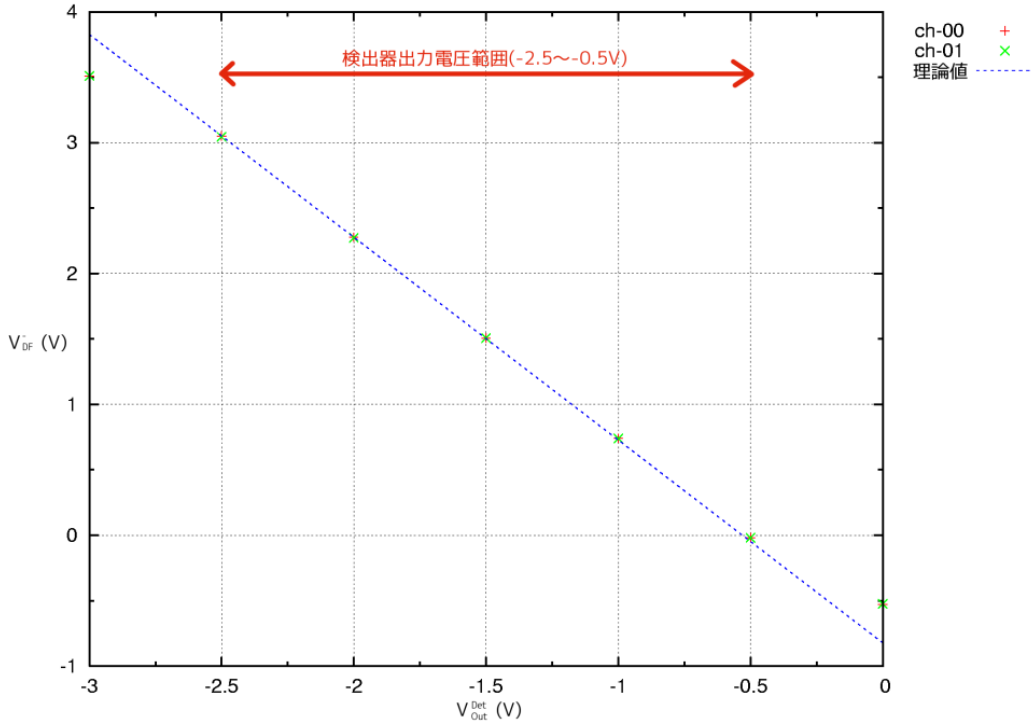

図 88　V_{DF}^{-} 電圧の比較

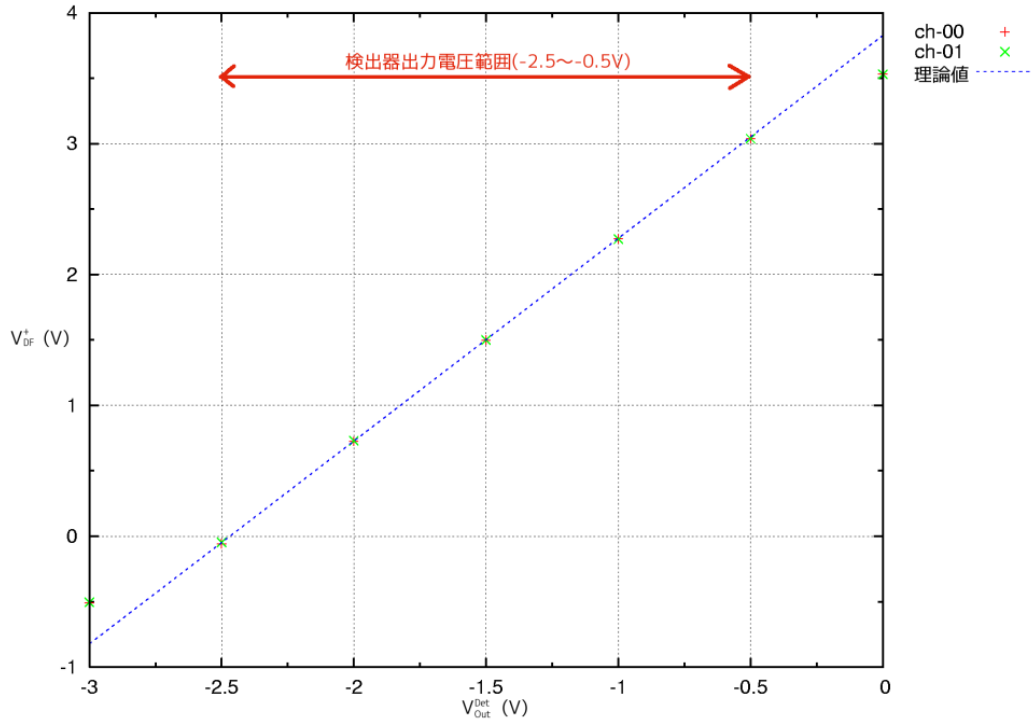

図 89　V_{DF}^{+} 電圧の比較

5.1.3　ADC ボードの特性測定試験 (フィルター回路の動作確認)

次に ADC ボードの AD 変換回路群の中で ch-00 について、ベッセルフィルター回路が理論値通りの動作を行っているか周波数特性の測定実験を行った。図 90 に ADC ボードのフィルター回路の動作確認を行った際の実験状況を示す。V_{Out}^{Det} として Function Generator を用いて検出器の出力範囲 (5.1 参照) に合わせて、最小値が-2.5V、最大値が-0.5V、振動数が 100(Hz)~2.2(MHz) の sin 波を入力する。この入力電圧に対してフィルター回路から出力される電圧 V_{BF}^{-}、V_{BF}^{+} の振幅をオシロスコープを用いて測定する。