# 小型衛星 Cute-1.7 APD モジュールの性能試験

津布久 佳宏

卒業論文

東京工業大学 理学部 物理学科

2006 年 2 月

## 要旨

本研究室ではアバランシェフォトダイオード (APD) と呼ばれる放射線半導体検出器の宇宙での利用を目指し研究開発を行ってきた。我々は APD の宇宙動作実証のため、工学部松永研究室と共同で、超小型衛星 Cute-1.7 プロジェクトを立ち上げ、2006 年 2 月 22 日に MV-8 号機によるサブペイロードでの打ち上げに成功した。

本論文では、Cute-1.7 衛星搭載 APD モジュールのフライトモデルに対する性能試験について述べる。まず、フライトモデルに対して単体レベルの電気的動作試験を行い、全ての基板について正常な動作を確認した。そして、本研究室が確立した APD の温度に対する増幅率依存性を改善するための APD 印加電圧制御システムを Cute-1.7 衛星に対して組み込み、衛星軌道環境を想定して本システムが正常に機能するか試験を行った。恒温槽を用いて $-20$ ℃から 40 ℃まで温度サイクルを経験しても、DC-DC コンバータからの出力電圧は見積もられている誤差内に収まっており、本システムは正常に機能することが確かめられた。最後に、Cute-1.7 衛星が荷電粒子観測を行う異常磁気帯において予想される $10^6$ cts/sec という高い計数率を APD モジュールが正しく計測できるか試験を行った。入射レート $1.2\times10^6$ cts/sec に対して出力レートは 30 % 程度の数え落しが確認できたが、衛星軌道上での計数には問題なく使用できると考えられる。

# 目 次

# 図 目 次

# 表 目 次

# 第1章　はじめに

## 1.1　小型衛星プロジェクト

現代天文学では、天体から放射される様々な波長の電磁波を観測する事で天体に関する物理情報を得ている。例えば可視光領域では地上から望遠鏡を使って、直接天体を観測する事ができる。しかしながら、X線や$\gamma$線の領域では電磁波が地球大気による吸収を受けるため、地上で観測を行う事ができない。このため大気の影響を受けない高度まで、衛星や気球をあげる必要がある。

現在の開発の主流となっている衛星は、大型で1トン～数トンもの重量がある。しかし、大きな衛星の開発には非常に長い期間が必要であり、また国家規模での経費もかかる。こういった大型衛星に対して、近年関心が高まっているのが、大学レベルで行っている小型衛星の開発である。小型衛星開発は大型衛星に比べ次のような利点がある。

- 短期開発が可能 (1～2年)
- 著しく経費を軽減する事が可能
- 大型衛星には負担させられないリスクを複数の小型衛星に分散させる事で、新しい技術を積極的に取り入れたミッションを遂行する事が可能
- 大学を開発の基盤とすることで衛星の設計から打ち上げまでを通した実践的な学生教育を行える
- 小型であるため主衛星としてではなく、ピギーバック (相乗り) 衛星という形をとることで、打ち上げ機会を格段に増やす事が可能

東京工業大学でも、4年ほど前からCuteプロジェクトと名づけた小型衛星開発を行っている。

### 1.1.1 Cute-I

Cute-I は東京工業大学工学系機械宇宙システム工学専攻松永研究室が設計開発を行い、2003 年 6 月に打ち上げに成功した超小型衛星である (図 1.1)。Cute-I は、大きさ 10×10×10 $cm^3$、重さ 1 kg と世界最小クラスの衛星である。

Cute-I は、高度 820 km、軌道傾斜角 98.7 度の太陽同期軌道で周回する。Cute-I は CW(モールス符号) 送信機、FM 送信機を積んでおり、衛星の加速度、角速度、温度、姿勢のデータを保存し地上に送信する事ができる。すでに予定されていたミッションは全て成功し、打ち上げから約 2 年半経つ今も運用されている。

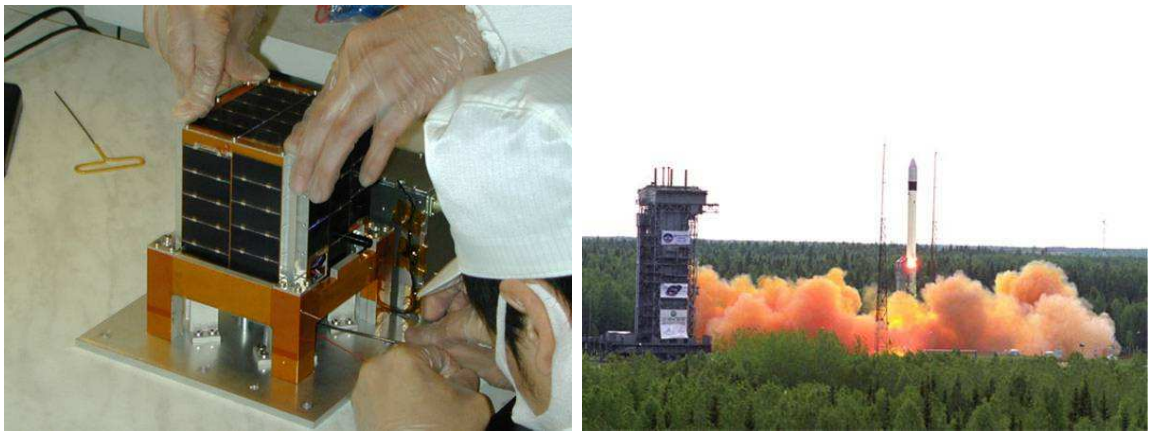

図 1.1: Cute-I の製作風景 (左)。2003 年 6 月ロシアでの打ち上げ (右)。

### 1.1.2 Cute-1.7

Cute-I の成功で小型衛星バスの製作に自信を深めた Cute-I チームと我々理学系基礎物理学専攻河合研究室のメンバーで、今度は理学ペイロード部を備えた次期小型衛星 Cute-1.7 の開発・製作に乗り出した。Cute-1.7 は次のような工学目標、理学目標をもつ。

#### 工学目標

- PDA(携帯端末) 等民生製品の宇宙動作実証
- 3 軸姿勢決定と磁気トルカを用いた軌道上制御の実証
- 将来の非デブリ化を目的とした衛星軌道離脱のための超小型テザー伸展機構の軌道上実証

- 超小型衛星用分離機構システムの軌道上実証
- アマチュア無線帯、民生通信機を用いた通信技術の確立

### 理学目標

- アバランシェフォトダイオード (APD) の宇宙動作実証
- 南大西洋磁気異常帯 (SAA) を含む放射線異常帯の軌道荷電粒子探査

アバランシェフォトダイオードは、小型で省電力な将来有望な検出器として次世代天文衛星 $NeXT$ への搭載が検討されているが、未だかつて放射線検出器としては宇宙で用いられたことがない。そこで、Cute-1.7 に搭載して世界初の宇宙動作実証を目指している。また、衛星軌道上に多く分布していることが予想されているが、未だ直接観測例のない低エネルギー荷電粒子の軌道上分布を調査することも目標となっている。このような理学目標を達成するために我々の研究室では 2006 年 2 月打ち上げを目指し、Cute-1.7 の性能試験を行ってきた。

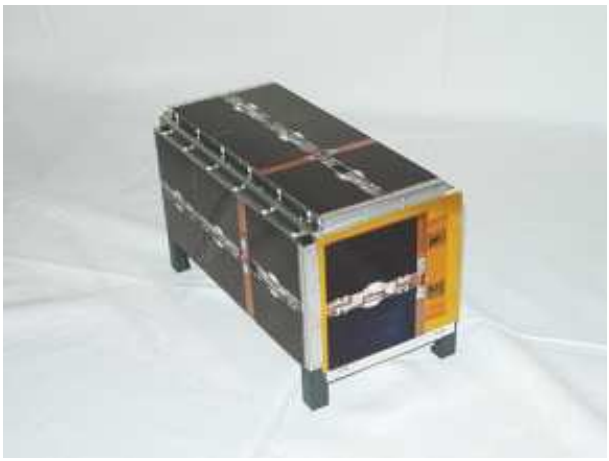

図 1.2: Cute-1.7 フライトモデル。

### 異常磁気放射線帯

地球の周りには磁気境界面と呼ばれるものがある (図 1.3)。これは、太陽からの太陽風プラズマや太陽フレア粒子、または超新星爆発などで飛んでくる銀河宇宙線に対し、地球自身の持つ磁場でそうした放射線がその内部に流入しないように保護している境界面のことを指している。この磁気境界面は、太陽側で圧縮され、反対側では尾のように広がり月軌道まで達している。

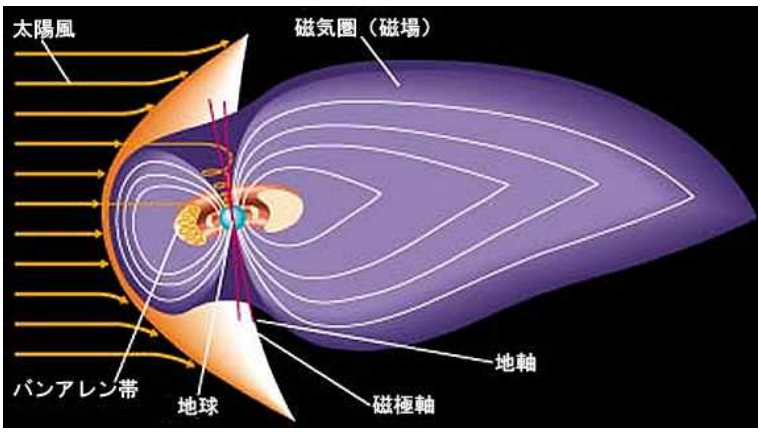


図 1.3: 地球磁気境界面。

しかし磁気圏境界内部にも、ほぼ定常的に地球磁場に捕捉された放射線帯粒子が存在している。地球の磁場は高緯度地方では磁場によるシールド効果は弱くなるため放射線帯が低緯度まで達する場合が多くなる。なお、地球近傍の磁場は双極磁場で近似することができて、地球の回転軸とは 11.5 度の差があるため、放射線帯の下限高度が緯度によって異なってくる。赤道上空を土星の環のように地球を取り囲み、平均高度が約 3,600 km で数 MeV から数 100 MeV の高いエネルギーをもった陽子で満たされた内帯と、約 18,000 km の 40 keV から数 10 MeV の電子と陽子で満たされた外帯のことを放射線帯（バンアレン帯）と呼んでいる (図 1.4 左)。内帯の一部は南大西洋の上空 300 km と極端に垂れ下がっているところがあり (図 1.4 右)、これを SAA（南大西洋異常帯 : South Atlantic Anomaly）と呼ぶ。高緯度を中心とした帯状領域には荷電粒子 (主に電子) が地表近くまで達し、オーロラ帯と呼ばれる特異な領域を作っている。図 1.4 から非常に高いフラックスが SSA やオーロラ帯に集中していて、一番高い領域ではおよそ $10^{6}$cts/s/cm$^2$ となっていることが読み取れる。

SAA やオーロラ帯などの高いフラックスを持っている領域は、衛星電子機器の放射化による損傷や検出器の放射化など、様々な問題が起きる領域として細心の注意を払う必要がある。例えば、光電子増倍管や比例計数管など、1000 V を超える高電圧を必要とする検出器は、装置自体が放電により壊れてしまう危険があるため、検出器の電源を落とす必要がある。このため SAA の詳細な観測には制約が多かった。そこで我々は、検出器の動作実証を兼ねて、これまでほとんど例にない、SAA を含む 30 keV 以下の低エネルギー荷電粒子探査を行うことにした。30 keV 以下のエネルギー領域はフォトダイオードでは回路雑音に埋もれてしまうエネルギー領域で、高利得の APD を用いて初めて計測が可能と

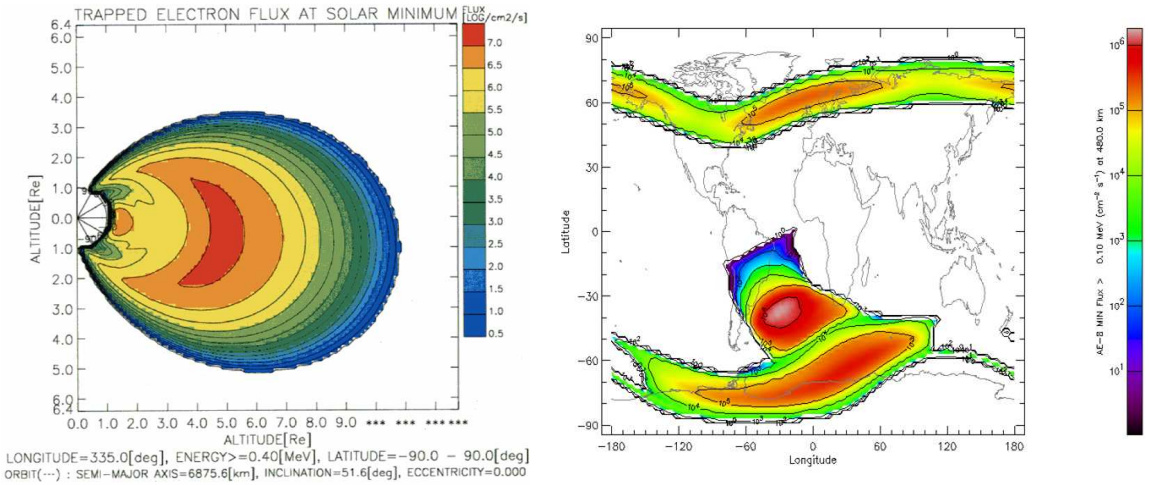


図 1.4: (左) 太陽極小期における地球磁場に捕捉された荷電粒子 (電子 : E≥400 keV) 分布。高度 1.5 Re(Re は地球半径) 付近の赤い部分が内帯、5 Re 付近が外帯。(右) 南大西洋磁気異常帯 (South Atlantic Anomaly)。低緯度ブラジル南下に存在するのが SAA で、高緯度地方に存在するのはオーロラ帯である。

なる。30 keV 以下の低エネルギー粒子のフラックスは、100 keV の時に比べさらに高いと考えられる。荷電粒子の冪乗分布を仮定した計算では、約 $10^8$ cts/s/cm$^2$ 以上と見込まれる。今回 Cute-1.7 に搭載する APD の受光面は 3.5 mm$\phi$ であるため、$\sim 10^6$ cts/s 以上の割合で入射する粒子を測定する必要がある。

## 1.2 本論文の目的

理学系河合研究室では 2 年前の Cute-1.7 プロジェクトの立ち上げ時から工学系松永研究室と密に連携をとりつつ APD モジュールの開発・製作、そして試験を行ってきた。今年度は打ち上げの年にあたり、主にフライトモデルに対する試験が中心となっている。試験内容は、(1) 各基板に対して電気的動作を試験する単体電気試験、(2) 宇宙環境を想定しての環境試験、(3)APD の増幅率を一定に保つ印加電圧制御システムの動作試験、(4) アバランシェフォトダイオードが荷電粒子帯などでどこまでのレートを正確に計測できるかを試験する高計数試験、(5) 衛星バスとの噛み合わせを試験する理工学統合試験がある。私はそのうち、(1) の単体電気試験、(3) の印加電圧制御試験、(4) 高計数試験を特に担当したので、以下この試験についての報告を行っていくことにする。

# 第2章　Cute-1.7 APDモジュールの概要

## 2.1　APDモジュール諸元

Cute-1.7 衛星のサイズは 10×10×20 $cm^3$ であり、その約半分が APD モジュールに割り当てられている。固定用の治具を取り付けるスペースなどがあるため、実際に使用できるスペースは 8×8.5×7.5 $cm^3$ 程である。電力については、衛星の太陽電池によって約 3W の発電が可能であり、そのうち APD モジュールは 1.5 W 使用できるようになっている。しかし、APD モジュールの電圧変換効率を考えると、さらに少ない電力でミッションを遂行しなければならない。表 2.1 に APD モジュールへ分配した設計電力値について示す。

| | 荷電粒子観測部 | 理学制御部 | 合計 (最大使用可能電力) |
|---|---|---|---|
| 設計電力 | 700 mV | 300 mV | 1000 mV (1500 mV) |

表 2.1: APD モジュールの電力設計要求。

## 2.2　アバランシェフォトダイオード

本衛星には、アバランシェフォトダイオード (APD) と呼ばれる放射線半導体検出器を搭載する。この検出器は高い増幅率をもつ光電子増倍管 (PMT) と、量子効率が高く、応答速度が早いといったフォトダイオード (PD) 両方の長所を兼ね備えている (表 2.2)。さらに非常に軽量でコンパクト、そして電力消費が少ないため、小型衛星のような容積やリソースの限られた環境でも使用可能となっている。

APD は半導体内部に高い電場を持たせることによって半導体内部に増幅機能を備えた半導体検出器である。可視光や X 線、$\gamma$ 線といった放射線によって生成された電子正孔対は、APD 内部の電場によって十分に加速を受け、電極に到達するまでに多数の価電子帯

| | 光電子増倍管 (PMT) | フォトダイオード (PD) | アバランシェフォトダイオード (APD) |
|---|---|---|---|
| 量子効率 | $\leq 25$ % | $\geq 80$ % | $\geq 80$ % |
| 増幅率 | ○ (あり) | × (なし) | ○ (あり) |
| 印加電圧 | $\sim 1000$ V | $\leq 100$ V | $\sim 300$ V |
| 容積 | × (大) | ○ (小) | ○ (小) |
| 回路雑音の影響 | ○ (小) | × (大) | △ (大) |
| 磁場の影響 | × (大) | ○ (小) | ○ (小) |
| 構造 | × (複雑) | ○ (単純) | ○ (単純) |

表 2.2: シンチレーション検出器の比較。

の電子を伝導帯に叩き上げて信号をなだれ増幅させる。信号を検出器内部で増幅すれば回路内で発生する雑音を相対的に小さく抑えられるため、通常のフォトダイオードよりも遥かによい S/N 比が得られる。APD の性質を特徴づける増幅率、暗電流、増幅ゆらぎは素子内部の電場に依存するため、APD 素子の構造によって性能が異なる。

また APD には電圧依存性と温度依存性があり、それぞれの変化に対して APD の増幅率が変化してしまう性質をもっている。

#### 増幅率の印加電圧依存性

APD の増幅は電場が強い領域で起こり、その電場の強さは印加電圧に依存するため、増幅率と暗電流は印加電圧に依存する。増幅率の印加電圧による変化率は、APD の種類によって異なるが、reverse 型の APD S8664-55 に関しては温度 20 ℃、増幅率 50 のときに式 (2.1) に従う [1]。

$$\frac{1}{G} \cdot \frac{dG}{dV} \cong 3.4[\%/\mathrm{V}] \tag{2.1}$$

#### 増幅率の温度依存性

APD は衝突電離を起こしてキャリアをなだれ増幅する。APD を冷却すると、電子が半導体の原子核と衝突する頻度は少なくなることによって平均自由工程が長くなり、さらに加速を受ける。従って、常温よりも低い印加電圧で十分な増幅率が得られるようになる。増幅率が 50 になるために必要な電圧は、20 ℃では 340~350 V であるのに対し、−20 ℃で

は310~320 V である。reverse 型 APD に関して、電圧を一定にしたときの増幅率 $M = 50$ となる点における増幅率の温度依存性は、−20~ −10 ℃において、

$$\frac{1}{G} \cdot \frac{dG}{dT} \cong -2.6[\%/\mathrm{K}] \tag{2.2}$$

となることが知られている [1]。

### 2.2.1 reverse-type APD

reverse-type APD は、他の APD に比べてシンチレーション光の検出に特化するよう改良されたもので、表面から 5 $\mu$m 程度の深さに狭い増幅領域が存在する。また、reverse-type APD は空乏層の暑さが ~40 $\mu$m と薄く、300 V 程度の低い電圧を印加するだけで十分な増幅率を得ることができる。空乏層が薄いと容量が大きくなるが、増幅領域の奥に n 型半導体を挿入して容量を小さくすることで印加電圧の変動に対する安定性を持たせ、かつプリアンプの雑音特性を低減させている。一般的なシンチレーション光の出力波長は 550 nm よりも短く、表面から 1~3 $\mu$m の領域で電子正孔対に変換されるため、シンチレーション光のほぼ全てが増幅領域の手前で電子に変換され、完全に増幅される。

この構造では完全増幅を起こす暗電流は正孔のみで、暗電流の主成分である熱電子はほとんど増幅を受けない。結果として正孔はほとんど増幅されず暗電流を非常に低く抑えることができる。そのため、reverse-type APD は他の種類の APD よりも雑音レベルが低く、低エネルギー領域における性能が非常に優れているので、我々は特にこのタイプの APD の宇宙動作実証を行う。

## 2.3 APD モジュール

APD モジュールは 4 つの基板で構成されている。その全体像は図 2.1 の通りである。各基板には様々な機能を持った IC が搭載されている。アナログ基板は APD からのアナログ信号を増幅・整形し、デジタル基板はそのアナログ信号の計数を行う。電源基板は、衛星バスから供給されるメインバス電源と 3.3 V 電源を+5 V、−5 V、+12 V に変換して各基板に供給することができる。そしてマザー基板は高圧電源を搭載していて、APD に必要となる高電圧を作り出すことができる。各基板についての詳しい説明と搭載されている機能についての説明を以下で行う。

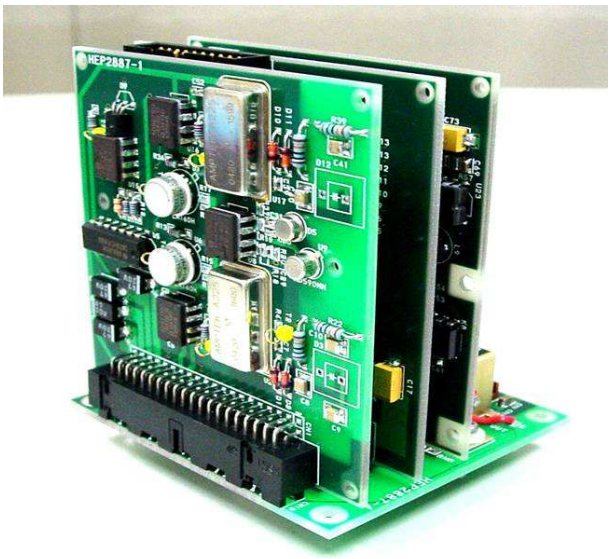

図 2.1: APD モジュールのフライトモデル基板。手前 : アナログ基板、中央 : デジタル基板、奥 : 電源基板、下 (土台) : マザー基板。

### 2.3.1 アナログ基板と APD

アナログ基板には、APD の信号をアナログ信号処理を行う IC が多数搭載されている。

#### APD と前置増幅器 (プリアンプ)

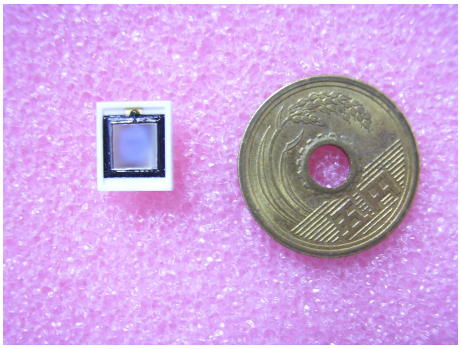

図 2.2: APD モジュールに搭載する Al 蒸着された reverse-type APD。

アナログ基板には上で説明した理学検出器 reverse-type APD が搭載されている。この APD は S8664-55mod と呼ばれるもので、面積は 5×5 $mm^2$ となっている。APD の受光面には、可視光や X 線などの光に反応しない様に Al 蒸着が施されているが、後の試験で

APD に光が洩れ込んでいることが発見されたため、遮光用として APD 表面を覆う治具を取り付けることにより光洩れ問題を解決した (付録 A.1 節)。受光面は 3.5 mm$\phi$(0.096 $cm^2$) となっている。

この APD に入射した荷電粒子は後段の回路でアナログ信号処理されるようになっている。その原理はまず、高圧電源でバイアス電圧をかけることにより APD に電場を作り出し、素子内部で増幅を行う。増幅率は G = 30~100 程度である。増幅されたアナログシグナルが出てくると、AmpTek 社製 MIL 規格前置増幅器 (プリアンプ)A225 の入力の前にあるカップリングコンデンサによって直流高電圧がカットされ、交流成分のみが通ることができるようになっている (図 2.3)。そして、その信号電流が A225 によって積分される。このように電荷を積分するため、非常に良い S/N 比が得られる。しかし、積分した電荷を放電するするため A225 の時定数はおよそ 2.4 $\mu$sec というように、時間応答が遅いという欠点もある。その問題は後段回路の微分器によって解決される。+5 V、−5V の直流電源に接続されているダイオードはクランプダイオードと呼ばれるもので、これは衛星の電源電圧の突然の電圧降下などの理由でカップリングコンデンサに溜められていた電荷が大電流となって A225 に流入して破壊することを防ぐ役割がある。

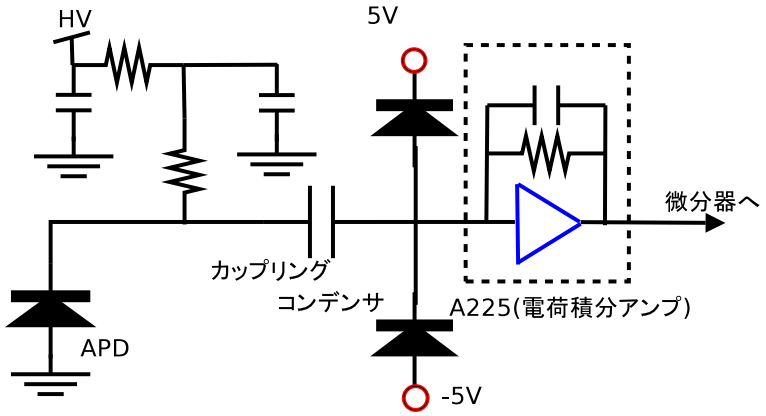


図 2.3: 検出器と前置検出器 (プリアンプ) を含む前段信号の回路図。(図中のプリアンプは等価回路)

アナログ基板には信号 2 系統が搭載され、両系にそれぞれ同じ性能・特性をもった APD が搭載されている。こうすることによって片方の系に不足の事態が生じても、もう片方の系で計測を継続できるようになっている。

### ゲインアンプとコンパレータ

前置増幅器 A225 によって積分増幅されたアナログ信号は、さらに Analog Device 社製 OP アンプ AD827 によって微分増幅、そして増幅率 10 で反転増幅される (図 2.4)。これらを総称してゲインアンプと呼んでいる。プリアンプにより高速応答性が劣化しても、このゲインアンプで微分することにより、高速応答性を回復している。しかし、そのまま A225 からのシグナルを微分してもアンダーシュートが起きてしまう。これを改善するために、微分器には帰還コンデンサを挿入して、微分・積分を行ってから反転増幅を行うことにより、アンダーシュートを緩和させている。ゲインアンプの時定数はおよそ 10 nsec となるので、高速応答性は失われていないことになる。

さて、増幅されたアナログ信号はコンパレータ (National Ssemiconducter 社製 LM160H) へと送られる。このコンパレータには、スイッチング IC(Philips 社製 74HC4066) によって 4 つのスレッショルドレベル電圧を与えることができる。図 2.4 にスイッチング IC とコンパレータ周りの回路図を示す。図中に記されているスイッチング IC は THD1、THD2 でそれぞれ 3.9 kΩ、100 Ω の抵抗の接続を行う。そして作り出されたスレッショルド電圧はボルテージフォロワを介し、コンパレータへとつながれている。このボルテージフォロワは、出力インピーダンスを下げる役割がある。多くのセンサや抵抗・コンデンサといった受動部品を組み合わせた回路は、それに続く回路で電流を吸ってしまうと回路の特性に影響を与えてしまう。そのため、なるべく電流を流すことなく、電圧だけを後段回路に伝える目的で使われている。

表 2.3 にはスイッチング IC の 2 つのスイッチを on、off した場合の 4 つのスレッショルドレベルを示す。またその電圧値をエネルギー換算した値も同表に示す。異常放射線帯に分布している様々なエネルギーをもった荷電粒子の探査を、表 2.3 に与えた 5 つのエネルギーレンジで計測することができるようになっている。

| THD1 | THD2 | 電圧値 [mV] | 換算エネルギー [keV] |
|---|---|---|---|
| off | off | 241 | 10.1 |
| on | off | 710 | 29.3 |
| off | on | 2402 | 98.5 |
| on | on | 2499 | 102.4 |

表 2.3: 常温時におけるコンパレータに入力されるスレッショルドレベル電圧と APD の増幅率を 50 と仮定したときの換算エネルギー。

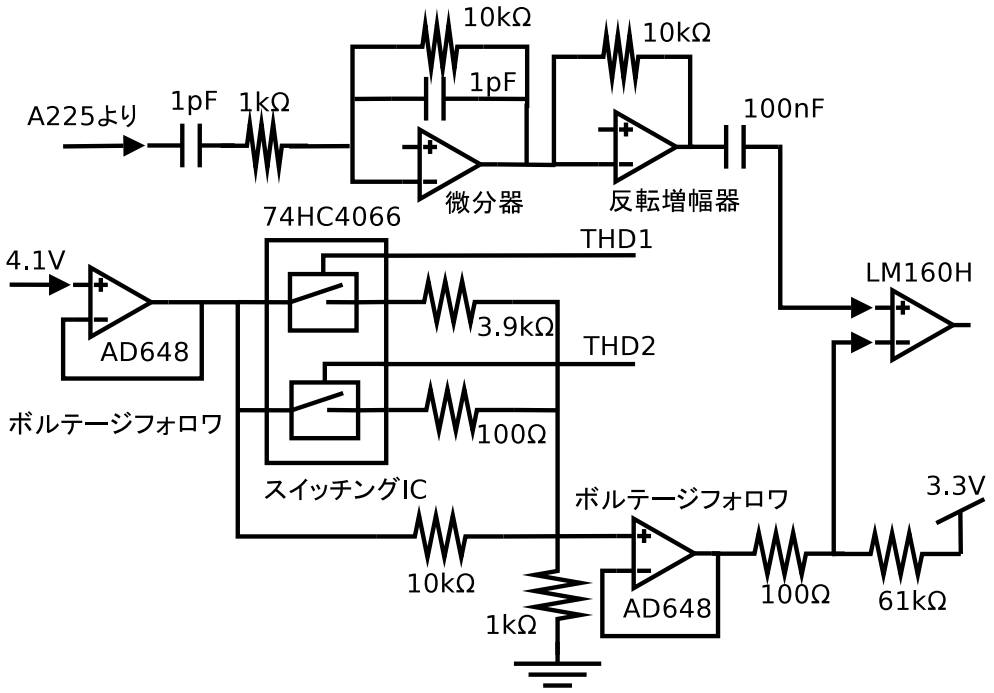


図 2.4: ゲインアンプ、スイッチング IC(74HC4066)、コンパレータ (LM160H) 周りの回路図。スイッチング IC は等価回路で、THD1、THD2 でスイッチの開閉を行い、回路の接続を行う。

**温度センサ AD590**

アナログ基板には Analog Devices 社製 AD590MH という温度トランスデューサ IC が搭載されている [3]。APD の増幅率に温度依存性があるため、その温度を知る必要があるからである。この IC は+4∼+30 V で電源電圧で動作し、−55∼150 ℃の範囲の温度に対して線形な電流を出力し、1$\mu$A/K に対応する高インピーダンスの定電流レギュレータとして動作する。この温度センサは、ツェナーダイオードによって 8.2 V で制御されている。そして、温度センサからの出力を Analog Devices 社製 AD648SQ で非反転増幅を行い、デジタル基板に搭載されているマイクロコンピュータ (H8 マイコン) へとアナログ信号が送られる。

### 2.3.2 デジタル基板

この基板はアナログ基板によりアナログ処理された信号をデジタル信号に変換し、その情報を衛星バスに送ることができる。さらに、APD モジュール全体をデジタル制御することができるようになっている。これらのデジタル処理はマイクロコンピュータ H8(RENESAS

TECHNOLOGY 社製 HD64F3048BVF25、以下 H8 マイコンと略する) で行っている。H8 マイコンにはいくつかの性能が備わっている。以下にその主な性能について説明を行う。またこのデジタル基板には、USB 通信を行うために、と呼ばれる USB デバイス用コントローラ IC(National Semiconductor 社製 USBN9604-28M) が搭載され、さらに直に理学 H8 と工学 H8 とをつなぐシリアル通信も可能である。これらより、APD モジュールの様々な情報を衛星バスに送受信することができるようになっている。以下、H8 マイコンの主な性能と 2 つの通信方法ついて説明を行うことにする。

### マイクロコンピュータ H8 (Renesas Tecnhology 社製 HD64F3048BVF25)

#### [1]A/D コンバータ (ADC)

ADC(Analog-degital converter) はアナログ・デジタル変換器と呼ばれるもので、アナログ信号を 8 ビットのデジタル信号に変換する。この ADC は主に温度センサ AD590 からのアナログ信号をデジタル変換するのに用いられる。ADC のダイナミックレンジは 3V なので、温度センサ出力に対する分解能は、$3.0[\mathrm{V}]/256[\mathrm{ch}] = 12[\mathrm{mV/ch}]$ となる。

#### [2]D/A コンバータ (DAC)

DAC(Digital-Analog Converter) はデジタル・アナログ変換器と呼ばれるもので、8 ビットのデジタル入力を 0∼400V の電圧に変換する。これは APD にバイアスをかける DC/DC コンバータの電圧制御に用いられる。

#### [3] カウンタ

カウンタはアナログ基板のコンパレータからの TTL 出力を 16 ビットで計数測定することができる。しかし $10^6$ cts/sec 以上の計数が H8 カウンタに直接入ってきた場合、16 MHz で動作する H8 では数え落しが多くなってしまう。そこで前段に 100 MHz まで 4 ビットで計測可能なカウンタ IC を置き、高計数のうち下 4 桁を前段回路のカウンタ IC で数え、5 桁目以上を H8 カウンタで数えることで高計数に耐えうるようにした。

### USB 通信とシリアル通信

理学 H8 と衛星バスの携帯端末 PDA とのデータの送受信は USB1.1 規格によってデジタル基板搭載の USBN9604 を介して行われる。特徴としては、12 Mbps もの転送速度に

対応しており、多様なマイコンインターフェースを備えている。内部には、クロック発生器が搭載されていて、24 MHz の外部クリスタル発振から内部で 2 倍の 48 MHz のクロックを生成している。

またシリアル通信により、理学 H8 と工学 H8 とのデータの送受信を直接行うことができる。これは RS232 規格になっており、データの伝送はシングルエンド (片線接地) 方式を採用している。このシングルエンド方式とは、1 つの信号経路に 1 本の専用線を使用し、グラウンド線を各経路共通に使っている。この方式は線が少なくて済むが、他の信号や外部のノイズの影響を受けやすいという欠点が存在している。

衛星において、オンボードコンピュータと周辺装置との通信には通常シリアル通信を用いるのが主流であったが、本衛星では初めて民生の携帯端末 PDA を搭載したということもあり、USB 通信を用いる。但し、PDA に不具合が生じても PDA を用いずに直接工学 H8 との通信ができるようにシリアル通信をバックアップ手段として準備している。

### 2.3.3 電源基板

電源基板は、衛星バスから送られる 3.3 V バス、メインバスから DC-DC コンバータ (Linear Technology 社製 LT1111MJ8) を介して+5 V、−5 V、そしてもう 1 つの DC-DC コンバータ (Linear Technology 社製 LT1027MJ8) を介して+12 V を作り出すことができる。図 2.5 には電源基板の概念図を示す。 図 2.5 に示すように、LC filter によって電源の

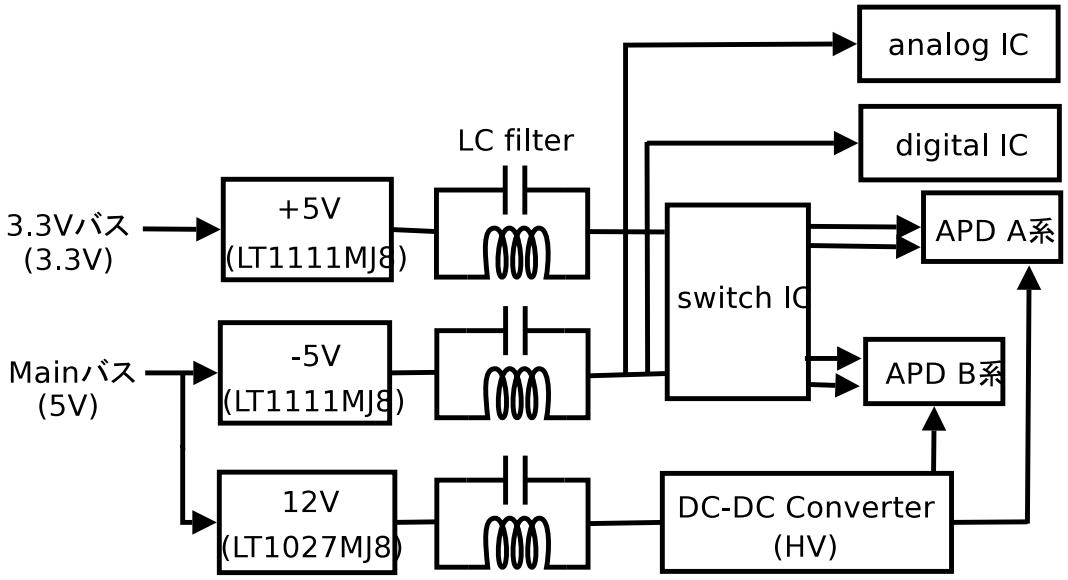


図 2.5: 電源基板の概略図。衛星バスからのメインバス、3.3 V バスから+5 V、−5 V、+12 V を作り出している。LC filter は等価回路。

リップルノイズを低減する工夫がなされている。LC filter とはリアクタンスが低周波になるほど大きくなる性質を持っているコンデンサと、逆にリアクタンスが高周波になるほど大きくなる性質を持っているインダクタを組み合わせることによって、ある特定の周波数のみを通し、低周波及び高周波ノイズを除去するものである。+5 V、−5 V の出力電圧は LC filter を通りアナログ基板の A 系 B 系の切替えを行うスイッチング IC(NAIS 社製 AQV212) に送られる。この切替えは photoMOS リレーによって行われ、後段のアナログ基板、デジタル基板に電源電圧が供給される。photoMOS リレーはフォトダイオードを用いて光接続になっており、チップ内での電圧降下が非常に小さいという特徴がある。

### 2.3.4 マザー基板

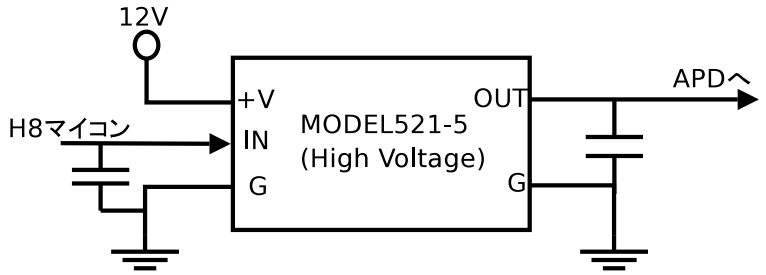


図 2.6: マザー基板の回路図。DC-DC コンバータ (MODEL521-5) は 12V で動作。

この基板は他の 3 つの基板の土台になっているだけではなく、衛星バスと連絡するためのコネクタ、そして APD にバイアス電圧を供給する DC-DC コンバータ (Analog Modules 社製 Model521A-5) が搭載されている (図 2.6)。Model521A-5 は軍用規格で作られていて、非常にコンパクトで消費電力が低いので小型衛星に適していることから Cute-1.7 に搭載することにした。この DC-DC コンバータは電源電圧として 12V を使用し、入力直流を約 120 倍に昇圧することができる。スペックシートによると、動作保証温度は −40 ℃から 85 ℃で、0 V∼5 V の入力電圧に対して、出力誤差 0.3 % 以内の線型性を保って 0 V∼600 V を出力することができる [4]。APD モジュールが DC-DC コンバータに要求する電圧領域は、200 V∼430 V であるので DC-DC コンバータの出力範囲はこれを満たしている。

図 2.7 にはマザー基板に搭載したものと同タイプの HV を単体で使用して、常温下で入力電圧に対する出力電圧の線形性の試験を行ったときの結果を示す。図より、出力の線型性は非常によく保たれていて、要求電圧領域において誤差は 0.3 % 以内に収まっていることが分かる。

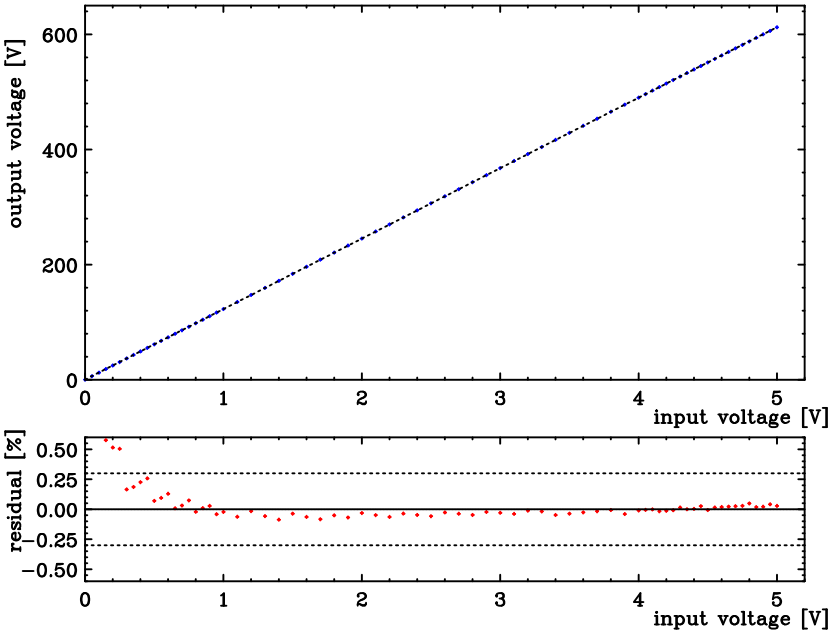


図 2.7: 入力電圧 [V] に対する出力電圧 [V] と出力誤差 [%]。

# 第3章　フライトモデル基板単体電気試験

我々はAPDモジュールの4つの基板に対して、外部環境に左右されずに動作するような回路の開発・設計を行い、また各基板に搭載するICの放射線耐性試験によるICの選定・性能評価といった様々な試験を行ってきた [2]。そして2005年10月に、フライトモデル基板が完成した。

## 3.1　目的

フライトモデルの4つの基板に対して正常な動作が保障されなければならない。そこで、それぞれの基板に対して単体電気試験を行い、各種動作確認を行った。

## 3.2　試験施設

フライトモデル基板を取り扱うにあたって、ほこりや汚れなどが故障の原因となる場合があるため、その環境を清浄に保たなければならない。我々はフライトモデル開発のため新しくクリーンブースを設置し、本試験を行った (図 3.1)。

図 3.1: クリーンブースの内部 (左) と外部 (右)。

## 3.3　各基板に対する試験結果

4つのフライトモデル基板に対して動作試験を行った。以下試験を行った順に結果と考察を述べていくことにする。

### 3.3.1　電源基板単体試験

電源基板は衛星バスから供給される3.3 Vバス・メインバスから安定した電源電圧を出力しなければならない。本試験は電源基板に対するそのものの試験、性能評価が目的であるため、電源基板のみで試験を行い、各電圧系の出力には電源系が供給しなければならないほぼ等価な負荷をかけた状態で、+5 V、−5 V、12 Vの各電源系の出力電圧が降下しないことを確かめる。