この測定した振幅と入力電圧の振幅の比を用いて Gain を計算する。この実験を行なう際も、V_{OFF} 電圧は-2.214(V) に調整を行った。図 91 は横軸に V_{Out}^{Det} の振動数 (Hz)、縦軸にそれぞれ V_{BF}^{-}、V_{BF}^{+} から出力された電圧の Gain をとったグラフである。グラフには ch-00 について測定から得られた実測値と式 (22) から得られる理論値がプロットしてある。グラフから実測値で得られた Gain と理論値の Gain がほぼ一致していることが分かる。また、カットオフ周波数 f_0=1.45(MHz) 付近で Gain~0.5 となったことも確認できる。以上より、正常にフィルター回路がローパスフィルターとして機能していることが分かった。

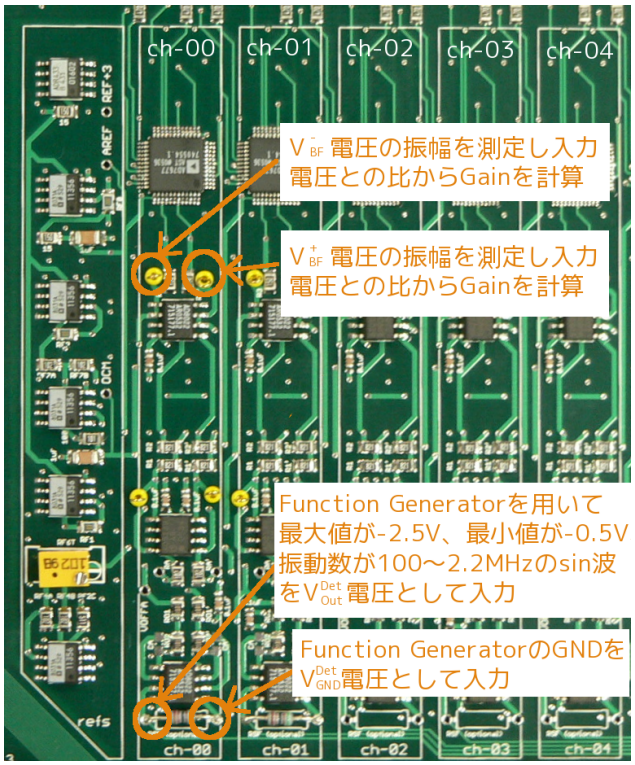

図 90　ADC ボードの特性測定試験状況 (フィルター回路の動作確認)