また電源基板は様々な雑音の発生源となっている。雑音にはいくつか種類があるが、電源線を通して伝播する伝導雑音、電磁波となって伝播する放射雑音などがある。電源系に雑音があると、浮遊容量及び電源出力線を介して各基板に搭載されているIC内に入り込み、能動回路によって増幅される。このように増幅された望ましくない信号は回路において雑音となり、デバイスの雑音性能を低下させてしまう。これらの発生源である電源電圧出力線の雑音についてのレベルを調べ、かつ雑音に敏感な回路への影響を見積もる。

#### セットアップ

試験回路を図3.2に示す。通常電源基板に供給される電源電圧は、衛星バスから供給される3.3 Vバスとメインバスの5 Vであるが、本試験ではバスレベルの最低時を想定し、メインバスとして3.3 Vを供給した。また、この試験は電源基板のみで試験を行ったので、図3.2で示されているように12 V電源の後段に580 Ω、5 V電源の後段に240 Ω、−5 V電源の後段に100 Ωの負荷抵抗を挿入することにより後段回路を模擬した。試験は常温・常圧下で行った。

#### 試験結果と考察

電源基板の電源電圧出力と電力についての結果を表3.1と表3.2に示す。電圧の値は図3.2のテストポイントをテスターを用いて計測した。また、供給電流量は安定化電源から読み取った。表3.1に示すように、バスレベル最低時においても2 %以内の誤差で規定の電源電圧は出力されており、心配された各電源系の電圧降下は見られなかった。また表

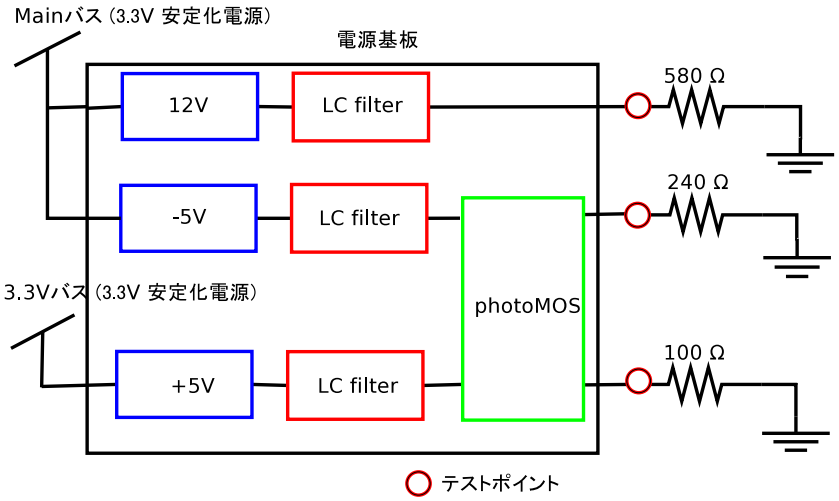


図 3.2: 電源基板単体試験セットアップ。電源基板からの出力値はテストポイントをテスターを用いて読み取り、電流値は安定化電源より読み取った。

| 電源系 | +5 V | −5 V | 12 V |
|---|---|---|---|
| 出力値 [V] | 4.90 | −5.04 | 11.92 |

表 3.1: 等価負荷をかけたときの電源電圧出力。

3.2 には両供給バスの供給電流値と消費電力値を示す。3.3 V バスは 5 V 電源を作り出し、メインバスは −5 V 電源と 12 V 電源を作り出している。そのうち 5 V 電源、−5 V 電源を作り出している DC-DC コンバータ (Linear Technology 社製 LT1111) は、スペックシート [7] によると供給電圧 3.3 V 時において負荷電流 180 mA 程度まで安定した電圧を出力できる。表 3.2 に示す通り、5 V 系は 110 mV、−5 V、12 V の両系を作り出すメインバスは合計して 160 mA を供給していることから、3.3 V バス、メインバスからの供給電流値よりも小さい電流が各 IC に流れていることになる。つまり、この電源系については安定した電圧が出力されることが保証され、かつ要求されている負荷電流がドライブされてい

| 供給バス | 供給電圧 [V] | 供給電流 [mA] | 消費電力 [W] |
|---|---|---|---|
| 3.3V バス | 3.3 | 110 | 0.36 |
| メインバス | 3.3 | 160 | 0.53 |

表 3.2: 等価負荷時における電源基板の消費電力。

ると考えられる。12 V 系でも正常に電圧が出力されていることから同様のことがいえる。

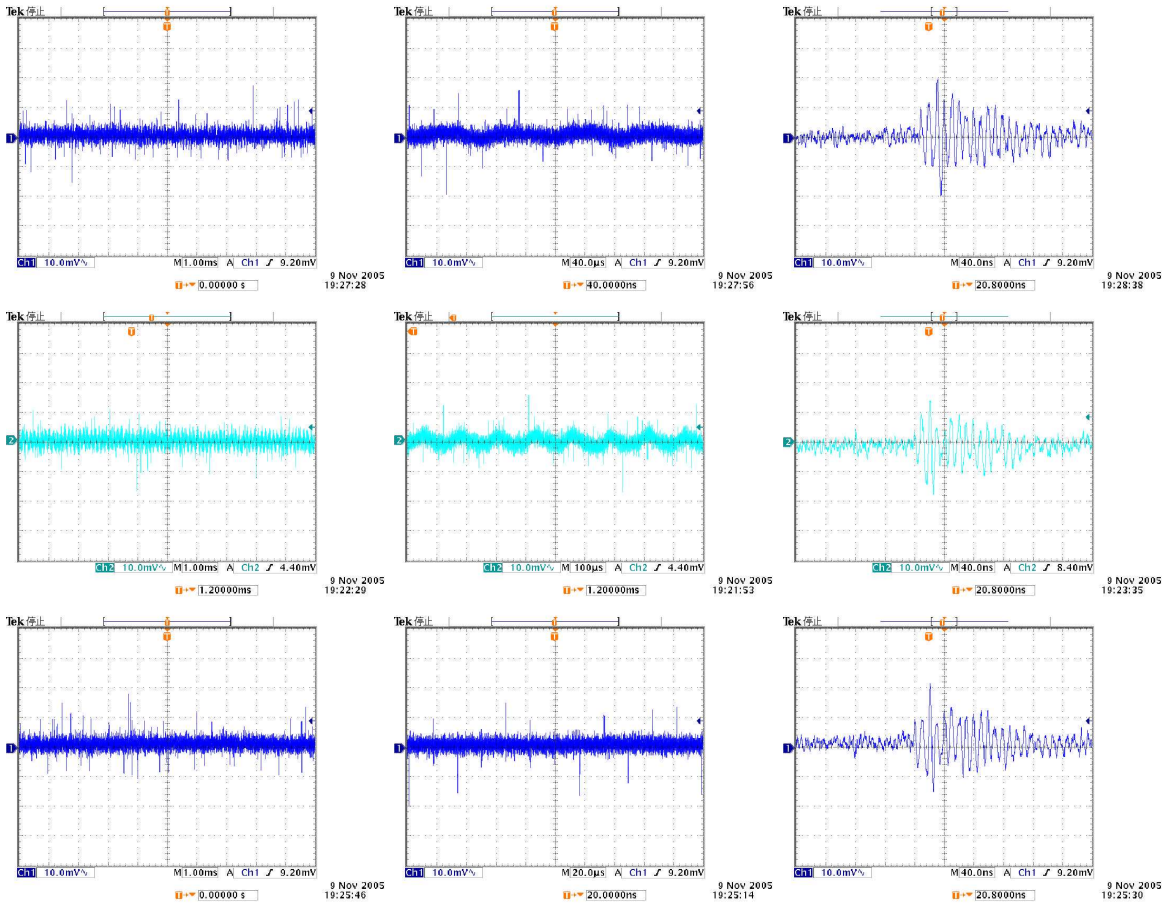

図 3.3: 電源電圧出力上のノイズ (時間スケールを変えて 3 種類測定したもの)。全ての図は縦スケールが 10 mV となっている。横のスケールは 1 列目が 1 msec、2 列目が 40 nsec、3 列目が 40 nsec。上段：5 V、中段：−5 V、下段：12 V。

図 3.3 には、電源基板から出力される雑音の代表的なものを示す。測定方法は電源電圧出力を測定したときと同様にテストポイントをオシロスコープで見た。時間スケール 40 nsec でそれぞれの系に見える雑音は、電源をオフの状態においても見られることから、外来雑音であると考えられる。5 V 系、−5 V 系出力にはともに周期 100 $\mu$sec 程度の雑音が見られ、振幅は大きなものでおよそ 20 mV 程度である。12 V 系ではこの周期の雑音は見られなかった。この 100 $\mu$sec という周期をもつ雑音は電源基板由来の雑音であると考えられるが、通信系で用いられるモールス信号 (CW) による雑音 (200 mV 程度) の約 1/10 である。さらに電源から出力線に及ぼすとされる電源雑音の影響は、OP アンプに対する電源電圧除去比 (PSRR) によって見積もることができる [5]。PSRR は、電源電圧の振れや雑音に対する耐性の指標となっていて、電源電圧の変化 ($\Delta$supply) に対する入力オフセッ

ト電圧の変化 (ΔOffset) の比を示したもので dB(デジベル) を単位とし、式 (3.1) のように表される。

$$\mathrm{PSRR}\ [\mathrm{dB}] = 20 \log \left( \frac{\Delta \mathrm{Offset}}{\Delta \mathrm{Supply}} \right) \tag{3.1}$$

PSRR は図 3.4 のように周波数の関数として表される。図 3.4 より、+5 V 電源から雑音が入り込んだ場合、雑音の周波数が 10 kHz(周期 100 $\mu$sec) でおよそ 80 dB、100 kHz(周期 10$\mu$sec) でおよそ 60 dB となっている。つまり、80 dB で 1 万分の 1 に雑音は減衰し、AD827 から出力される雑音は 0.01 mV 程度となり、60 dB の場合には雑音出力は 1 mV 程度まで減衰されることが理解できる。つまり、雑音が電源線からアナログ処理を行う回路に入り込んだとしても、十分減衰することが分かる。

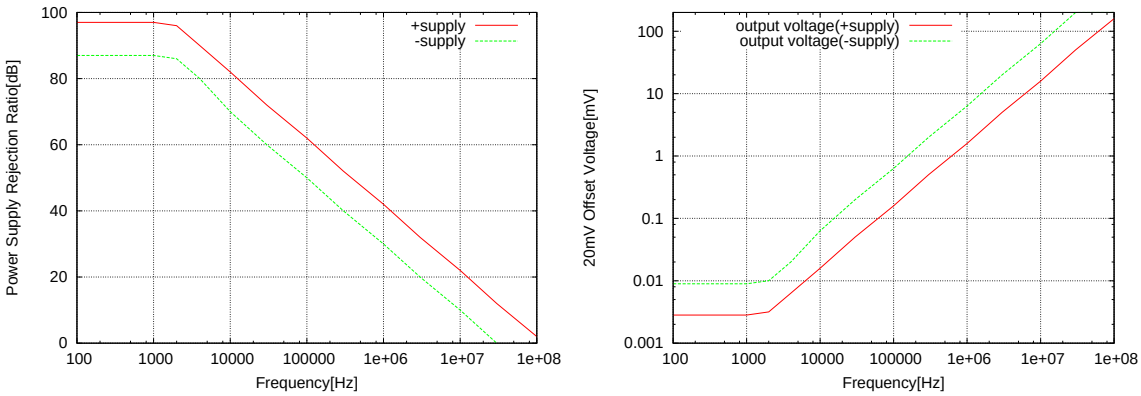


図 3.4: 上図 : 周波数雑音 [Hz] に対する PSRR[dB] と出力雑音 [mV]。下図 : 20 mV の導電雑音が AD827 によって増幅されたときの周波数 [Hz] に対する出力電圧 [mV]。+supply : AD827 の+5 V 電源から雑音が入り込んだ場合。−supply : −5 V 電源から雑音が入り込んだ場合。

### 3.3.2 デジタル基板単体試験

デジタル基板の試験は、シリアル通信、USB 通信を用いて各コマンドが正常に実行されるかどうかをみるものである。APD モジュールは全てこのデジタル基板で制御されるため、デバッグが必要不可欠となっている。

試験方法は、アナログ基板、デジタル基板、電源基板、そしてマザー基板を組み合わせて行った。アナログ基板は A 系・B 系の冗長切替え、スレッショルドレベルの切替えの確認のため組み合わせた。電源基板は各電源電圧の供給を行い、マザー基板は DC-DC コン

バータ (HV) の出力電圧の制御を行うために組み合わせた。デジタル基板へのコマンド入力は Windows マシンを用いてそれぞれ行った。

#### 試験結果と考察

| コマンド名 | 機能 | シリアル通信 | USB 通信 |
|---|---|---|---|
| 0x50 | 全データリクエスト | ○ | ○ |
| 0x70 | カウンタデータリクエスト | ○ | ○ |
| 0x36 | APD 冗長切替え (A 系→ B 系) | ○ | ○ |
| 0x37 | APD 冗長切替え (B 系→ A 系) | ○ | ○ |
| 0x30-33 | スレッショルドレベル切替え | ○ | ○ |
| 0x32,0x32,0x- | HV 設定 (瞬間的に上昇) | ○ | ○ |
| 0x33,0x33,0x- | HV 設定 (徐々に上昇) | ○ | ○ |

表 3.3: 試験したコマンド。○は確認、成功を表す。

試験結果を図 3.3 に示す。全てのコマンドは正常に実行された。一部、HV 出力の線型性に問題が生じたが、この点に関しては後述することにする。

### 3.3.3 アナログ基板単体試験

アナログ基板は APD からのシグナルをアナログ処理するための系であるが、この基板は APD からの微弱な出力信号処理を行うがゆえ、雑音に対して非常に敏感である。本試験では、本来のアナログ信号が雑音に埋もれないこと、また波形がくずれないことを確認する。

#### 試験結果と考察

まず、スレッショルドレベルを測定した。結果を表 3.4 に示す。換算エネルギーは APD の増幅率が 50 のときのコンパレータ手前の電圧値から換算した値である。4 つのスレッショルドレベルには表 2.3 とのズレが確認できる。2.3.1 に前述したように、4.2 V の入力電圧をスイッチング IC(74HC4066) を介するこによって分割し、抵抗値を変えてスレッショルド電圧値を設定している。これらにつながっている抵抗に存在する ∼10 % の誤差が、

| THD1 | THD2 | コマンド | 測定電圧値 [mV] | 換算エネルギー [keV] | 目標値 [keV] |
|---|---|---|---|---|---|
| off | off | 0x30 | 219 | 9.2 | 10.1 |
| on | off | 0x31 | 633 | 26.1 | 29.3 |
| off | on | 0x32 | 2080 | 85.3 | 98.5 |
| on | on | 0x33 | 2095 | 85.9 | 102.4 |

表 3.4: 4 つのスレッショルドレベル電圧値と換算エネルギー、設定値。

これらのスレッショルドレベル電圧の誤差の原因と考えられる。このままのスレッショルドレベル電圧値でもミッションには影響がないため、現状のままで使用することにした。

次の図 3.5 は A 系 B 系に出力テストパルスを入力することで、プリアンプ、ゲインアンプ、コンパレータのそれぞれの出力についてオシロスコープでみたものである。それぞれの波形はとてもきれいで、雑音は出力に対して相対的に小さく、コンパレータが誤作動していないことからアナログ基板は正常に動作していると考えられる。

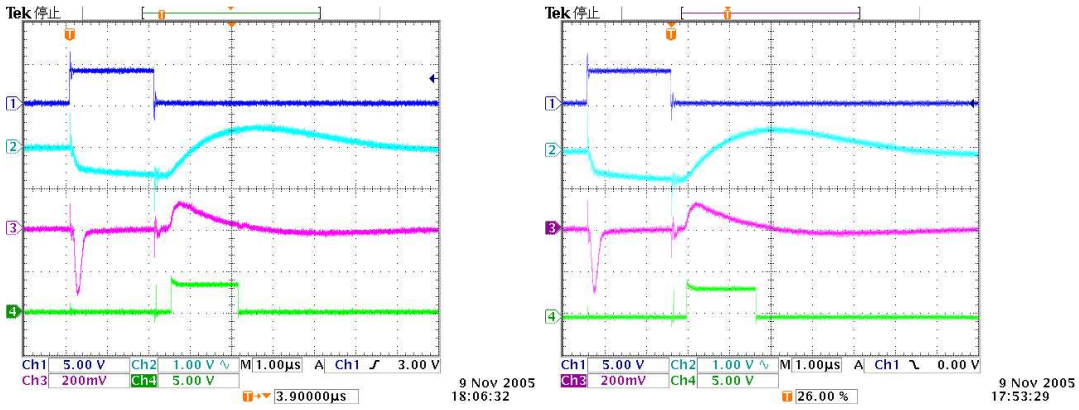


図 3.5: FM の A 系 (左)、B 系 (右) 出力。テストパルス (1)、A225 出力 (2)、Gain アンプ出力 (3)、コンパレータ (4)。

### 3.3.4 マザー基板単体試験

マザー基板には DC-DC コンバータ (以下 HV と略する) が搭載されていて、これは APD に電圧を印加するためのバイアス電源となっている。この HV は、デジタル基板の H8 マイコンから出力される電圧で制御されている。バイアス電源出力値の線型性からのずれ

は、APD に印加するために必要な要求電圧領域 (200 V∼430 V) において、0.3 % 以内に収まらなくてはいけない。これは、APD の内部増幅率を G とすると、

$$\frac{1}{G}\cdot\frac{dG}{dV}\cong 3.4[\%/\mathrm{V}] \tag{3.2}$$

という関係があるため、HV 単体における増幅率 G の変動を 0.3 % 以内に抑えるためには、その出力値の変動を 0.3 $V$ 以内に抑える必要から生じる。

### DC-DC コンバータ出力の線形性

HV の出力線型性を見るために Windows マシンからシリアル回線を用いて、デジタル基板搭載の H8 マイコンに DAC 入力値を指定し、HV からの出力をテスターを用い、直接計測して、バイアス電源の線型性を試験した。このとき、印加電圧制御システム (第 4 章参照) をオフにした状態にした。

結果を図 3.6 に示す。図を見ると、誤差 0.3 % 以内という要求を明らかに満たしていないことが分かる。第 2.3.4 節にて述べたように、HV 単体においての入力電圧に対する出力誤差は 0.3 % 以内を満たしている。よって原因は HV 自体にあるのではなく、DAC 出力から HV 入力までの間に原因があると考えられる。以下、原因について調査を行った。

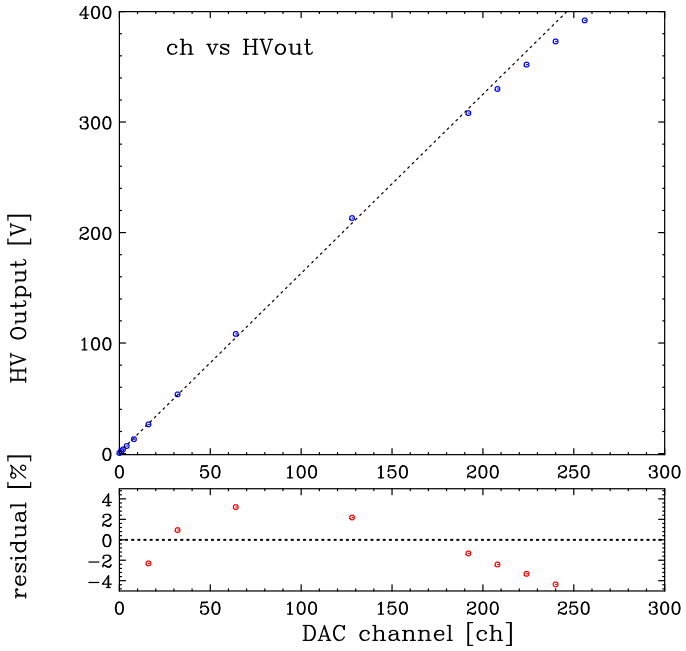


図 3.6: DAC channel[ch] の出力に対する HV 出力 [V]。

### DC-DC コンバータ入力部の改善点

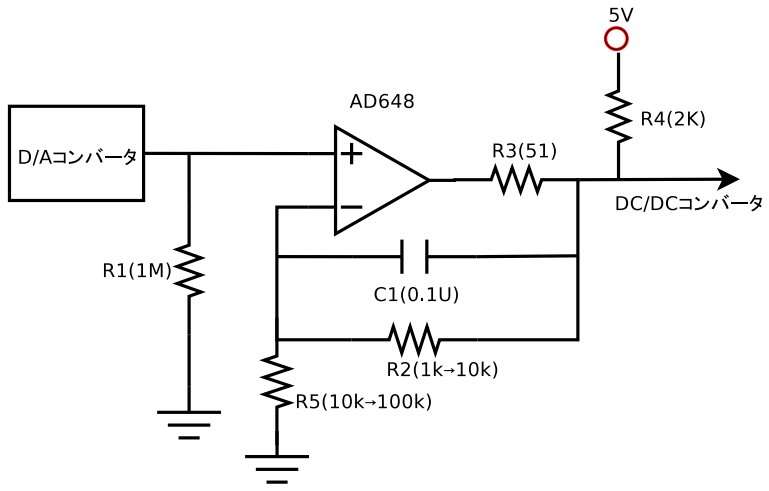


図 3.7: DC-DC コンバータ入力部の回路図。R2、R5 の抵抗値を変更。

図 3.7 は DC-DC コンバータ入力部の回路図である。AD648 の出力のダイナミックレンジは電源の 5 V を R2、R4、R5 によって抵抗分割することによって決められる。変更前のダイナミックレンジは、式 (3.3) によって与えられ、

$$V_{\mathrm{out}} \leq \frac{\mathrm{R2}+\mathrm{R5}}{\mathrm{R2}+\mathrm{R4}+\mathrm{R5}} \times 5[\mathrm{V}] = 4.2[\mathrm{V}] \tag{3.3}$$

これより、AD648 のダイナミックレンジは 4.2 V となっていた。そこで、抵抗分割の方法を変更して AD648 出力のダイナミックレンジに幅を持たせることにした。抵抗の変更は図 3.7 の抵抗名について、$\mathrm{R}_2$ を 1 kΩ から 10 kΩ、$\mathrm{R}_5$ を 10 kΩ から 100 kΩ に変更した。変更後のダイナミックレンジの値は 4.91 V となった。

### DC-DC コンバータ入力部の抵抗値の変更後

この抵抗分割の変更後に再度 DC-DC コンバータ出力の線形性の試験を行った。試験セットアップは前回と同様である。結果を図 3.8 に示す。図より前回の線形性からのずれは改善され、かつ要求されている誤差 0.3 % 以内が保たれていることが分かる。

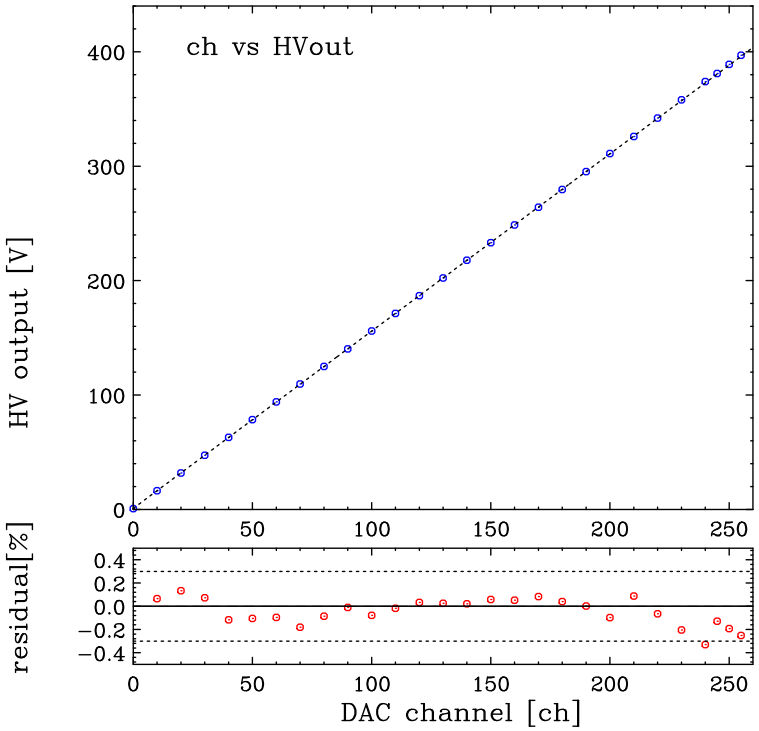


図 3.8: 変更後の DAC channel に対する HV 出力。

# 第4章　APD印加電圧制御試験

## 4.1　目的

APD は温度によってその増幅率が変動してしまう。Cute-1.7 のような超小型衛星では普通の衛星のような温度制御機構を衛星に搭載することができないため、軌道上で温度が変化しても APD の増幅率を一定に保つシステムを作ることが必要不可欠となる。河合研究室では、APD の増幅率を一定に保つ制御システムを確立し [8]、APD モジュールに本システムを組み込むことにした。

APD の増幅率には印加電圧依存性があり、式 (2.1) に示すように、1 V あたり増幅率が 3.4 % 変化する。ミッションを遂行するために許容される APD の増幅率のずれは、要求電圧領域において ∼5 % である。これをバイアス電源からの出力電圧の誤差に換算すると、およそ $\Delta V$= 1.5 V となる。また、温度依存性については式 (2.2) で与えられている。増幅率の温度依存性を打ち消す制御のために、要求される温度分解能は $\Delta T$=1.35 K である (付録 A.4 節参照)。以下、これらの要求を本システムが満たしているか、確認の試験を行う。

## 4.2　APD増幅率補償システムの原理

### 4.2.1　設計概念

温度によらず、APD の増幅率を一定に保つためには、APD の増幅率が温度と印加電圧の両方に依存する性質を用いて、増幅率が一定になるようにリアルタイムに印加電圧を変化させれば良い。図 4.1 に示すように、増幅率を一定に保つための電圧値が常に存在することが分かる。つまり衛星の温度 (APD 素子の温度) を知ることができれば、その値に応じて増幅率を一定に保つための印加電圧の値を APD に与えることができる。

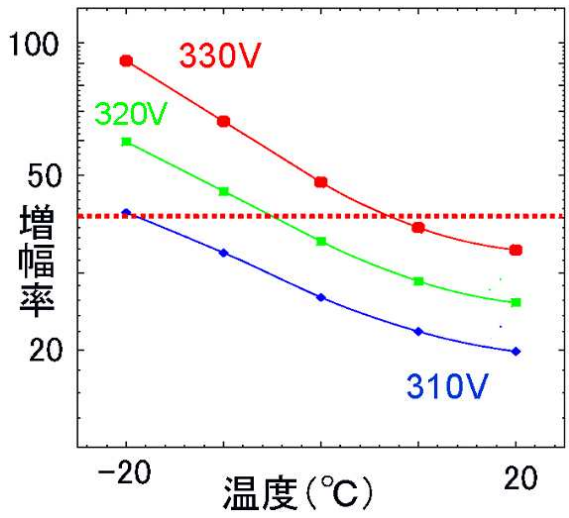


図 4.1: HV の各電圧値における温度に対する APD 増幅率。

### 4.2.2 印加電圧制御システムの構成

APD モジュールにおける印加電圧制御システムの概略図を図 4.2 に示す。このシステムは大まかにアナログ基板に搭載されている温度センサ (AD590) と、デジタル基板に搭載されているマイクロコンピュータ (H8 マイコン)、マザー基板に搭載されているバイアス電源 DC/DC コンバータ (HV) によって構成されている。マイクロコンピュータには、アナログ信号を 8 ビットのデジタル信号に変える A/D コンバータ (ADC) と、その逆の機能を持っている D/A コンバータ (DAC)、そして APD に印加する電圧を作り出す DC/DC コンバータ (HV) が搭載されている。

印加電圧制御システムの大まかな流れは以下の通りである。

1. 温度センサ (AD590) からの出力を H8 マイコンの ADC で 2 秒毎に読み出す
2. 増幅率を一定に保つための HV の電圧値を H8 上に組み込まれたプログラムにより求め、8 ビット信号で返す
3. 8 ビット信号を DAC でアナログの電圧値に変換する
4. D/A コンバータから出力されるアナログ信号を DC/DC コンバータ (HV) で APD の増幅率を一定にする電圧値まで昇圧する

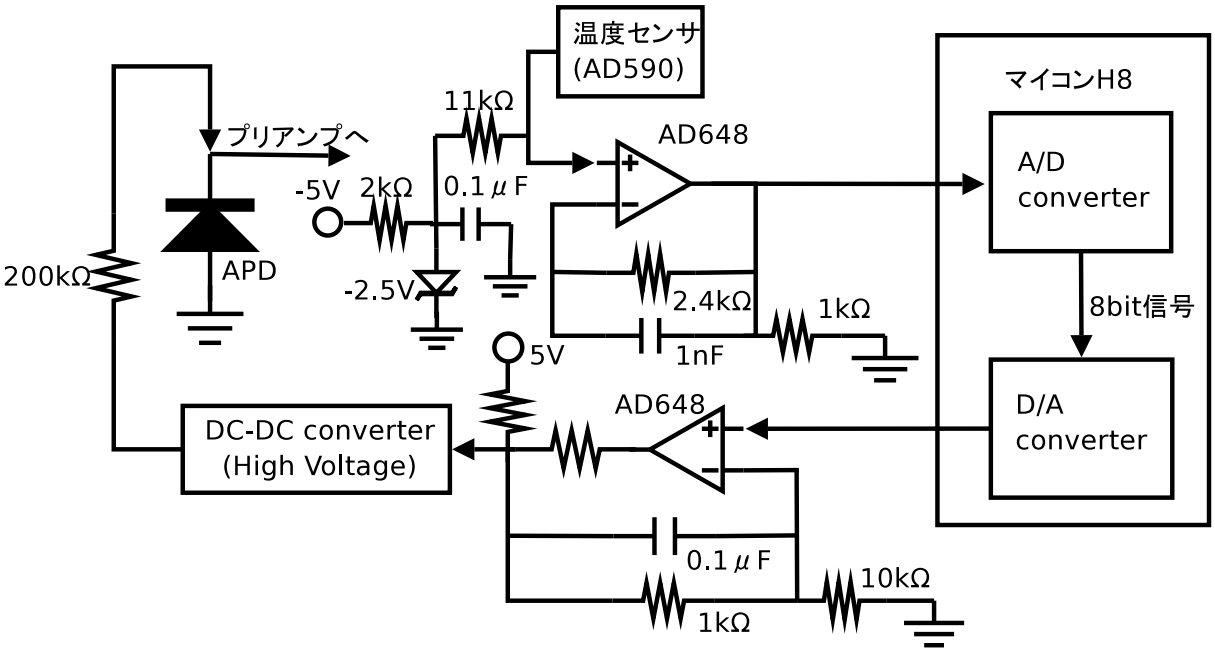


図 4.2: APD モジュールの APD 印加電圧制御システムの概略図。

### 4.2.3 温度センサ (AD590) の性能

AD590 は温度を変数とした 1 次関数として電圧を出力し、途中 AD648 により非反転増幅をされることによって、ADC がその値を感知する仕組みとなっている。しかし、AD590 の出力が変化してもその分解能 $\Delta T$ 以下の変化であれば、ADC はその変化を感知することができず、DAC 出力も同様に変化しないことになるので、結果 APD の増幅率が変化してしまう。それゆえ、AD590 の出力誤差からくる温度変化を読み取るためにも、分解能 $\Delta T$ は要求されている温度分解能よりも小さくなければならない (付録 A.4 節参照)。

恒温槽に APD モジュールを入れ、$-20$ ℃ ~60 ℃と温度を変化されたときの AD590 の出力値を図 4.3 に示す。AD590 からの出力値はテスターを用いて AD648 の出力部を計測した。この図から、AD590 の出力は線形近似を行うことができて、

$$V_{temp}(T) = 0.03753 \times T[\mathrm{deg}] + 1.879[\mathrm{V}] \tag{4.1}$$

となる。ADC のダイナミックレンジは 3.0 V で、ADC は 8 ビットで動作しているので、ADC の分解能は $3.0[\mathrm{V}]/256[\mathrm{ch}] = 11.7[\mathrm{mV}]$ と求まる。図 4.3 に示されている線型性からの各温度における温度センサ出力のずれは、1ch 内の精度には収まらず、2ch 以内に辛うじて収まることになる。2ch のずれ、つまり図に示されている誤差の最大値である 20 mV

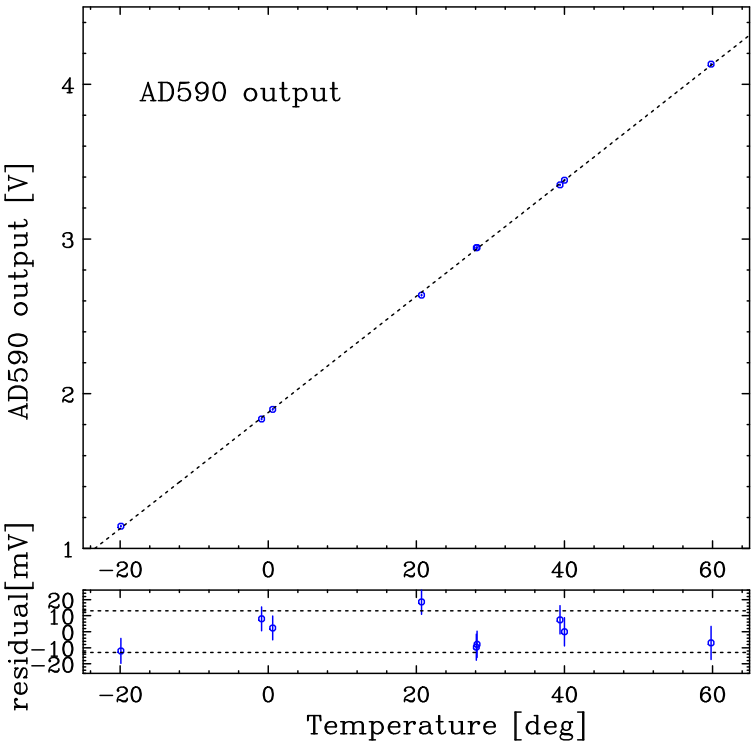


図 4.3: AD590 の温度に対する出力と近似関数 (上) とその残差 (下)。下図には ADC の 1ch に相当する電圧を点線で表示 (12mV)。

のずれを、温度のずれ $\Delta T$ に直すと、式 (4.1) より、

$$\Delta T = \frac{20[\mathrm{mV}]}{0.0375[\mathrm{V/deg}]} = 0.53[\mathrm{deg}] \tag{4.2}$$

と求まる。この値は温度の要求分解能 1.35[℃] より小さいので、AD590 を温度センサとして使用してよい。

### 4.2.4 印加電圧制御システムの原理とその関係式

具体的に印加電圧制御システムの原理について述べることにする。まず、AD590 から出力されるアナログ信号が 1 秒ごとに ADC によって読み出される。このときのアナログ出力を $V_{\mathrm{temp}} = V(T)$ と定義する。このアナログ信号は ADC に読み込まれると 8 ビット、つまり 256 段階のデジタル値に変換される。そして ADC のダイナミックレンジを $V_{\mathrm{ADC}}$、アナログ出力に対応したデジタル出力を $n_{temp}[\mathrm{ch}]$ とすると、$V_{\mathrm{temp}}$ と $n_{\mathrm{temp}}$ の関係を、

$$n_{\mathrm{temp}}[\mathrm{ch}] = \mathrm{int}\left[\frac{V_{\mathrm{temp}}[V]}{V_{\mathrm{ADC}}[V]} \times 255[\mathrm{ch}]\right] \tag{4.3}$$

と表すことができる。ここでAD590のアナログ出力$V_{\rm temp}$と温度$T$の関係は前説で求めたように、式(4.1)の逆を計算し、

$$T[\mathrm{deg}] = 26.64 V_{\rm temp} - 50.06 \tag{4.4}$$

のように1次関数で表すことができる。式(4.3)と式(4.4)より、$n_{\rm temp}$と温度$T$の関係を式(4.5)のように表すことができる。

$$T[\mathrm{deg}] = \left( 26.64 \times \frac{V_{\rm ADC}[\mathrm{V}]}{255[\mathrm{ch}]} \right) \times n_{\rm temp} - 50.06 \tag{4.5}$$

ADCのダイナミックレンジ$V_{\rm ADC}$は3Vと求まっているので、ADCのchannel数と温度の関係式は以下で表される。

$$T[\mathrm{deg}] = 0.3134 \times n_{\rm temp} - 50.06 \tag{4.6}$$

我々は衛星に搭載するAPDに対して、増幅率を一定に保つ温度特性関数を調べた。その温度特性関数は線形近似で表すことができなかったので、3次の多項式近似を採用することにした。式(4.8)には、APDの増幅率が30、50のときの温度特性関数を示す。

$$V_{30}(T) = 0.0000630 T^3 + 0.004163 T^2 + 0.79314 T + 358.220 (\mathrm{gain} = 30) \tag{4.7}$$

$$V_{50}(T) = 0.000106 T^3 + 0.002211 T^2 + 0.81396 T + 381.346 (\mathrm{gain} = 50) \tag{4.8}$$

ADCの最高出力値に対応するHV出力を$V_{\rm max}$とすると、DACの出力は式(4.8)を用いて、

$$n_{\rm out} = \mathrm{int} \left[ \frac{V_{50}[V]}{V_{\rm max}[\mathrm{V}]} \times 255[\mathrm{ch}] \right] \tag{4.9}$$

となる。式(4.8)と式(4.9)の関係式はDC-DCコンバータの特性を無視した形となっている。これは、DC-DCコンバータの出力が線型性に優れているためである(第2.3.4節参照)。

**DC-DCコンバータへの要求**

次に、マイコンH8に実装する3次の温度特性関数によって制御されたDC-DCコンバータの出力がどの程度の誤差に収まっていれば良いか評価する。

マイコンH8に実装されているプログラムには、その計算過程で式(4.8)、式(4.9)で表されているように、3次の温度特性関数で得られた値(実数)からDACのchannel数(整数)を決めるという離散的な計算を行っている。ゆえに、必然的にDAC1ch分の誤差が生じてしまう。$V_{\rm max}$は396.5 [V]と得られているので、DACの分解能に相当する電圧差は、

$$\Delta V = 396.5[\mathrm{V}]/256[\mathrm{ch}] = 1.549[\mathrm{V/ch}] \tag{4.10}$$

となる。これからHVの出力誤差は$\Delta V$=1.5[V]に収まっていれば良いことにする。第4.1節にもあるように、許容精度範囲は1.5 Vとなっているので、以下本システムを用いてAPDの増幅率の変動を抑えるために許容されるDC-DCコンバータ出力の誤差レベルを、$\Delta V$=1.5[V]と設定して試験の評価を行うことにする。

## 4.3 疑似温度センサ入力におけるDC-DCコンバータ出力の確認

温度試験を行う前に、直流安定化電源を用いて外部から電圧を供給することにより、温度センサ出力を模擬し、試験を行った。用いたAPDモジュールは衛星搭載品ではなく、それと同等な性能をもつ調整用予備基板である。試験セットアップは図4.2の温度センサ(AD590)を直流安定化電源に置き換えたもので、DC-DCコンバータからの出力はデジタル電圧計を用いて直接読み取った。また、ADC、DACのデジタルデータの取得はデジタル基板に搭載されているマイコンH8からWindowsマシンを用いてシリアル回線を介して行った。APDの増幅率は30と一定にして試験を行った。

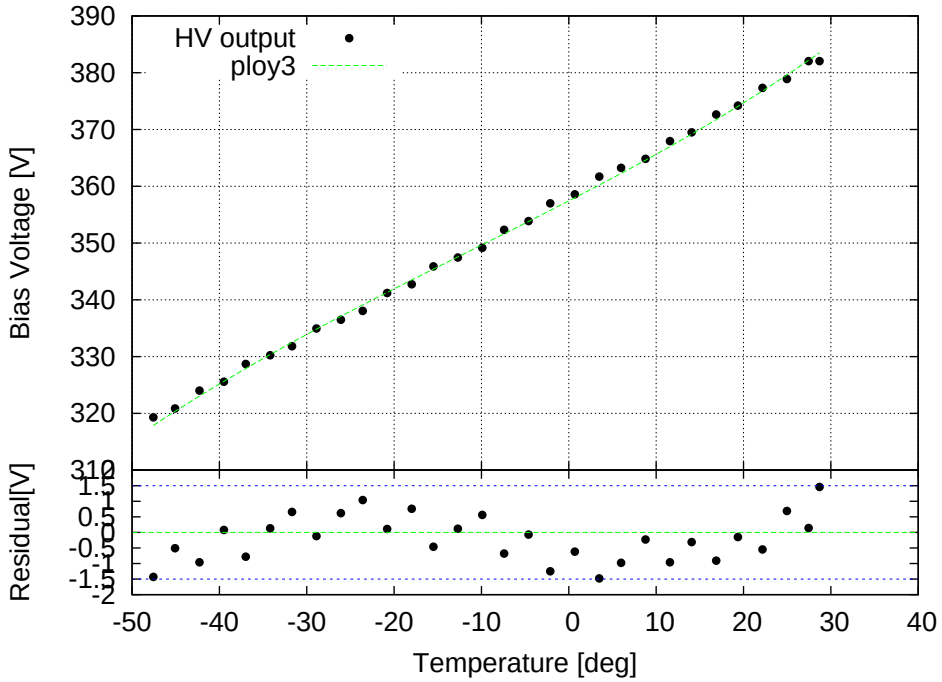


図 4.4: 直流安定化電源による疑似温度センサ入力に対するDC-DCコンバータ出力とその残差。下図には誤差1.5Vラインを表示した。

直流安定化電源による疑似温度センサ入力に対する DC-DC コンバータ出力と補正関数に対する残差を図 4.4 に示す。図 4.4 より、得られたデータは −40∼27 ℃の範囲において表記されている誤差内にほぼ収まっていることが読み取れる。27 ℃以上のデータがないのは、直流安定化電源からの出力が ADC のダイナミックレンジを越えてしまったためである。以上から、印加電圧制御システムは疑似的に温度を変化させた場合において正常に動作していることが分かる。

## 4.4 温度サイクル試験

### 4.4.1 目的

この試験では、衛星軌道上における温度変化を想定して、温度変化率 15deg/h という急な温度変化の中で APD の増幅率を一定に保つべく、温度センサの出力から HV の出力を変化させることができるのかを試験する。

### 4.4.2 試験方法

本試験は初めて温度試験を行うため、不慮の事故が起きる可能性も考えて、調整用予備基板を用いて行うことにした。温度サイクルは 40 ℃から −20 ℃までを 4 時間で変化させ、−20 ℃から再度 40 ℃に戻すという合計 8 時間を 1 サイクルとし、温度サイクルを 3 回行った (図 A.4)。本試験では APD の増幅率を 30 と設定した。

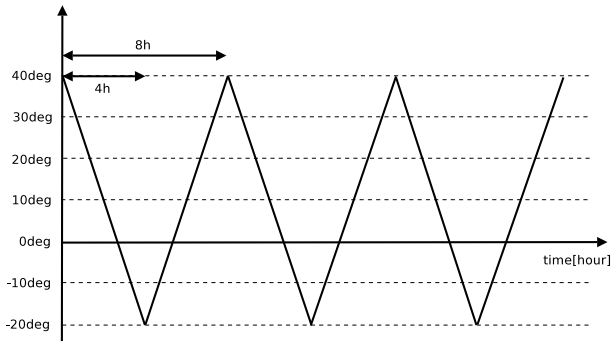


図 4.5: 温度サイクル試験における恒温槽の温度変化の様子。

### 4.4.3　試験結果と考察

図 4.6 には時間に対する恒温槽の温度変化と AD590 の出力から算出された温度変化のグラフと、HV からの出力のグラフを示す。恒温槽の温度と AD590 の温度には、多少の遅れが見られる。これは温度センサが基板やセンサ自身の持つ熱容量があるため、少し遅れて温度変化しているためである。また、30 ℃で AD590 の出力が飽和してしまっていて、HV の出力が頭打ちとなってしまっている。これは上で述べた通り、AD590 の出力がマイコン H8 のダイナミックレンジを越えてしまったためであり、そのために HV の出力が 385 V 程度までしか出力できなかったと考えられる。以上、調整用フライトモデル基板において印加電圧自動制御システムは正常に機能していることが証明された。