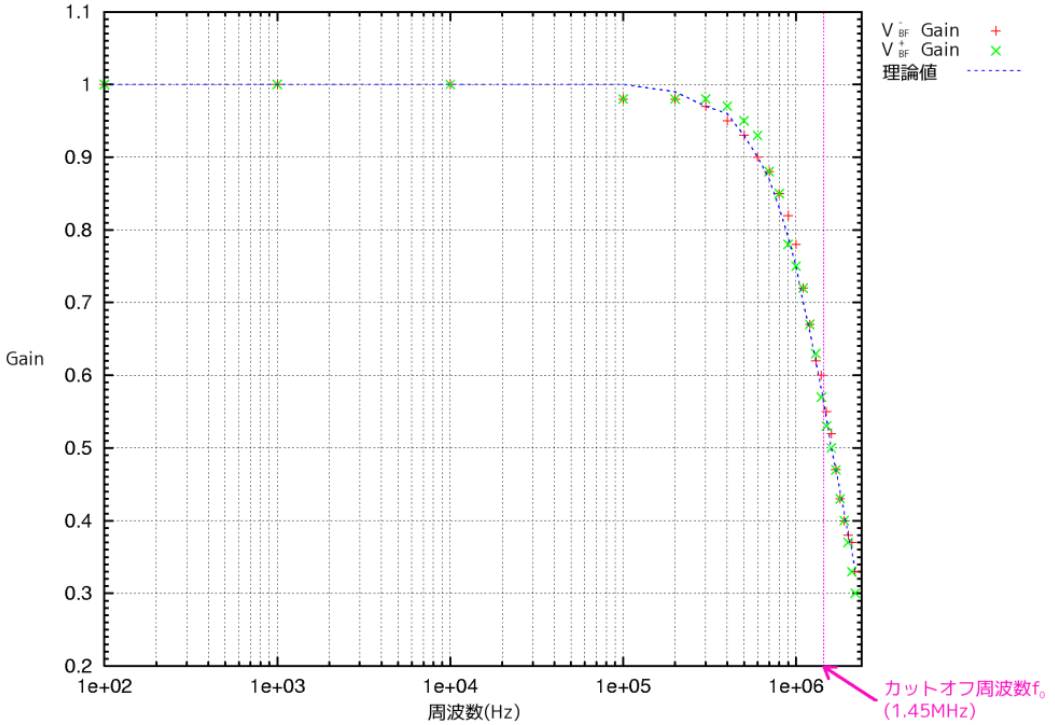

図 91　Gain の比較

5.2　TAC システムを用いた DRV ボードの駆動試験

Si:As 128×128 array 検出器では 19 個のバイアス電圧を設定、制御する必要があった。今回、立上げを行っている Si:As 320×240 検出器では 17 個のバイアス電圧を設定する必要がある。3.5 章で述べた bias 設定方法を参考にしつつ、新たに Si:As 320×240 array 検出器用 DRV ボード (図 92) の実装と TAC システムの設定を行ったので、本節では TAC システムの変更点と DRV ボードの駆動試験について述べていく。