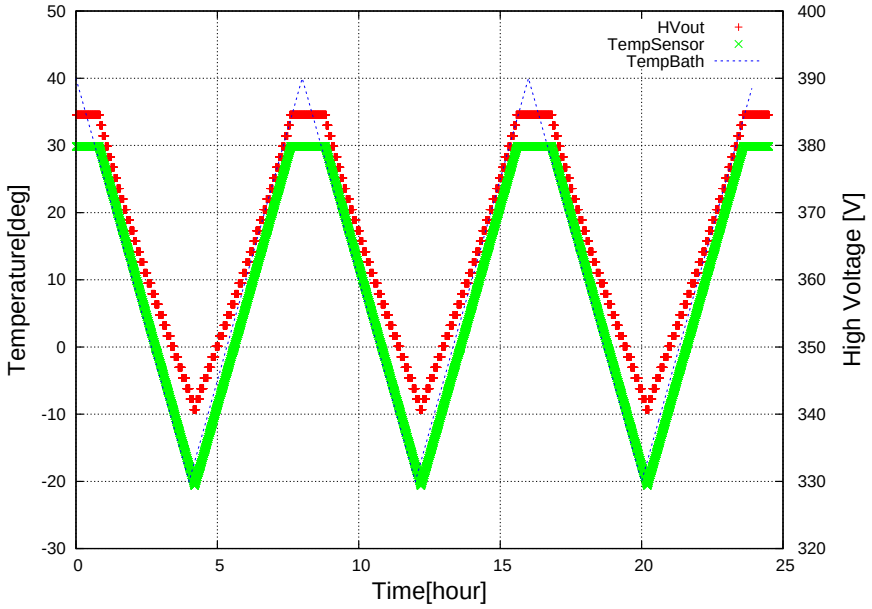


図 4.6: 左縦軸：時間 [hour] に対する恒温槽の温度変化 (破線) と AD590 の認識した温度 (×)[deg]。右縦軸：時間 [hour] に対する HV 出力 (＋)[V]

## 4.5　フライトモデル基板による印加電圧制御試験

調整用フライトモデル基板において印加電圧制御システムは正常に機能していることが確かめられた。フライトモデル基板についても上記システムが正常に機能するか試験を行った。本試験は AD590 の温度に関する出力動作は正常であると仮定し、直流安定化電源を用いて疑似温度センサを入力し、HV からの出力は Digital-Volt meter を用いて読み取った。

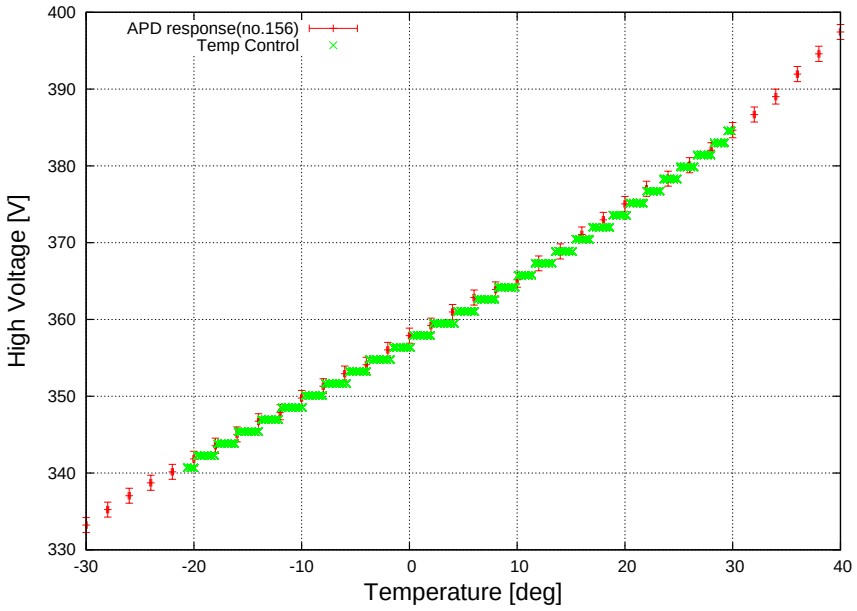


図 4.7: 温度 [deg] に対する HV 出力 [V]。

### 4.5.1 結果と考察

直流安定化電源からの疑似温度センサ入力に対する HV の出力を温度特性関数とともに図 4.8 に示す。温度に対する residual(残差) のグラフに、DAC の 1 ch に相当する 1.5 V のラインを表示した。APD の増幅率が 30 のとき、HV からの出力は全て DAC1 ch 以内に収まっていることが分かる。つまり、H8 マイコン内での A/D 変換、D/A 変換時における出力のずれは最小限に抑えられていることが分かる。増幅率を 50 とした時も、増幅率 30 時と同様に、出力のずれが 1.5 V 以内に収まっている。以上、フライトモデル基板に対して印加電圧制御システムは DC-DC コンバータ出力が要求精度誤差内に収まっていることから、本システムは衛星軌道上において正常に機能することが確かめられた。

## 4.6 結論

Cute-1.7 に搭載された APD 印加電圧制御システムは、急激な温度変化を経ても正常に機能することが確かめられた。

また、本衛星に搭載しても DC-DC コンバータから出力される高電圧を調整することによって、APD の増幅率の誤差が要求されている 5% 以内に抑えられることが確かめられた。

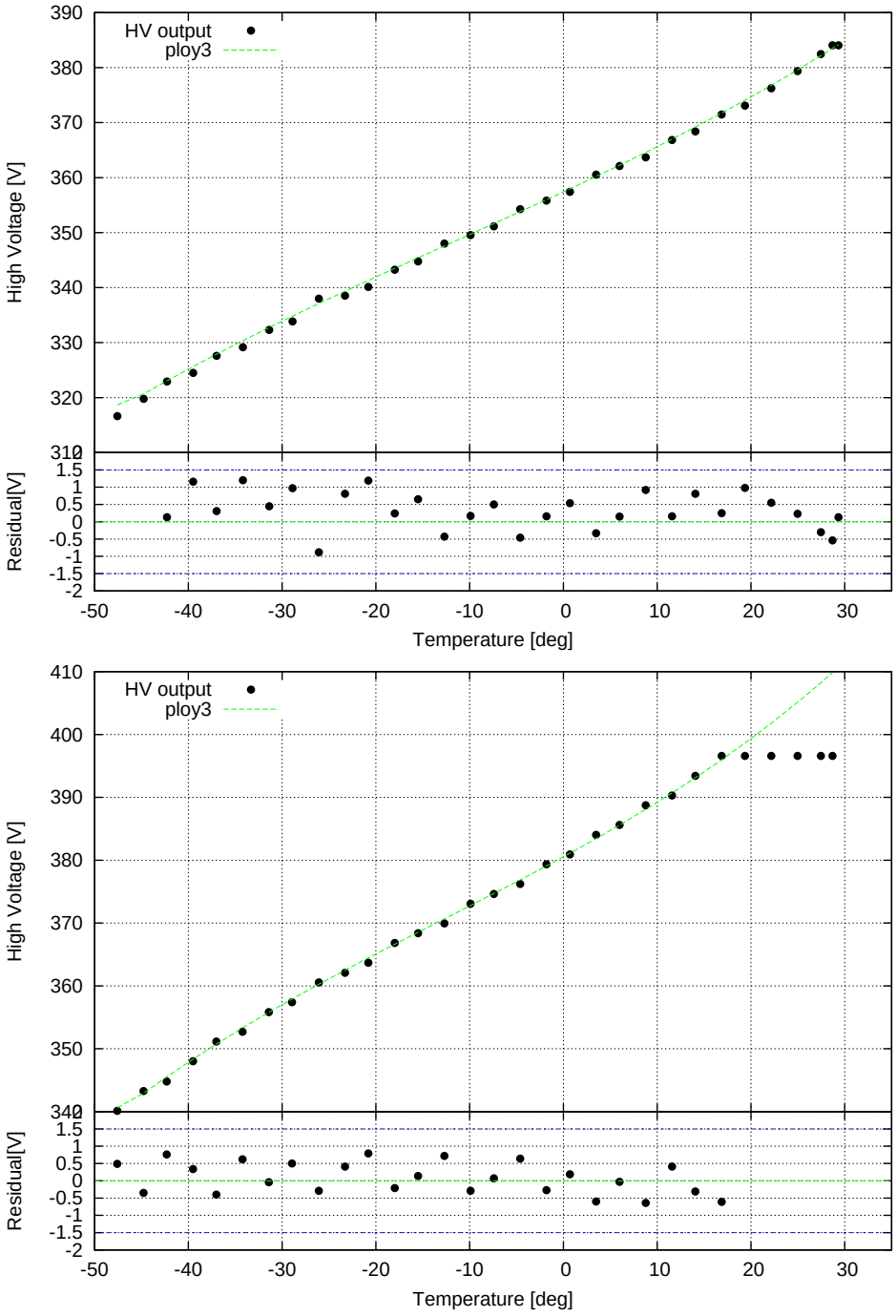


図 4.8: 疑似温度センサ入力における HV 出力と、温度特性関数と HV 出力の残差。図には 1.5 V ラインを表示した。上：G=30、下：G=50。

# 第5章　フライトモデル基板高計数試験

## 5.1　目的

APD モジュールは、異常放射線帯中の 30 keV 以下の低エネルギー荷電粒子を世界で初めて測定する。このエネルギー領域では 1 cts/s という低レートから $10^6$ cts/s という高レートで計数を行う必要があると予想されるので、フライトモデルの計数測定能力を試験した。

### 5.1.1　APD モジュールで期待される計数率

Cute-1.7 のアナログ回路を完全な拡張型とすると、不感時間を $\tau$=110 nsec としたとき、検出器に入射したイベント数と H8 カウンタの数えたイベント数の関係は式 (A.6) で表され、これをプロットしたものが図 5.1 である。図より拡張型の場合、$10^6$ cts/s の入力レートに対してカウントレートはおよそ $0.9 \times 10^6$ cts/s となっており、現状の APD モジュールは $10^6$ cts/s の入力レートに対して 90 % の計数測定能力を持っていると予想される。

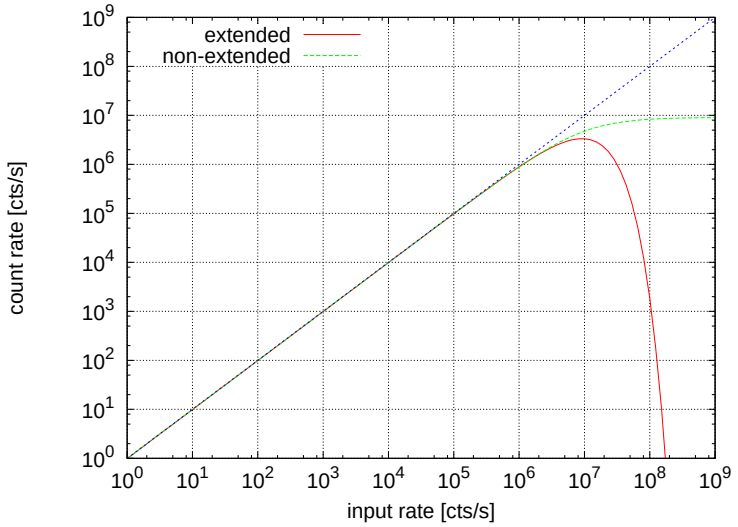


図 5.1: 不感時間を $\tau$=110 nsec と設定したときの入力レートに対する出力レート。拡張型 (曲線)、非拡張型 (破線)。

## 5.2　試験方法とセットアップ

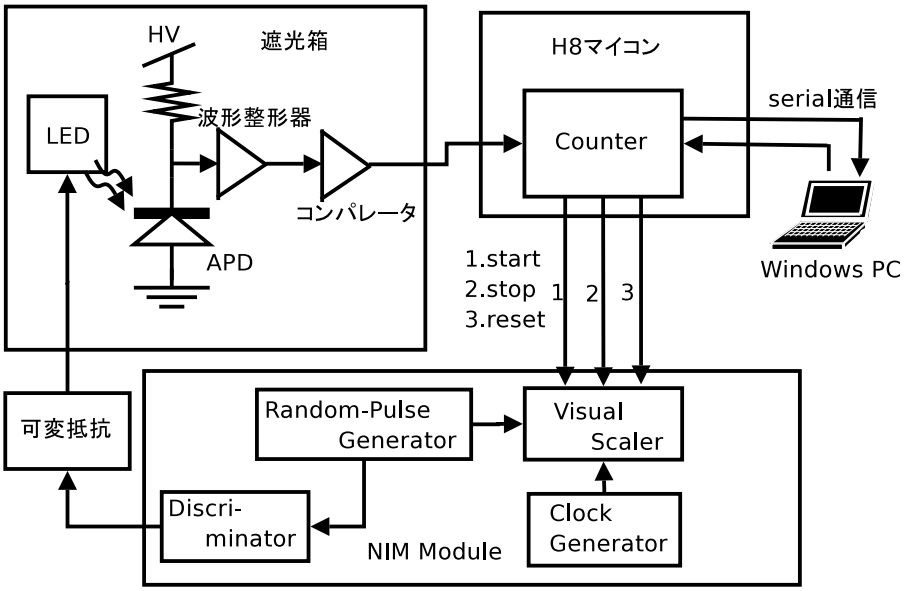


図 5.2: 高計数試験での回路のセットアップ。

理学ミッションは荷電粒子を捉えることが目的であるが、本試験では青色 LED 光を入射粒子の代わりとし、NIM モジュールのランダムパルサから疑似的に $10^3$ cts/s から $10^6$ cts/sec までのレートを作り出すことにした。実験セットアップを図 5.2 に示す。

LED をランダム発光させて疑似的信号とする。LED の発光量、発光時間の調整のため、ランダムパルサのトリガー出力を一旦ディスクリネーターに入れ、その後可変抵抗器を用いてパルス波高を調節できるようにした。こうすることによって、異なるエネルギーを持った荷電粒子が APD に入射してきたときの計数率を模擬することができる。このディスクリネーターにランダムパルスが入るとそれと同期して LED が動作するようになっている。あらかじめ高電圧を印加しておいた APD に LED 光を照射することによってアナログ信号を作り出し、プリアンプ、ゲインアンプによりアナログ処理を行う。このとき、図 5.2 にあるように、LED 以外の光が APD に入らないようにするため、塩化ビニルでできたブラックシートを用いて完全な遮光を行った。その後、コンパレータを通してデジタル基板に搭載されているマイコン H8 のカウンタで計数を測定する。そしてそのデータをシリアル通信を用い、Windows マシンで取得することにした。計数率は NIM モジュールの Visual scaler を用いて、ランダムパルサの出力計数と、Clock generator からの計数を数えた。また、この Visual scaler のスタート、ストップ、そして計数のリセット全てのコマンドをデジタル基板の H8 マイコンを介し、Windows マシンで行えるようにした。

試験方法は、まずランダムパルサのパルス幅を 200 nsec と一定にし、可変抵抗を用いてゲインアンプのパルス波高をオシロスコープでモニターすることによって調節した。パルス波高は 600 mV から 2600 mV まで変化させた。APD には増幅率を 50 と一定にするために第 4 章で述べた印加電圧制御システムを用いて電圧を印加した。

## 5.3 結果

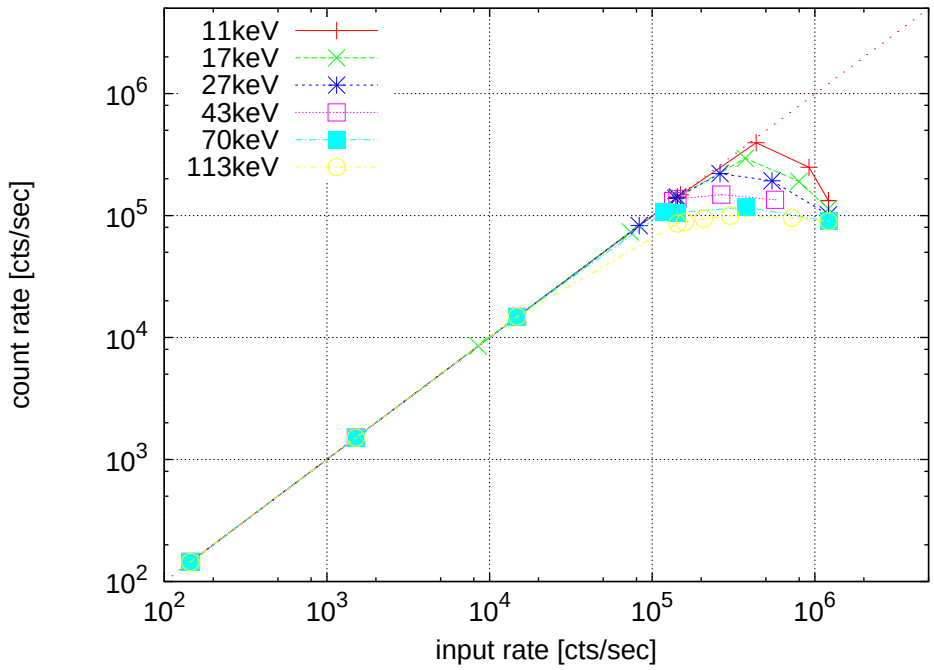


図 5.3: ランダムパルサ入力に対する H8 カウンタの計数。

結果を図 5.3 に示す。図中のエネルギー表示は APD の増幅率を 50 としたときに、$^{55}$Fe を用いて $\gamma$ 線を照射したときにゲインアンプより出力されるパルス波高が 144 mV であったので、これを基準にとり換算した。図を見ると、$10^6$ cts/s の入力レートに対してそれぞれ 80~95 % 程度の数え落しが見られる。これは、入力レート $10^6$ cts/s 時において予想されている数え落し率 (10 %) に比べはるかに大きい。図の特徴としては、見掛けの入射エネルギーが大きいほど低計数率域での数え落しが多くなっていて、高計数率域になるに従い、山なりに数え落しが少なくなっている。さらに特筆すべきは、ランダムパルサの計数率が $10^6$ cts/s のあたりで、エネルギーに違いはあるが、H8 カウンタが計測できた計数率が $10^5$ cts/s とほぼ一定になっているということである。APD モジュールのアナログ回路は拡張型であるが、図 5.3 を見る限りでは、高エネルギーの可視光を APD に入射した

ときは非拡張型の特徴を持っている様に見える。つまり、高エネルギーの可視光が入射されると、入射レートの増大とともにアナログ回路に何らかの誤動作が生じていると考えられる。

## 5.4 原因究明

### 5.4.1 試験内容

このように数え落しが起きてしまう原因を究明し、改善するために、以下に挙げる試験を行った。

1. 安定化電源を使用する
2. クランプダイオードを外す
3. LED の点滅信号幅を短くする

これらの試験は、ハードウェアの改修が必要とされるので、調整用としてフライトモデル基板と同等なアナログ基板を新たに作成し、以前の試験結果を再現することを確認した後試験を行った。以下、それぞれの試験結果の報告と考察を行っていくことにする。

### 5.4.2 安定化電源の使用

電源基板から安定した電圧と負荷電流をドライブできていなかった可能性が考えられる。それを検証するために本試験を行った。

APD モジュールでは、本来衛星バスから供給されるメインバス電源と 3.3 V バス電源を電源基板で変換して、+5、-5、12 V 電源を各基板に供給しているが、ここではこの各電圧値を安定化電源を用いて供給した。試験結果を図 5.4 に示す。試験結果はゲインアンプ出力値が 600 mV、2600 mV のときである。図 5.4 において、電源基板を用いているときと有意な改善は見られなかった。つまり、電源基板からの駆動力は問題ではない。

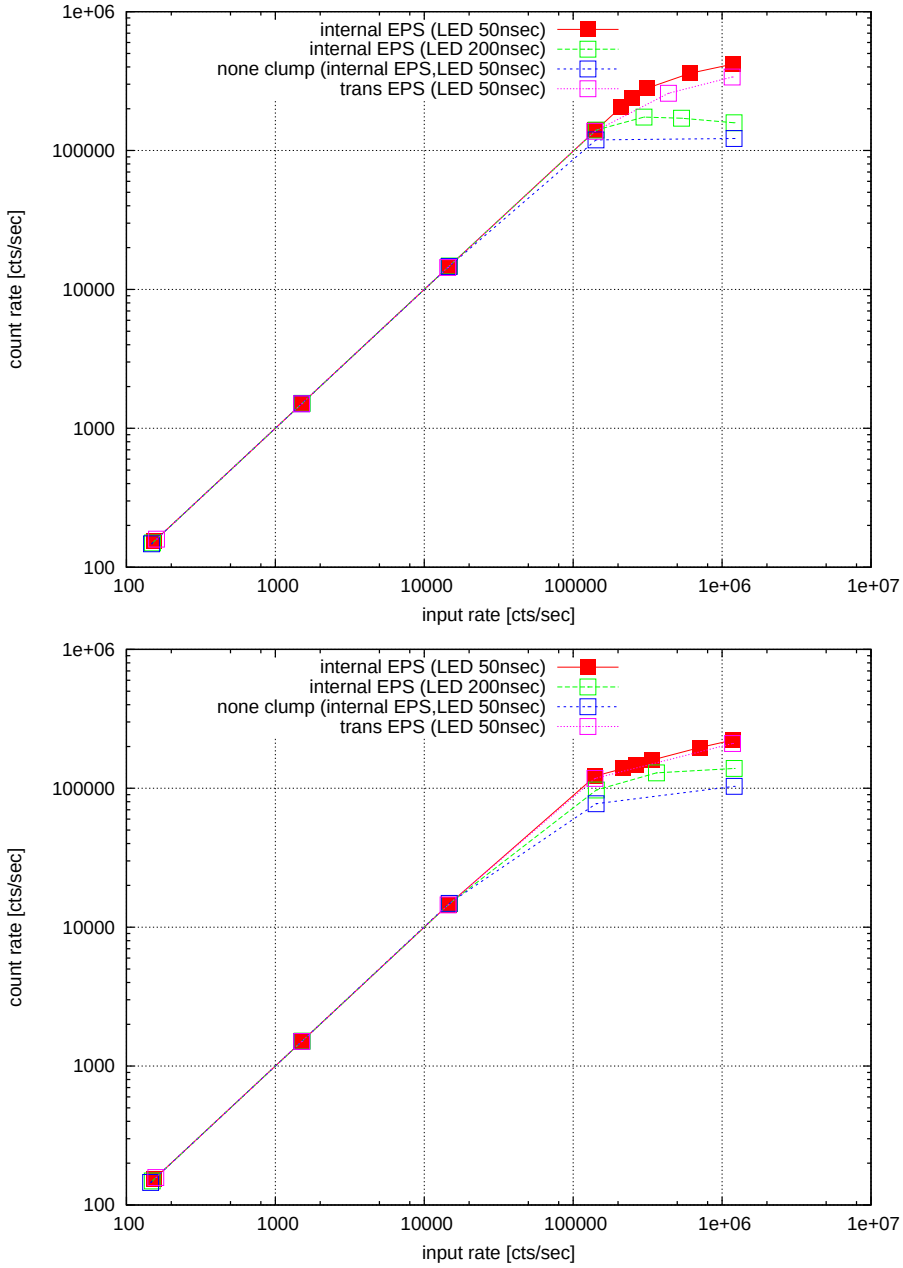


図 5.4: 入力レートに対するカウントレート (上：ゲインアンプ出力 600 mV、下：ゲインアンプ出力 2600 mV)。internal EPS(曲線)：電源基板からの電源供給で LED 幅 50 nsec、internal EPS(破線 (大))：電源基板からの電源供給で LED 幅 200 nsec、none clump(破線 (小))：クランプなし、trans EPS(点線)：安定化電源での電圧供給。

### 5.4.3　クランプダイオードの取り外し

前置増幅器 (以下プリアンプ) の前部にはクランプダイオードが挿入されていて、それぞれを+5、−5 V の DC 電源に接続してある (図 2.3)。こうすることによって、プリアンプに大電流が流れないようにしてきた。しかしこの操作により、APD が検出した荷電粒子により生成された電荷全てががプリアンプに流れ込まず、クランプダイオードを通して各電源系に流れ出し、電荷洩れがおきている可能性がある。その検証のため、クランプダイオードを取り外す試験を行った。外したクランプダイオードは+5 V 側のものである。

試験結果は安定化電源を用いた時と同様に、ゲインアンプ出力が 600 mV、2600 mV のときで行った。結果は前節の図 5.4 に示した通り、状態はまったく改善されなかった。以下の図 5.5 は、そのときのオシロスコープの波形である。一般的に、APD から生成されるパルス幅は、プリアンプ出力の立上り幅とほぼ等しいが、ディスクリミネーターを用いて幅 50 nsec と短くしたランダムパルサ出力に対するプリアンプ出力の立上りが約 400 nsec と、クランプダイオードがある場合に比べて 200 nsec 程度遅くなっていることが理解でき、応答速度が低下していることが分かる。以上のことから、クランプダイオードを取り外すことは数え落しの改善に対して有効な方法ではない。

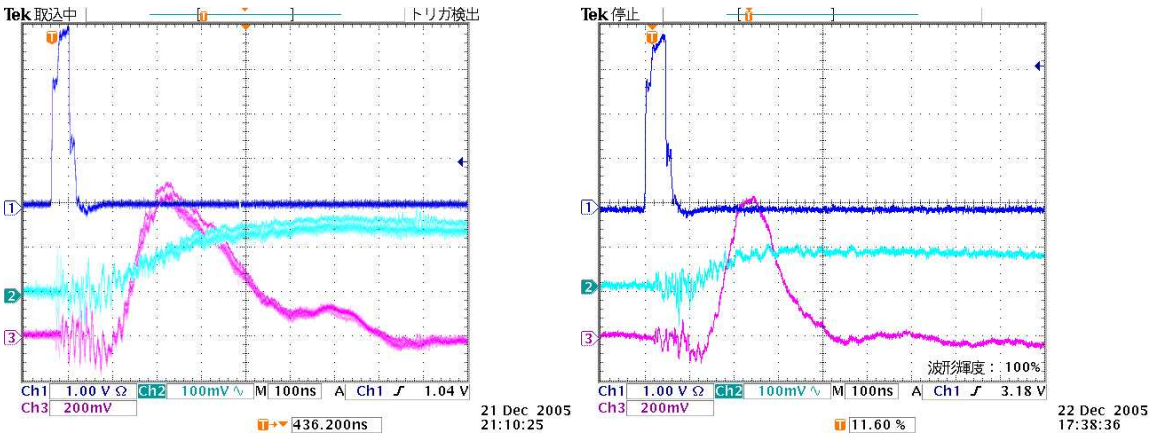


図 5.5: 左：クランプダイオードなし、右：クランプダイオードあり。ランダムパルサ出力 (1)、プリアンプ出力 (2)、ゲインアンプ出力 (3)

### 5.4.4　LED の点灯信号幅を短くする

前回の試験では LED の信号幅を 200 nsec と一定にしていたが、これを NIM モジュールの Desciriminator を用いて 10 nsec まで短くした。LED はアルミ箔で覆うことによって

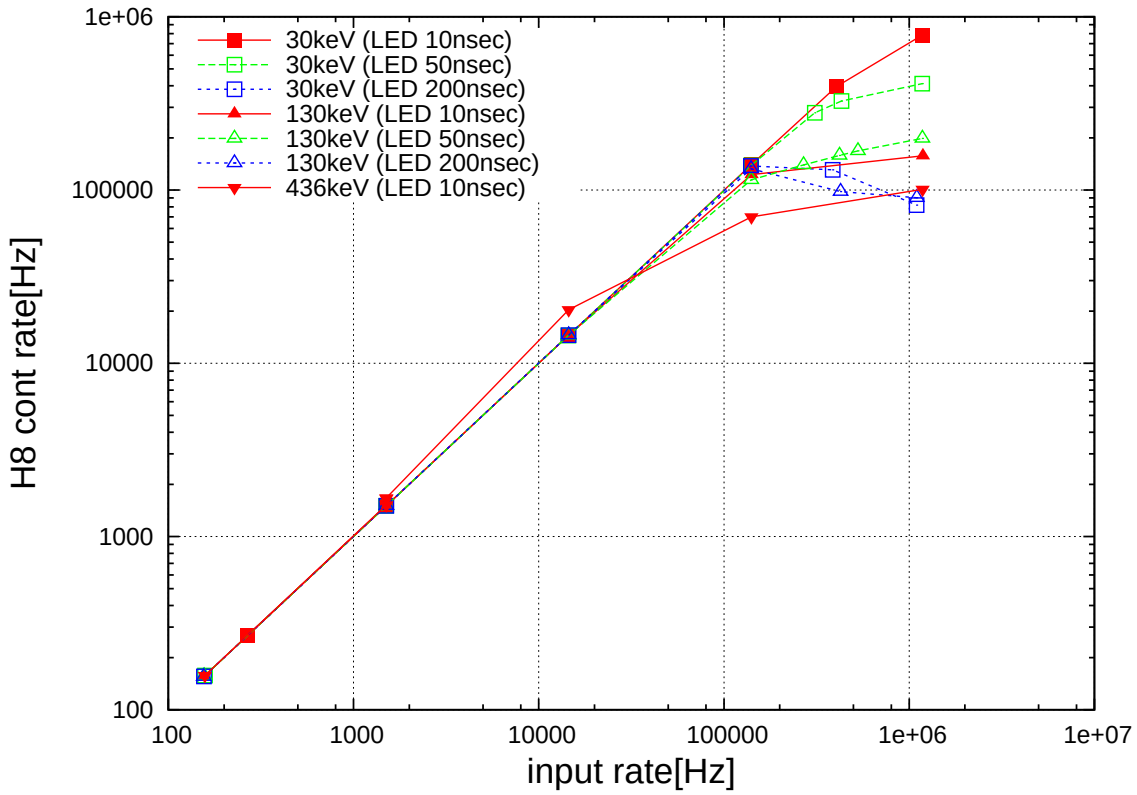


図 5.6: 入力レートに対するカウントレート。赤：LED 信号幅 10 nsec、緑：LED 信号幅 50 nesc、青：LED 信号幅 200 nsec

指向性を高めて計数試験を行った。試験結果を図 5.6 に示す。図 5.6 を見ると、LED 光のパルス幅を短くしていくことにより、$10^5$ cts/s 以上での数え落しが減っていくことが分かる。30 keV 相当の光を APD に入射した場合、$10^6$ cts/s の入射レートに対してカウントレートは 70 % の計数率を示し、以前の結果と比べると有意に数え落しが改善されている。この計数率は、第 5.1.1 節で計算した計数率の限界値にかなり近い。また、入射される光のエネルギーが小さい方が、より数え落しが少ないことが図から読み取れる。

では、何故 LED の入力信号幅を短く、そしてエネルギーを小さくした方が数え落しが少なくなるのか。一般に入力されるエネルギーによって計数されるレートが減ってしまう原因として、プリアンプの飽和が考えられる。次節においてこの問題について議論を進める。

## 5.5 考察

### 5.5.1 プリアンプの飽和について

プリアンプが飽和している様子を詳しく調べた。以下の図 5.7 左は、プリアンプの挙動がレートを上げていくごとに変化し始めているところを捉えたものである。

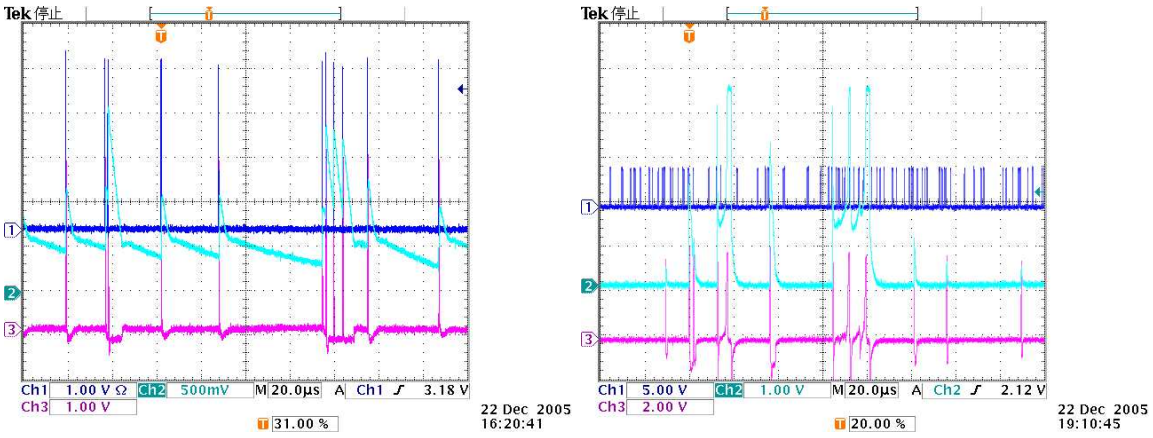


図 5.7: ランダムパルサ出力 (1)、プリアンプ出力 (2)、ゲインアンプ出力 (3)。パルス波高の高さを変えずに、レートだけを増加させた。

この図 5.7 左を見てみると、プリアンプ本来の時定数 (2.8 $\mu$sec) とそれよりも長い時定数 (数 10 $\mu$sec) の 2 成分が存在しているようにみえる。また、この 2 成分の変わり目は、およそオフセット電圧値あたりであることが読み取れる。このレート時において、入力レートに対するカウントレートの数え落しはさほど起きていないが、この第 2 成分が電圧値 0 に完全に落ち込んでる状態では、LED が点滅してもプリアンプが応答しない、つまり窒息してしまっているということが分かる (図 5.7 右)。ゲインアンプについては、オシロスコープで波形を見る限りプリアンプ出力とゲインアンプの出力が 1 対 1 対応、つまりプリアンプが動けばゲインアンプも動くといったようになっているので、ゲインアンプが数え落しの原因にはなっていないと考えられる。第 5.1.1 節で述べたランダムパルサの高レート成分が数え落しを引き起こしている場合についてだが、パルス間の幅が長くても短くてもプリアンプの飽和は起きているので、図 5.7 を見る限りではその様な現象は起きていないと考えられる。

このようにプリアンプが飽和を起こしてしまった背景としては、プリアンプとゲインアンプの時定数が深く関わっていると考えられる。ランダムパルスのパルス幅 10、50、200 nsec のときのプリアンプ出力、ゲインアンプ出力を図 5.8 に示す。

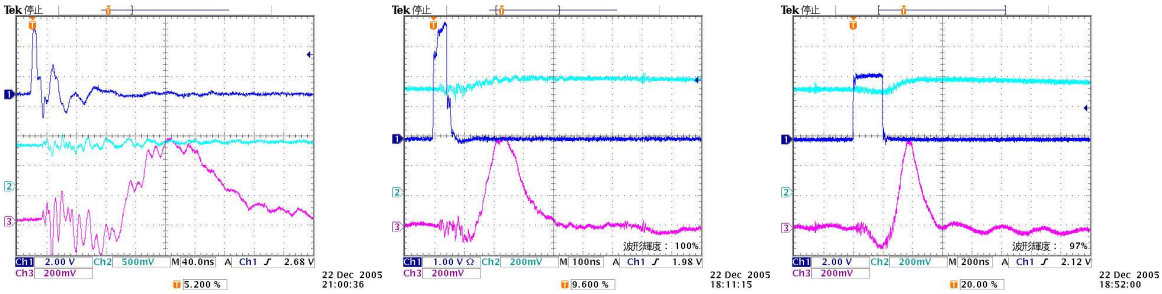

図 5.8: 左 : 10 nsec、中央 : 50 nsec、右 : 200 nsec。ランダムパルサ出力 (1)、プリアンプ出力 (2)、ゲインアンプ出力 (3)

3 つの図を比較すると、ランダムパルサのパルス幅 50 nsec、200 nsec の場合、LED の時間幅とプリアンプの立ち上り幅はそれぞれに対してほぼ等しくなっている。ただ、10 nsec の場合はアルミ箔で LED の光を絞っているので、多少反射の成分が入ってしまっているため、立上りが等しくなっていないと考えられる。

さて、同じエネルギーでもランダムパルサのパルス幅、つまり LED 光の幅によってプリアンプの応答が変わってきてしまう原因について考えてみる。まず、LED のパルス幅が 10 nsec の場合、光が入射して生成された APD の電流は 10 nsec で立ち上がり、2.8 $\mu$sec で立ち下がる波形になる。このとき、ゲインアンプの時定数は数 10 nsec となるので、プリアンプの出力のほとんどを信号として取り出すことができる。つまり、数 10 nsec 程度で入射してきたエネルギーをそのまま得ることができるということになる。次に LED のパルス幅が 200 nsec と長い場合、プリアンプの立ち上がりは 200 nsec とゲインアンプの時定数よりも長くなってしまっている。このためゲインアンプは、プリアンプ出力の一部分しか信号として取り出すことができない。つまり、30 keV 相当のエネルギーである読み出された LED 光は、実はより大きいエネルギーを持っていて、アンプで微分されるときに減らされてしまい、30 keV 相当に見えているだけであるということになる。

以上のことから、プリアンプの立ち上がりがゲインアンプの時定数よりも早ければ、入射してきた荷電粒子のエネルギーをそのままパルスとして取り出すことができると考えられる。さらに、200 nsec のときの方が電荷の量が 10 nsec の場合よりも大きいため、プリアンプが早くに飽和を起こしてしまうことが理解できる。

### 5.5.2　高エネルギー粒子入射時の応答

以上の考察が正しいとすると、幅 200 nsec のときと同等なエネルギーのランダムパルスを LED 光の点滅時間幅を 10 nsec にして APD に入射した場合には、パルス幅が 200 nsec の時と同様に $10^5$ cts/s 程度でプリアンプが飽和し始めると考えられる。以下では、高エネルギー粒子が APD に入射した状態を模擬した試験を行った。

#### 結果と考察

ランダムパルス幅を 10 nsec とし、エネルギー 436 keV 相当の LED 光を APD に入射したときの計数試験の結果を図 5.6 に示す。

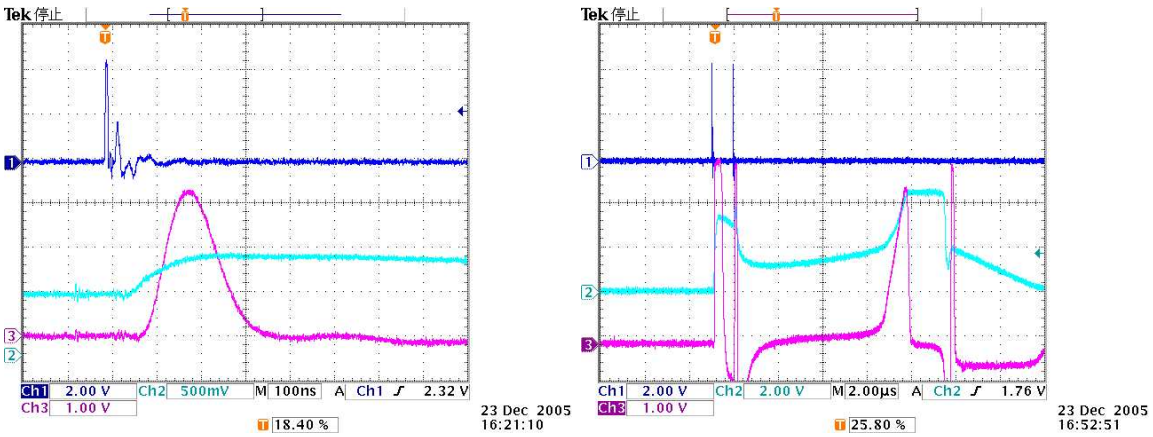


図 5.9: 左 : 正常時、右 : 飽和時。ランダムパルサ出力 (1)、プリアンプ出力 (2)、ゲインアンプ出力 (3)。テストパルスが入力されていないところでプリアンプが誤動作してしまっている。

前回の試験において、入射レートが $10^5$ cts/s 未満で飽和を起こしてしまっていた。図 5.6 を見てみると、$10^4$ cts/s 付近で数え過ぎていることが分かる。図 5.9 は、そのときのオシロスコープで撮った図であるが、プリアンプが飽和を起こしてしまっている様子が見て取れる。つまり、プリアンプの飽和によって、仮想接地の条件が崩れ、プリアンプに疑似パルスが生じてしまったことによって、結果数え過ぎていることが理解できる。さらに高レートになってくると、この数え過ぎよりもプリアンプの飽和からくる数え落しの割合が大きくなってしまっていると考えられ、その結果、数え落しが生じていると推測できる。以上より、プリアンプが窒息する程度の高レート、もしくは大エネルギー入射時には、カ

ウントできる計数率はプリアンプが窒息してから回復するまでに必要とされる時間によって制限されてしまい、結局 $10^5$ cts/s 程度で数え落しが起きると考えられる。

## 5.6 衛星軌道上における荷電粒子計数率の見積もり

衛星軌道上で予測される計数率の見積もりを行い、実際に APD モジュールは荷電粒子探査を行えるか評価を行う。

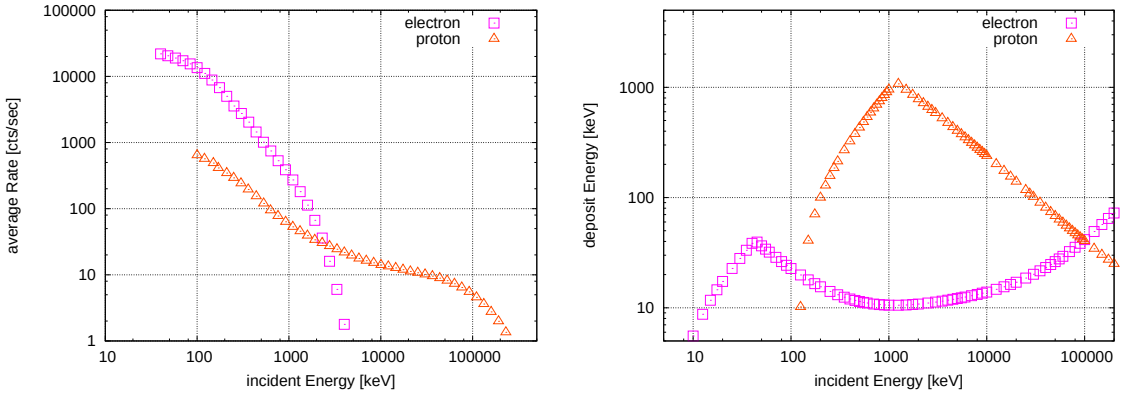


図 5.10: 衛星軌道上における陽子と電子のエネルギーに対する APD に対する衛星軌道上の平均計数率 (左図) とデポジットエネルギー (右図)。平均計数率は積分フラックスを用いて APD の受光面積 (0.096 $cm^2$) より算出した [9]。