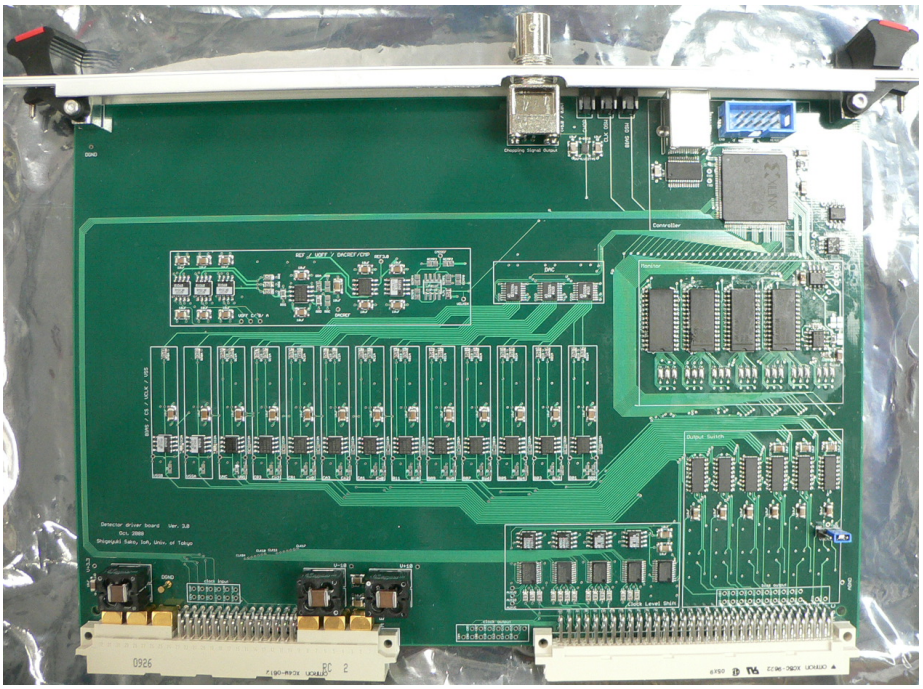

図 92　Si:As 320×240 array 検出器用 DRV ボード

5.2.1 TAC システムの変更点

Si:As 128×128 array 検出器用バイアス電圧の設定を行う header ファイルをもとに、SiAs 320×240 array 検出器用バイアス電圧の設定を行う header ファイルを作成した (図 93)。まず電圧設定値、電流設定値、バイアス名 (Alias) を検出器のマニュアルと酒向氏の博士論文を参考に header ファイルの V、I、Alias 欄にそれぞれ記入した。更に、各バイアス電圧の DDAC 値を 3.5 章で述べた計算方法を用いて計算し、header ファイルの DDAC 欄に記入した。また、header ファイルの OSWORD 欄には各バイアス電圧を出力する際の出力順を記入する必要があるのだが、現在の所は上から昇順に番号を記入しておくことにした。このバイアス電圧の出力順序は今後、実験を行っていく上で決定する。この際、使用しないバイアス電圧の OSWORD 欄に「0」を記入しておくと、このバイアス電圧が出力されなくなる。不要な電圧が出力されないよう、使用しないバイアス電圧の OSWORD 欄には 0 を記入しておくことが重要である。

```
/* 内部バイアス電圧(=内部生成電圧) */
/* Bias設定値は、本当は全てにマイナスが付く*/
/*  TYPE, NAME, MUXCH, DACH, OSWCH, V, I, DDAC, OSWORD, ALIAS */
static struct struct_bias btbl_data[TBL_NUM] = {
  { "BIAS", "BIAS0" ,14, 0 , 0 , 6.9,   0,  490,  1, "VDETGRV" },
  { "BIAS", "BIAS1" ,0 , 1 , 1 , 7.0,   0,  360,  2, "VNEG"    },
  { "BIAS", "BIAS2" ,9 , 2 , 2 , 5.1,   0, 2954,  3, "VDETGRD" },
  { "BIAS", "BIAS3" ,15, 3 , 3 , 6.0,   0, 1715,  4, "VPW"     },
  { "BIAS", "BIAS4" ,1 , 4 , 4 , 0.0,   0,    0,  0, "--"      },
  { "BIAS", "BIAS5" ,7 , 5 , 5 , 5.5,   0,  663,  5, "VCLCOL"  },
  { "BIAS", "BIAS6" ,6 , 6 , 6 , 4.1,   0, 2604,  6, "VCLOUT"  },
  { "BIAS", "BIAS7" ,8 , 7 , 7 , 3.8,   0, 2984,  7, "VGATE"   },
  { "BIAS", "BIAS8" ,30, 8 , 8 , 3.5,   0,  636,  8, "VCASUC"  },
  { "BIAS", "BIAS9" ,24, 9 , 9 , 3.0,   0, 1259,  9, "VSHPCHG" },
  { "BIAS", "BIAS10",25, 10, 10, 2.8,   0, 1532, 10, "VOFFSET" },
  { "BIAS", "BIAS11",31, 11, 11, 2.0,   0, 2654, 11, "VRSTUC"  },
  { "CS"  , "CSA0"  ,46, 12, 12, 1.7, 100, 2062, 12, "IMUX"    },
  { "CS"  , "CSA1"  ,36, 13, 13, 1.9, 320, 1680, 13, "VGG2"    },
  { "CS"  , "CSA2"  ,34, 14, 14, 1.8, 160, 2917, 14, "VGG3"    },
  { "CS"  , "CSA3"  ,47, 15, 15, 1.6, 160, 3220, 15, "VGG4"    },
  { "CS"  , "CSB0"  ,62, 16, 16, 0.0,   0,    0,  0, "--"      },
  { "CS"  , "CSB1"  ,52, 17, 17, 0.0,   0,    0,  0, "--"      },
  { "CS"  , "CSB2"  ,50, 18, 18, 0.0,   0,    0,  0, "--"      },
  { "CS"  , "CSB3"  ,63, 19, 19, 0.0,   0,    0,  0, "--"      },
  { "VCLK", "VCLKNO",22, 20, 22, 5.0,   0, 3100, 16, "CLKL"    },
  { "VCLK", "VCLKNC",23, 21, 22, 0.0,   0,    0,  0, "--"      },
  { "VSS" , "VSSA"  ,37, 22, 20, 5.5,   0, 2045, 17, "VSS"     },
  { "VSS" , "VSSB"  ,53, 23, 21, 0.0,   0,    0,  0, "--"      }
};
```