図 5.10 には衛星軌道上における電子、そして陽子の入射粒子のエネルギーに対する衛星軌道上の計数率を平均化した値と APD へのデポジットエネルギーの値を示す。数え落しの原因はプリアンプの飽和にあり、計数率測定の正確さは APD に入射してくる粒子のデポジットエネルギーとその入射粒子の計数率に依存している。まず電子について考える。図 5.10 右を見てみると、低エネルギー領域での最大デポジットエネルギーは 30 keV で、これは入射エネルギーが 30 keV 程度のときである。この時の平均計数率はおよそ $\sim 10^4$ cts/s で、図 5.6 よりこのレートに関して数え落しをせずに計測できると考えられる。次に陽子の場合について考える。400 keV 以上のエネルギーをデポジットするのは、入射エネルギーが 0.5 MeV~4.0 MeV の時である。図 5.11 は衛星軌道上の陽子の積分フラックス分布を示しており、この図より 0.5 MeV 以上の積分フラックスは最大 $4\times10^4$ cts/s/$cm^2$ で、4.0 MeV 以上の積分フラックスは最大 $5\times10^3$ cts/s/$cm^2$ である。よって APD の受光面積 (3.5 mm$\phi$) を考慮すると、0.5 MeV~4.0 MeV の範囲での計数率は $3.4\times10^3$ cts/s と求まる。この計数率は図 5.6 に示されている 10 nsec で 436 keV 相当の LED 光を照射し

た時の数え過ぎるレート ($2\times10^3 \sim 2\times10^4$ cts/s) 内にある。一方、0.5 MeV 以下のエネルギーをデポジットする陽子については、平均計数率が $10^3$ cts/s 以下となっており、図 5.6 から正確に計数率を測定できると考えられる。

一連の実験は荷電粒子が 10 nsec 程度の時間で APD にエネルギーを付与するものとして考えてきたが、実際には荷電粒子は 1 nsec 以下の時間で付与する。本実験では実験設備に限界があり、LED を 1 nsec 以下で点滅させることができなかった。もしも 1nsec という短い時間で光を点滅させることができれば、先で述べた数え過ぎや数え落としが改善され、正確な計数率測定ができると考えられる。

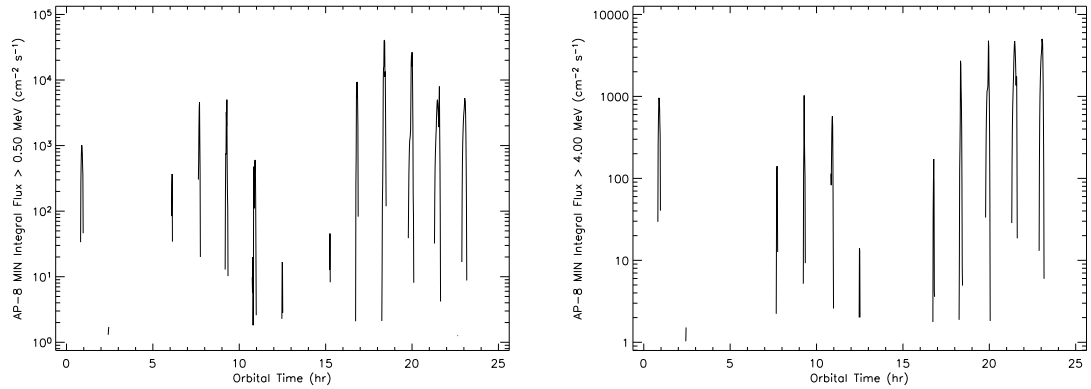


図 5.11: 太陽極小期における陽子の衛星軌道上の積分フラックス。左図は 0.5 MeV 以上の積分フラックスを表し、右図は 4.0 MeV 以上の積分フラックスを表す [10]。

## 5.7 結論

APD モジュールのミッションである荷電粒子探査では、30 keV 以下の低エネルギー領域まで計測を目的としている。現状の回路では、このエネルギー領域に対して 10 nsec の信号幅で入力される $10^6$ cts/s ものレートに対して、70 % 程度の計数能力が保障された。補足すると、荷電粒子が APD に入射する速度を考えれば、1 nsec 以下の短い時間で電子を作り出すと考えられるので、更なる数え落しの改善が期待される。また、衛星軌道上で実際に荷電粒子の計数率を測定できるか見積もりを行ったところ、電子ついては、ほぼ計数率を測定できることが確認できた。陽子については、0.5 MeV∼4.0 MeV のエネルギーをデポジットする場合において数え過ぎてしまう可能性がある。しかし、これもまた荷電粒子が 1 nsec 以下の短い時間で光を APD に落とすことを考えると、数え過ぎを起こさずに正確に計数率を測定できると考えられる。

# 第6章　まとめ

### [1] フライトモデル基板単体電気試験

4つのフライトモデル基板に対して単体電気試験を行った。電源基板の各電圧は正常に得られた。また、電源基板から生じるノイズは振幅 20 mV、周波数 100 $\mu$sec 程度で、アナログ回路にこの伝導ノイズが入り込んだ場合、電源電圧変動除去比 (PSRR) から 0.1 mV 程度にまで減衰されることが分かった。デジタル基板については、コマンドに対応した全ての動作が正常に機能することが確認できた。アナログ基板についてはテストパルス入力に対するプリアンプ (A225) 出力、ゲインアンプ、コンパレータ出力をオシロスコープでモニターし、全ての出力が外来ノイズに埋もれることなく、正常に動作していることを確認した。最後に、マザー基板に搭載されている DC-DC コンバータの入力電圧に対する出力電圧の線形性を確認した。当初、高電圧領域において線形性からのずれがあったが、AD648 周りの抵抗のパラメータを変更することにより、線型性からのずれを 0.3 % 以内に修正することができた。

### [2]APD 印加電圧自動制御試験

APD の増幅率は温度に対して変化してしまう。河合研究室ではこの問題を解決するべく、温度変化に対しても一定に保てるよう APD の印加電圧をマイコン H8 で自動制御する独自の制御システムを確立した。試験は調整用フライトモデル基板を用いて行った。恒温槽を用いて、衛星軌道上で考えられる温度範囲 ($-20$ ℃ ∼40 ℃) で温度サイクルを行った。このときの温度センサ出力、DAC チャンネル、そして ADC チャンネルを Windows マシンを用いてシリアル通信よりデータの取得を行った。試験の結果、$-20$ ℃ ∼30 ℃の範囲では温度に対応した高電圧出力が得られた。30 ℃以上では ADC のダイナミックレンジを超えてしまうが、軌道上でそのようなことが起きた場合は、ゲイン制御を行わないようにした。

フライトモデル基板に対しては、温度センサの代わりに直流安定化電源を用いて印加電圧制御試験を行った。得られた結果は、要求されている 1.5 V(1 ch) の誤差内に収まって

おり、印加制御システムは正常に機能することが確かめられた。

### [3] フライトモデル基板高計数試験

Cute-1.7APD モジュールのミッションの 1 つは、異常放射線帯における荷電粒子の計測である。この異常放射線帯で予想される $10^6$ cts/sec の計数率を APD モジュールが正しく計測できるかどうか試験を行った。試験はランダムパルサから 1 cts/s∼1 cts/s のレートを作り出し、LED 光を APD に照射した。試験結果は、信号の立上り時間を 10 nsec としたときに、$1.2\times10^6$ cts/s の入射レートに対して 70 % の $8\times10^5$ cts/s をカウントすることができた。

本論文では述べてこなかったが、全ての試験を終えた APD モジュールを組み込んだ Cute-1.7 衛星は、1 月 29 日に鹿児島県内之浦へシッピングし、JAXA MV-8 号機フライトオペレーションに参加した。そして、2 月 22 日午前 6 時 30 分頃、打ち上げに成功した。

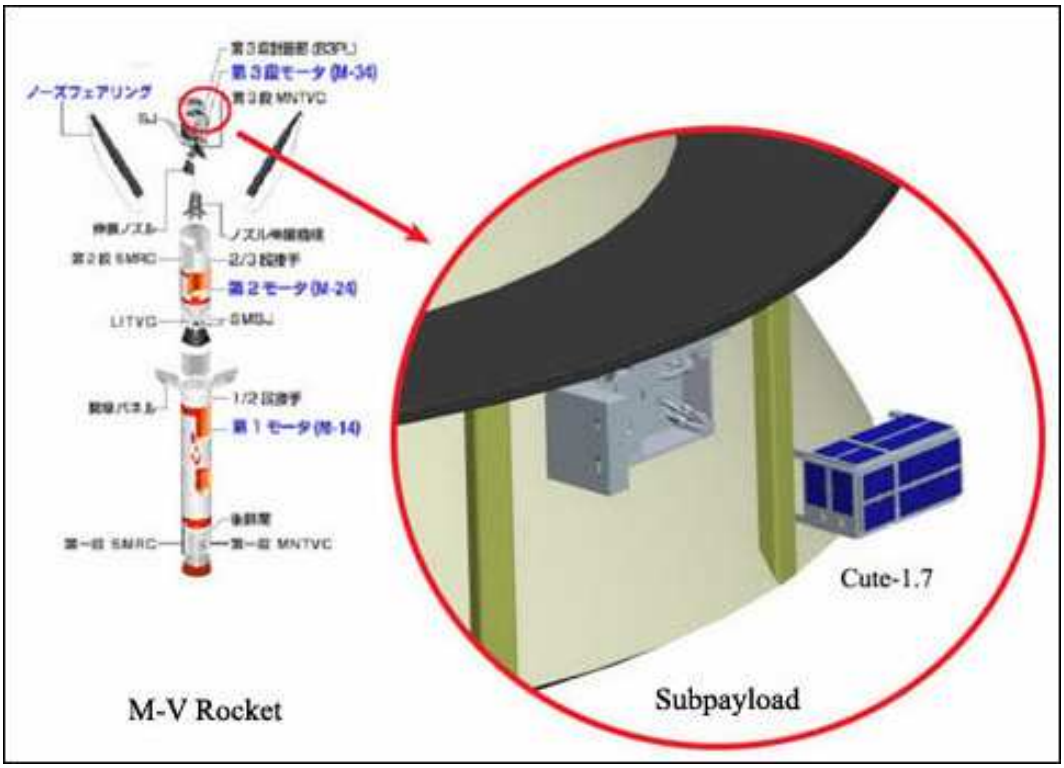


図 6.1: Cute-1.7 の M-V 8 号機サブペイロード。

# 関連図書


[1] T.Ikagawa,et.al,*Nucl Instr A*,Vol.538,p.640,December 2003

[2] 倉本 裕輔、修士論文「東工大衛星 Cute-1.7 搭載 APD 荷電粒子モニターの開発」東京工業大学 (2006)

[3] ANALOG MODULES, AD590 スペックシート

[4] ANALOG MODULES, Model 521A スペックシート

[5] 黒田 徹、「解析 OP アンプ & トランジスタ活用」CQ 出版社

[6] 野瀬製機, httb://www.noseseiki.com/

[7] Linear Technology, LT1111 スペックシート

[8] 五十川 知子、修士論文「宇宙利用に向けた X 線・$\gamma$ 線検出用アバランシェ・フォトダイオードの開発」東京工業大学 (2005)

[9] SPENVIS, http://www.spenvis.oma.be/spenvis/

[10] NIST, http://physics.nist.gov/

# 付 録A

## A.1　APD遮光漏れ

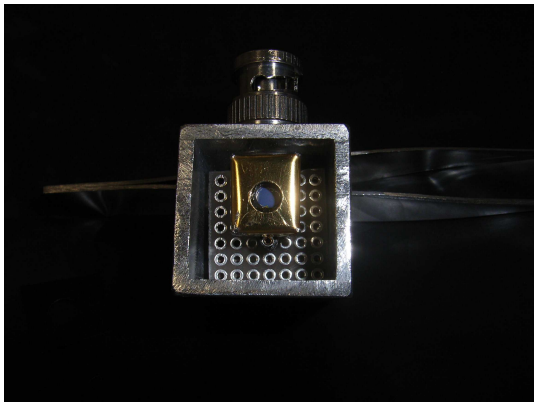

図 A.1: 遮光漏れ対策用の治具を取り付けたAPD。

APDには異常放射線帯の荷電粒子分布を調べるというミッションのために、APD受光面にAl蒸着を施し、さらに側面からの光洩れを防ぐために黒い樹脂で周囲を埋めている。しかし、一般に完全遮光を行ったときの暗電流値は印加電圧数10 Vに対し数nAであるのに対して、室内照明に晒し、印加電圧をかけない状態で暗電流の測定を行ったところ、100 nA程度の電流が流れた。これは、側面からの光りの洩れ込みのためであると考えられ、APD表面を覆う治具を取り付ける対策を行った(図A.1)。治具は0.1 mm厚の真鍮のパッケージと1 mm厚のアクリ板の中敷から構成され、その下にAPDをはめ込むことにした。受光面は図のように3.5 mm$\phi$となっている。この治具を取り付けた状態で遮光性能確認試験を行った。

### A.1.1　試験方法

太陽光の全輻射量は、$3.8 \times 10^{26}$ [W]である。これを地球近傍の単位面積当たりの輻射量に換算すると、$1.37 \times 10^{3}$ [W/m$^2$]となる。APDの受光面積は3.5 mm$\phi$であるので、

APD に照射する輻射量は 13.2[mW/3.5mm$\phi$] となる。このように算出された輻射量を、疑似的に APD に照射するために、LED を 800 mW で光らせることによって、受光面から 5 mm の距離から照射さ、24.4 [mW/3.5mm$\phi$] の輻射量を作り出した。このとき、光が拡散しないように指向性を高めて、APD に光を照射した。暗電流の測定には微小電流計 (keithley237) を用いて行った。

### A.1.2 結果と考察

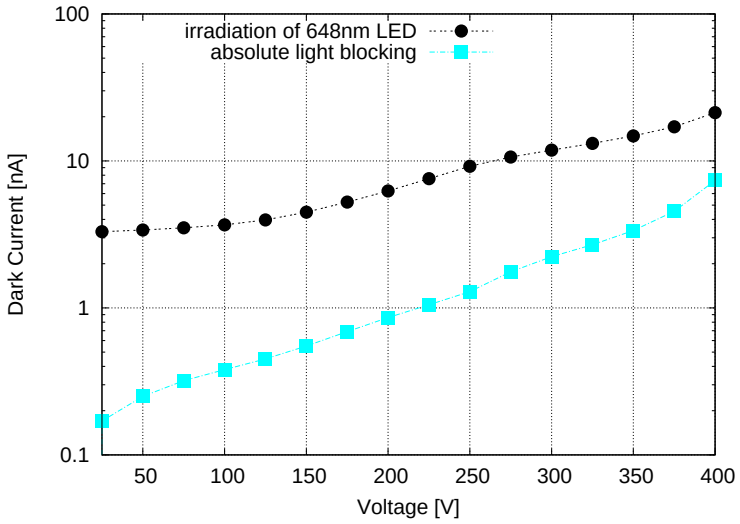


図 A.2: 完全遮光を行った場合 (□) と LED 光を照射した場合 (○) の印加電圧値に対する暗電流値。

試験結果を図 A.2 に示す。完全遮光を行った場合に比べて LED 光を照射したときの方が暗電流の値は大きく、光洩れの影響が見られる。しかし、電流値は印加電圧に対してあまり増大している様には見られないので、表面から入り込んだ光が増幅されているのではなく、素子の側面から何らかの形で洩れ込んできた光が暗電流の増大に大きく影響を及ぼしていると考えられる。

APD を増幅率 50 で使用する際にかける印加電圧の大きさは約 370 V 程度である。この時、完全遮光した時の暗電流の大きさは約 4 nA、LED 照射時には約 17 nA である。APD の電荷の回収に要する時間は 1 nsec 以下である事が知られている。仮に電荷回収時間を 1 nsec とすると、1 keV のエネルギー付与があった時に APD の出力する電流は 2.2 $\mu$A となる。以上のことから、光洩れの影響が少し見られたが、現在の治具を APD に取り付けることによって、低エネルギー粒子のシグナルを有意に検出できると考えられる。

## A.2　ランダムパルサの挙動

このようになってしまった背景には、検出器の持つ不感時間が大きく関わっている。不感時間とは検出器、増幅回路がシグナルを処理するのに必要な時間のことである。この不感時間を $\tau$ として検出器の最大応答速度はこれによって決まってしまう。検出器によっては不感時間中に入射したイベントに感度があるものと、そうでないものが存在する。前者を拡張型、後者を非拡張型という (図 A.3)。これらは、低いレートにおいてほとんど差が生じないが、不感時間に相当する周波数 ($=1/\tau$) 付近では大きく異なってくる。

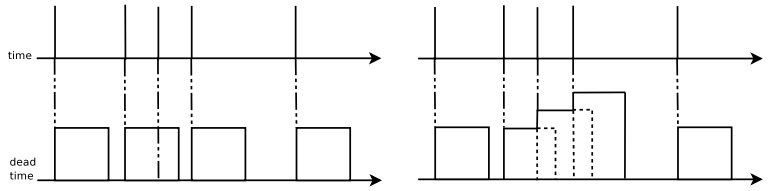


図 A.3: 非拡張型 (左) と拡張型 (右) の不感時間モデル。

### A.2.1　拡張型

検出器で計数できるのは不感時間 $\tau$ よりも長い時間間隔で入射してくるイベントである。不感時間 $\tau$ より長い時間間隔で放射線が検出器に入射する確率は、

$$P = \int_{\tau}^{\infty} I(t)dt \tag{A.1}$$

とできる。ここで $I(t)$ は入射イベントの時間間隔を変数とする関数である。検出器にイベントが生じる確率はポアソン統計に従う。ある時間 $t$ の間に発生したイベント数を $x$ としたとき、イベントが検出される確率 $P_x(t)$ を考える。$\epsilon$ を検出器の検出効率、$n$ を真の入射レートとすると、時間 $t$ に計数される平均のイベント数は $\epsilon nt$ となる。以上のことから $P_x(t)$ は式 (A.2) と書ける。

$$P_x(t) = \frac{(\epsilon nt)^x \exp[-\epsilon nt]}{x!} \tag{A.2}$$

特に $x = 0$ のとき、$P_0(t)$ は時間 $t$ までイベントが起きない確率を表している事が分かる。また $dt$ 間に起きるイベント数は $\epsilon ndt$ で表すことができる。よって不感時間より長い

時間間隔で入射するイベントが存在する確率は、

$$P = \int_{\tau}^{\infty} P_0(t)\epsilon n dt \tag{A.3}$$

$$= \int_{\tau}^{\infty} \epsilon n \exp[-\epsilon n t] dt \tag{A.4}$$

$$= \exp[-\epsilon n \tau] \tag{A.5}$$

となる。このことから観測したレートを $m$ とすれば、

$$m = \epsilon n \exp[-\epsilon n \tau] \tag{A.6}$$

となる。

### A.2.2 非拡張型

非拡張型の場合、不感時間 $\tau$ 中に検出器に入射したイベントには反応しない。そこで不感時間 $\tau$ 中に発生したイベント数の期待値 $N_{miss}$ を考える。時間 $\tau$ の間に $x$ 回イベントが発生する確率は、式 ( A.10) によって与えられる。

$$N_{miss} = \sum_{x=0}^{\infty} x P_x(\tau) \tag{A.7}$$

$$= \sum_{x=0}^{\infty} x \frac{(\epsilon n \tau)^x \exp[-\epsilon n \tau]}{x!} \tag{A.8}$$

$$= \sum_{x=0}^{\infty} \frac{(\epsilon n \tau)^{x-1} \exp[-\epsilon n \tau]}{(x-1)!} \tag{A.9}$$

$$= \epsilon n \tau \tag{A.10}$$

真の入射レートは観測されたイベントと、そのイベントによって生じた不感時間中に検出器に入ったイベントの和で与えられるので、

$$\epsilon n = m + m N_{miss} = m + m \epsilon n \tau \tag{A.11}$$

が成立する。これを整理すると、

$$m = \frac{\epsilon n}{1 + \epsilon n \tau} \tag{A.12}$$

となる。

## A.3　調整用フライトモデル基板温度試験

### A.3.1　試験内容

本試験は、フライトモデル基板に対して温度サイクル試験を行う前に様々な温度での基板の動作確認を行った。直接測定用のケーブルを半田付けをするため、フライトモデル基板を傷つけないためにも、フライトモデル基板と同等な調整用フライトモデル基板を用いて行った。試験内容は以下の通りである。

1. アナログシグナルの動作試験

2. スレッショルドレベルの出力確認試験

3. 電源電圧出力の確認試験

本試験は理学部河合研究室で行った。温度サイクルは実験室に置いてある恒温槽を用いて行った。試験項目がいくつかあるため、電気試験を行うために図 A.4 のような温度サイクルで行った。温度サイクルの手順は図に示されているように、温度変化を 10deg/h で行い、測定する温度になったら 1.5 時間恒温槽内部の温度を一定にしてから測定を行った。こうすることによって、恒温槽の温度と APD モジュールの温度を等しくすることができる。そして各温度について測定を行い、最終的に 40 ℃になったら APD モジュールを恒温槽から取り出した。最後に温度を 40 ℃にしたのは、恒温槽の開扉時の結露を防ぐためである。