図 93　Si:As 320×240 array 検出器用バイアス電圧の設定をするための header ファイル

5.2.2 DRV ボード駆動試験

TAC システムを変更後、Si:As 320×240 array 検出器用 DRV ボードが正常に動作をするか DRV ボードの駆動試験を行ったので、ここでは実験手順と結果を述べていく。図 94 に実験を行った際の端末のキャプチャをのせる。まず初めに、Si:As 128×128array 用ステータスファイルをもとに作成した Si:As 320×240 array 用ステータスファイルを db_upfile コマンドを用いて読み込ませる。次に、ステータスファイルに書かれた DAC 値 (全て 128) を bias コマンドを用いて全バイアス電圧に設定する。その後、一度モニターを行った後にバイアス電圧の出力スイッチを全て開き、再びモニターを行う。図 94 より、設定したバイアス電圧値が出力されていることが分かる。また、図 95 に示した電圧出力地点にテスターをあて、実際に DRV ボードから出力されている電圧も確かめた所、モニター値と同様の出力が確認できた。故に、DRV ボードが任意のバイアス電圧を正常に出力していることが分かった。

```
$ db upfile /home/mirsis/status file/mirsis_SPC_stat      ←ステータスファイルの読み込み
done.
$ bias DAC default  ←ステータスファイルに書かれたDAC値を全バイアス値に設定
$ bias MON all      ←DRVボードから出力される前のDRVボードで生成したバイアス電圧をモニター

 ch    Name    Alias   Vset      DAC      Vint      OSW     Vout      Iset  Idet
                       (V)               (V)               (V)       (uA)  (uA)
-------------------------------------------------------------------------------
 00    BIAS0   VDETGRV 6.90    128 V    6.90 V    OFF     0.04 F      0     0
 01    BIAS1   VNEG    7.00    128 V    7.00 V    OFF     0.07 F      0     0
 02    BIAS2   VDETGRD 5.10    128 V    5.10 V    OFF     0.09 F      0     0
 03    BIAS3   VPW     6.00    128 V    6.00 V    OFF     0.06 F      0     0
 05    BIAS5   VCLCOL  5.50    128 V    5.50 V    OFF     0.00 F      0     0
 06    BIAS6   VCLOUT  4.10    128 V    4.10 V    OFF     0.02 F      0     0
 07    BIAS7   VGATE   3.80    128 V    3.80 V    OFF     0.00 F      0     0
 08    BIAS8   VCASUC  3.50    128 V    3.50 V    OFF     0.22 F      0     0
 09    BIAS9   VSHPCHG 3.00    128 V    3.00 V    OFF     0.24 F      0     0
 10    BIAS10  VOFFSET 2.80    128 V    2.80 V    OFF     0.24 F      0     0
 11    BIAS11  VRSTUC  2.00    128 V    2.00 V    OFF     0.22 F      0     0
 12    CSA0    IMUX    1.70    128 V    1.70 V    OFF     0.00 F    100     0
 13    CSA1    VGG2    1.90    128 V    1.90 V    OFF     0.00 F    320     0
 14    CSA2    VGG3    1.80    128 V    1.80 V    OFF     0.00 F    160     0
 15    CSA3    VGG4    1.60    128 V    1.60 V    OFF     0.00 F    160     0
 20    VCLKNO  CLKL    5.00    128 V    5.00 V    OFF     5.00 F
 22    VSSA    VSS     5.50    128 V    5.50 V    OFF     0.00 F      0     0

$ bias OSW on     ←DRVボードから外部へのバイアス電圧の出力を開始
$ bias MON all    ←DRVボードから外部へ出力されている電圧をモニター

 ch    Name    Alias   Vset      DAC      Vint      OSW     Vout      Iset  Idet
                       (V)               (V)               (V)       (uA)  (uA)
-------------------------------------------------------------------------------
 00    BIAS0   VDETGRV 6.90    128 V    6.90 V    ON      6.90 V      0     0
 01    BIAS1   VNEG    7.00    128 V    7.00 V    ON      7.00 V      0     0
 02    BIAS2   VDETGRD 5.10    128 V    5.10 V    ON      5.10 V      0     0
 03    BIAS3   VPW     6.00    128 V    6.00 V    ON      6.00 V      0     0
 05    BIAS5   VCLCOL  5.50    128 V    5.50 V    ON      5.50 V      0   -21
 06    BIAS6   VCLOUT  4.10    128 V    4.10 V    ON      4.10 V      0     0
 07    BIAS7   VGATE   3.80    128 V    3.80 V    ON      3.80 V      0     0
 08    BIAS8   VCASUC  3.50    128 V    3.51 V    ON      3.51 V      0    21
 09    BIAS9   VSHPCHG 3.00    128 V    3.01 V    ON      3.00 V      0    21
 10    BIAS10  VOFFSET 2.80    128 V    2.81 V    ON      2.81 V      0     0
 11    BIAS11  VRSTUC  2.00    128 V    2.00 V    ON      2.00 V      0    21
 12    CSA0    IMUX    1.70    128 V    1.70 V    ON      1.70 V    100     0
 13    CSA1    VGG2    1.90    128 V    1.90 V    ON      1.90 V    320     0
 14    CSA2    VGG3    1.80    128 V    1.80 V    ON      1.80 V    160   -23
 15    CSA3    VGG4    1.60    128 V    1.60 V    ON      1.60 V    160     0
 20    VCLKNO  CLKL    5.00    128 V    5.00 V    ON      5.00 V
 22    VSSA    VSS     5.50    128 V    5.50 V    ON      5.50 V      0     0
```

図 94　Si:As 320×240 array 検出器用 DRV ボードの駆動試験を行った際の端末のキャプチャ

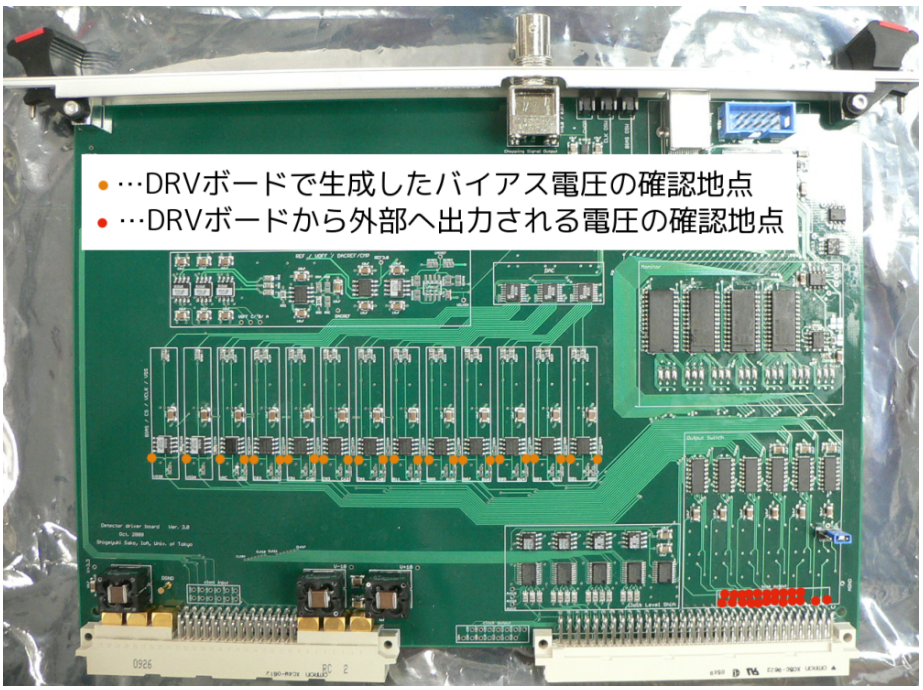

図 95　Si:As 320×240 array 検出器用 DRV ボードの電圧出力地点

6 まとめと今後の展望

6.1 まとめ

本研究では、MIRSIS の検出器系開発として撮像検出器である Si:As 128×128 array 検出器の立上げと分光検出器である Si:As 320×240 array 検出器の立上げを行った。

撮像検出器の立上げをするにあたって 2 つの駆動試験を行った。まず初めに、MIRSIS 検出器読み出し用 TAC システム (ADC、DRV 等の各ボード、計算機) の駆動試験として、MIRSIS 撮像検出器で使用する Si:As 128×128 array 検出器と同じ pixel 数の Si:Sb 128×128 array マルチプレクサの駆動試験を行った。この駆動試験の結果、TAC システムを用いたマルチプレクサの駆動に成功した。また、クロック電圧と Si:Sb 128×128 array マルチプレクサからの出力電圧の関係も確認できた。次に、MIRSIS 撮像検出器である Si:As 128×128 array 検出器の駆動試験を行った。この駆動試験を行う際、Si:Sb 128×128 array マルチプレクサの駆動試験に用いた TAC システムと同様のシステムを使用し、バイアス電圧の設定のみ Si:As 用にカスタマイズした。駆動試験を行った結果、MIRSIS 撮像検出器の駆動に成功した。その結果、検出器出力が、光、温度、積分時間で変化することを確認でき、おおよそ正常に動作していると考えられる。また、dead pixel の存在も確認された。

分光検出器の立上げとして、分光検出器用 ADC ボードの特性測定試験と、分光検出器用 DRV ボードの駆動試験を行った。ADC ボードの特性測定試験の結果、分光検出器の出力電圧範囲 (-2.5V∼-0.5V) において、ほぼ理論値通りの DC 電圧値と AC 電圧値を得ることができた。また、TAC システムを用いた DRV ボードの駆動試験も成功した。

6.2 今後の展望

分光検出器の立上げに関し、まず Si:As 320×240 array マルチプレクサの駆動を行うためのクロックソースファイルを作成する必要がある。このクロックソースファイルを作成するため、Si:As 128×128 array 検出器用 clkgen プログラムをもとに、Si:As 320×240 array 検出器用 clkgen プログラムを作成する必要がある。基本的なクロックソースファイルは、すばる COMICS で使用していたものを参考に作成すれば良い。また、撮像検出器の出力端子は 4 本であったが分光検出器の出力端子は 16 本あるので、16 出力端子用 acqdat プログラムを作成する必要がある。この 2 つのプログラムを作成した後、Function Genarator を用いた画像取得試験を行う必要がある。この画像取得試験に成功すると、Si:As 320×240 array マルチプレクサ、検出器の駆動試験を行うことができる。今後、撮像・分光両検出器とも MIRSIS に搭載して、性能評価と最適化を行う必要がある。

謝辞

指導教官である岡本美子氏には研究テーマの選定をはじめ、すばる望遠鏡の観測補助という貴重な体験を与えて頂きました。また、観測装置や天文学の基礎、今後社会人として生きていくための基本を教えて頂きましたことに深く感謝致します。ISAS/JAXA の片坐宏一氏には、赤外線検出器システムの基礎を教えて頂き、また検出器実験システムを立上げるにあたって必要となる様々な知識を親身に御指導頂きましたことに感謝致します。東京大学天文学教育センターの酒向重行氏には、検出器ソフトウェアを導入、使用方法について御指導頂き、また不具合が発生したときなど、私の質問に対してとても丁寧に答えて頂きましたことに感謝致します。茨城大学の百瀬宗武氏には、天文学の基礎を教えて頂き、また野辺山の電波観測実習という貴重な機会を与えて頂きましたことに感謝致します。宇宙観測研究室の皆さんには本研究を進めるにあたって、様々な支援をして頂きましたことに感謝致します。

最後に、大学生活を支えてくれた両親と友人達に感謝致します。

参考文献

[1] http://www.iac.es/project/CCam

[2] S.V.W.Beckwith「Near Infrared Techniques for studies of Star Formation」T.P.Pay(ed.) & S.V.W.Beckwith,Star Formation and Techniques in Infrared and mm-Wave Astronomy (Lecture Notes in Physics,vol.431,p.157-213)

[3] 家 正則・岩室史英・舞原俊憲・水本好彦・吉田道利編、「宇宙の観測 I －光 · 赤外線天文学」、日本評論社、2007

[4] 堀田厚生著「半導体の基礎理論」(技術評論社)

[5] 佐藤 圭悟、茨城大学大学院理工学研究科自然機能科学専攻修士論文、2007

[6] 真鍋 啓、茨城大学大学院理工学研究科自然機能科学専攻修士論文、2007

[7] 酒向 重行、東京大学大学院理学研究科天文学専攻修士論文、2000

[8] Raytheon CRC-774 320×240 Readout Integrated Corcuit User's Guide and Operating Manual (Si:As 320×240 array 検出器マニュアル)

[9] Rockwell MF2 Engineering Grade Multiplexer 説明文書 (Si:As 128×128 array 検出器マニュアル)

[10] DRS Technologies Device Description and Operation Guide for the DRS Technologies MF-128 BIB Hybrid Focal Plane Array (Si:Sb 128×128 array 検出器マニュアル)

[11] 酒向 重行、TAO array controller(TAC) Ver 5.0.0 developer's manual、2007

[12] S.SAKO, Y.K.OKAMOTO, H.KATAZA, T.MIYATA, S.TAKUBO, M.HONDA, T.FUJIYOSHI, T.ONAKA, AND T.YAMASHITA、「Improvements in Operating the Raytheon 320 × 240 Pixel Si:As Impurity Band Conduction Mid-Infrared Array」、2003