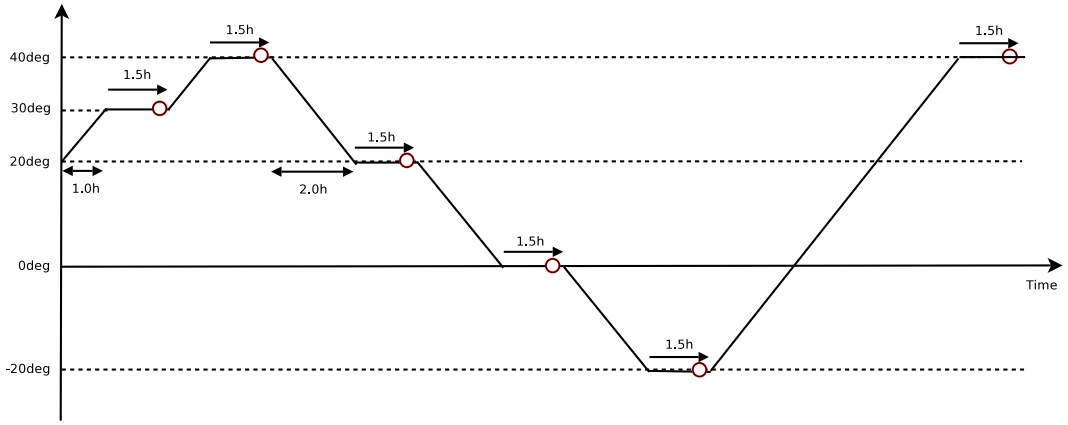


図 A.4: 電気試験を行ったときの温度変化の様子。

### A.3.2　温度サイクル下での電気試験の結果と考察

#### アナログシグナル動作試験の結果と考察

本試験ではAPDをアナログ基板に搭載せず、代わりにアナログ信号を作り出すものとしてテストパルスを用いた。このテストパルスは、NIMモジュールのGate Generatorによりパルスの幅と周期を決めて、NIM-TTL変換したのちアテネーターを介してパルスの高さを調節し、アナログ基板にテストパルスを送った。本試験で用いた恒温槽の温度サイクル (30℃→40℃→20℃→ 0℃→ −20℃→40℃) の順にオシロスコープで記録したテストパルス、ゲインアンプ、コンパレータの出力波形を図A.5～図A.10に示す。30℃においてのみテストパルスのパルスの高さが約20 mVとなっているため、テストパルスに乗っているノイズが相対的に大きく見える。しかし、後段回路のゲインアンプ、そしてコンパレータ出力に影響が出ていないことからノイズの影響は無視できることが分かる。他の温度では全て同じレベルのテストパルスを入れた。このレベルではノイズは相対的にとても小さく、ゲインアンプ、そしてコンパレータ出力波形がとてもきれいに整っている。さらにA系、B系には同じレベルのテストパルスを入力しているが、それぞれのシグナルを比較してみると、両系における差はまったくない。つまり、両系の個体差はないものと考えられる。以上のことから、−20℃～40℃におけるアナログ処理動作は温度サイクル下でも正常に機能していることが分かる。

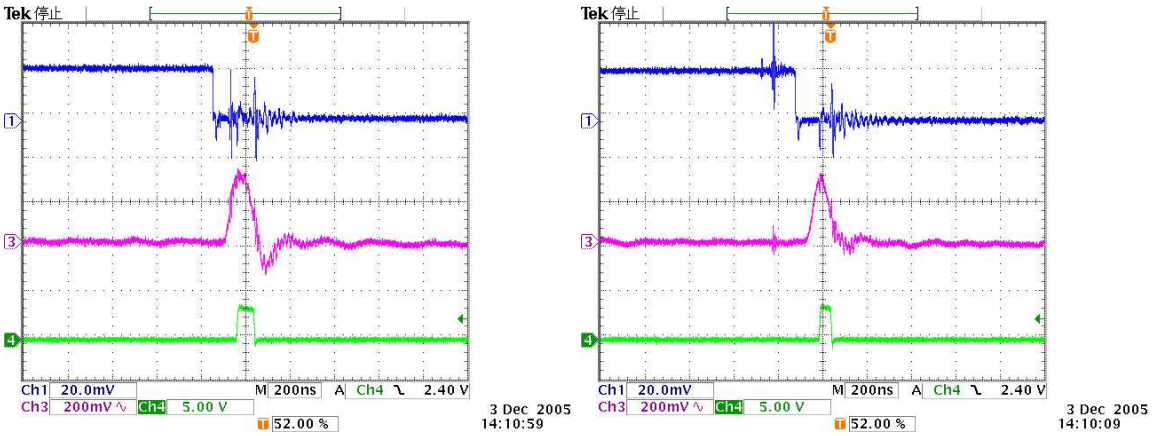


図 A.5: A 系 (左)、B 系 (右) の 30 ℃におけるアナログシグナル。テストパルス (1)、ゲインアンプ (2)、コンパレータ (3)。

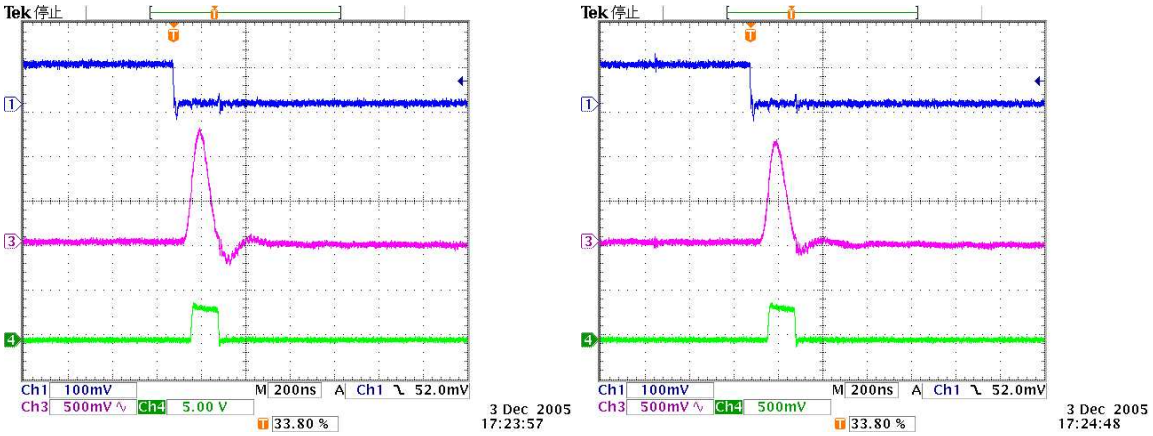


図 A.6: A 系 (左)、B 系 (右) の 40 ℃におけるアナログシグナル。テストパルス (1)、ゲインアンプ (2)、コンパレータ (3)。

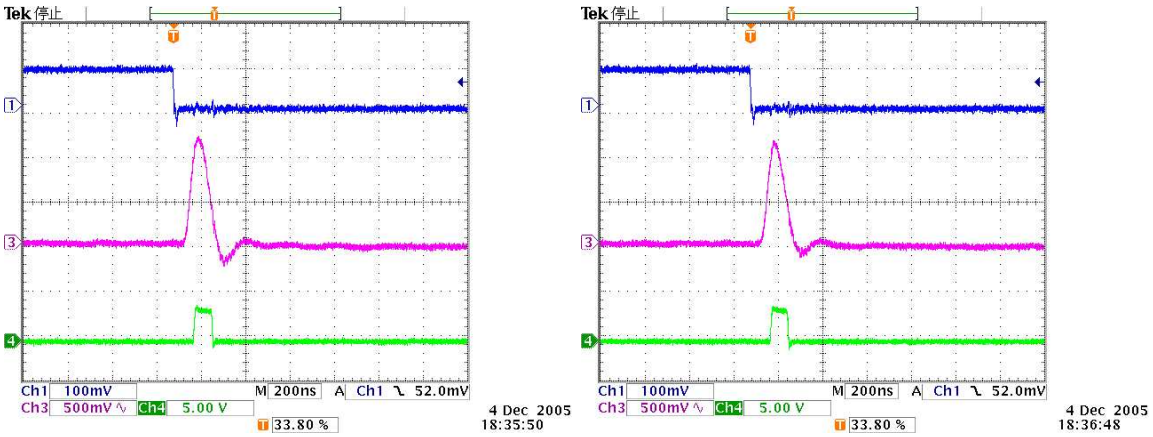


図 A.7: A 系 (左)、B 系 (右) の 20 ℃におけるアナログシグナル。テストパルス (1)、ゲインアンプ (2)、コンパレータ (3)。

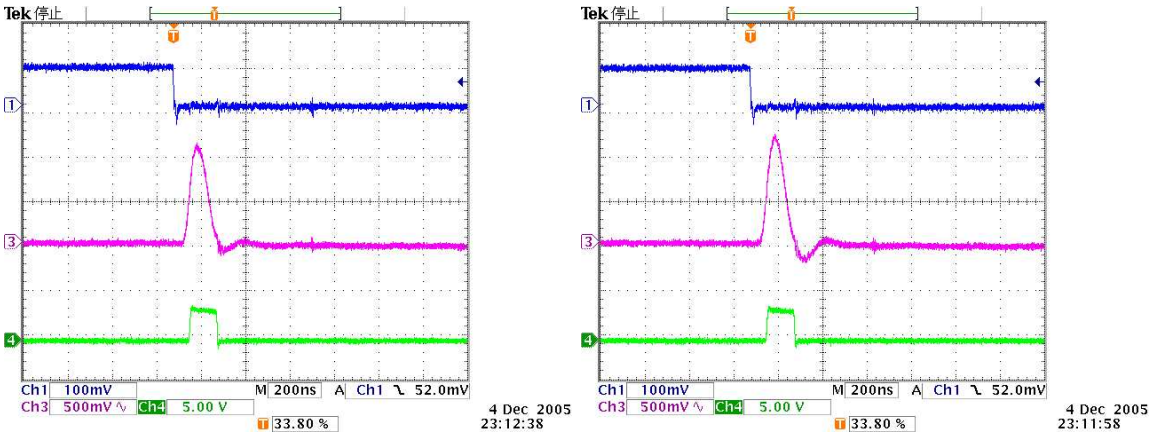


図 A.8: A 系 (左)、B 系 (右) の 0 ℃におけるアナログシグナル。テストパルス (1)、ゲインアンプ (2)、コンパレータ (3)。

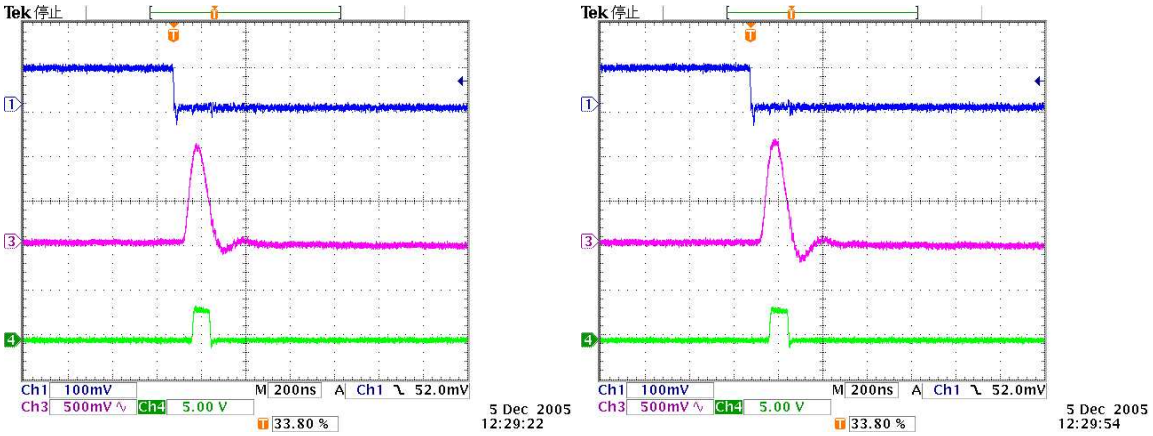


図 A.9: A 系 (左)、B 系 (右) の −20 ℃におけるアナログシグナル。テストパルス (1)、ゲインアンプ (2)、コンパレータ (3)。

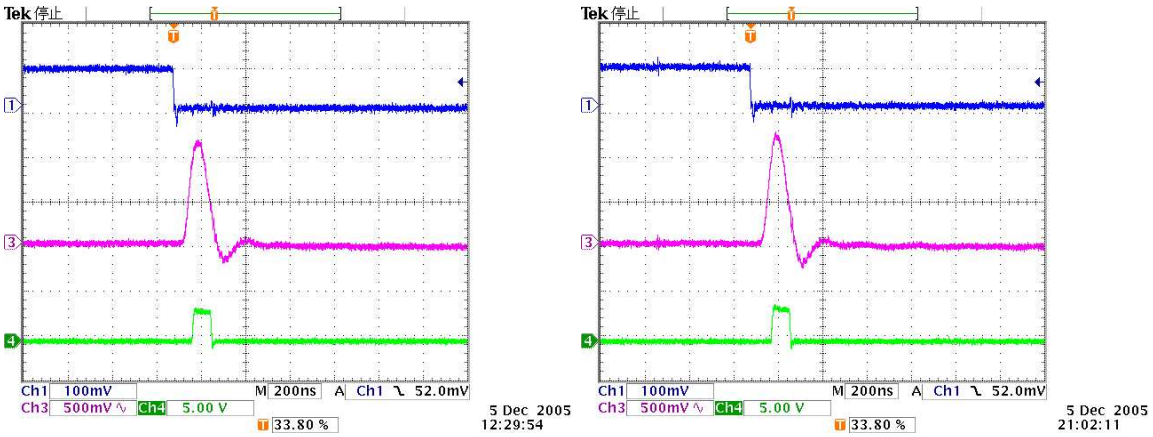


図 A.10: A 系 (左)、B 系 (右) の 40 ℃におけるアナログシグナル。テストパルス (1)、ゲインアンプ (2)、コンパレータ (3)。

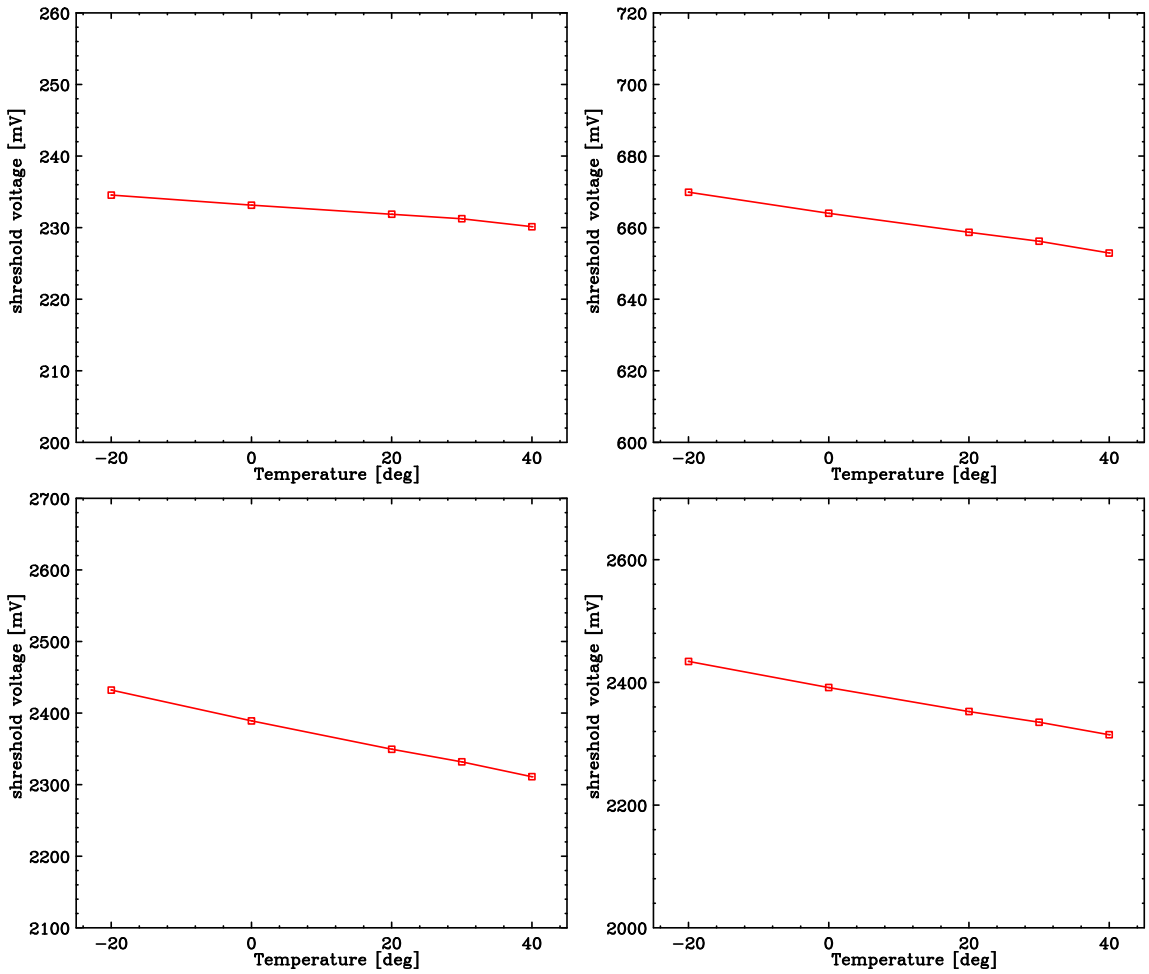


図 A.11: 温度に対するスレッショルドレベル。左上 : コマンド 0x30、右上 : コマンド 0x32、左下 : コマンド 0x31、右下 : コマンド 0x33。

アナログ基板のコンパレータに与えるスレッショルドレベルの温度変化についての結果を図 A.11 に示す。それぞれの図を見てみると、温度が高くなるにつれてスレッショルドレベルが線形に減少していることが読み取れる。これは、抵抗の温度依存性が関係していると考えられる。抵抗の温度依存性は絶対温度 T[K] に対して、式 (A.13) で表される。$R_0$ は規格抵抗値で、$\alpha$ は [Ω/K] で表される定数で、この定数は物質によって異なる。

$$R(T) = R_0 + \alpha T \tag{A.13}$$

式 (A.13) より、絶対温度の一次関数として抵抗値 $R(T)$ は一般的に上昇していく。スレッショルドレベル電圧は 4.1 V の入力電圧に対して抵抗分割を行って電圧値を決めているの

で、多くの抵抗を用いた方が温度に対するスレッショルド電圧値の変化率は大きいと考えられる (図 2.4 参照)。

調整用フライトモデル基板でのスレッショルドレベル電圧値の温度変化率と、20 ℃におけるスレッショルド電圧値 (SHD)、そしてフライトモデル基板でのスレッショルド電圧値を表 A.1 に示す。表より、2 つの基板のスレッショルドレベルの値にはずれが確認できる。スイッチング IC 周りの抵抗は 2 つの基板に対して同じものを使用し、かつ同じ回路になっているので、このずれは抵抗のパラメーターの誤差からくるものであると考えられる。

さて、式 (A.13) を見てみると、温度変化に対する抵抗値の変化は定数 $\alpha$ に依存していることが分かる。つまり、同じ種類、性能を持つ抵抗を用いたら、その温度依存性は全て等しく、フライトモデル基板についても表 A.1 に示されているスレッショルドレベルの温度変化率と同様な温度特性が得られると考えられる。

以上のことから、フライトモデル基板に対するスレッショルドレベルの温度依存性を示した式とそれをエネルギーに換算した式を表 A.2 に示す。これらの式から、衛星軌道上にある衛星の温度を知ることができれば、5 つのエネルギーレンジを見積もることができ、正しい計測を行うことができると考えられる。

| コマンド | 変化率 [mV/deg] | デバッグ用 FM[mV] | FM[mV] |
|---|---|---|---|
| 0x30 | −0.0708 | 232 | 219 |
| 0x31 | −0.2784 | 659 | 633 |
| 0x32 | −2.0012 | 2350 | 2080 |
| 0x33 | −1.9769 | 2353 | 2095 |

表 A.1: スレッショルドレベルの温度変化率と電圧値。

**電源電圧出力**

電源電圧出力についての結果を図 A.12 に示す。電源電圧出力におけるノイズは、オシロスコープで読み取るためのプローブの数に限りがあったので、本試験では見ることができなかった。図 A.12 を見ると、5 V、−5 V、12 V それぞれ各温度に対して安定した出力がなされていることが確認でき、さらに各電源系のドロップは見られなかったので、−20 ℃ ∼40 ℃の温度範囲において電源電圧出力は保証された。

| コマンド | 温度 $t$[deg] に対する電圧値 $V$[mV] | 温度 $t$[deg] に対するエネルギー $E$[keV] |
|---|---|---|
| 0x30 | $V = -0.0708t + 220.416$ | $E = -0.0029t + 9.24$ |
| 0x31 | $V = -0.2784t + 638.568$ | $E = -0.0114t + 26.17$ |
| 0x32 | $V = -2.0012t + 2120.02$ | $E = -0.0899t + 85.36$ |
| 0x33 | $V = -1.9769t + 2134.54$ | $E = -0.0809t + 85.97$ |

表 A.2: 温度 $t$[deg] に対するスレッショルドレベル電圧値 [mV] と換算エネルギー [keV]。

フライトモデル基板についても同様に、電源電圧出力を見る温度サイクル試験を行い、各電源系は正常に出力されていることが確かめられた [2]。

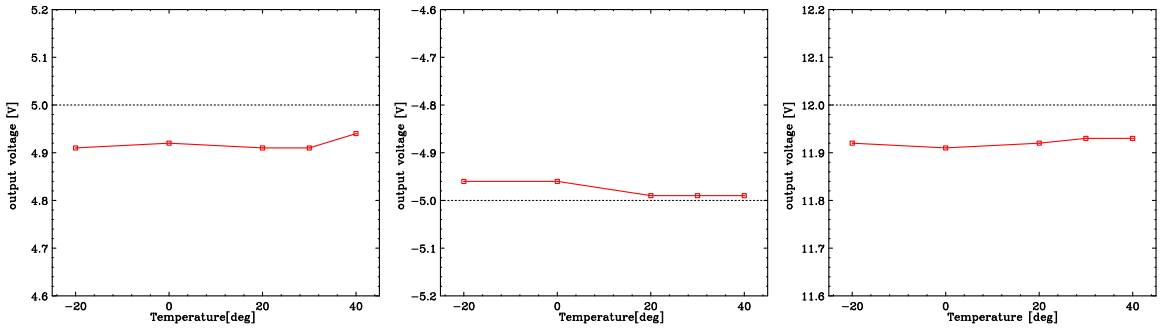


図 A.12: 温度に対する電源電圧出力。左：5V 電源、中央：−5V 電源、右：12V 電源

## A.4 印加電圧自動制御システムで要求される温度測定精度

APD モジュールは AD590 からの出力をマイコン H8 内の ADC を用いて温度変化を感知し、DAC により印加電圧を変えるという離散的な制御を行っている。実際に温度が変化している場合でも、温度センサの分解能が $\Delta T$ 以下の変化であれば、印加電圧の制御は行われず、その間は増幅率がずれてしまう。温度センサに要求される分解能は、フィードバックがかからない間にずれる増幅率 $\Delta G$ がエネルギー分解能に及ぼす影響から評価される。

あるエネルギー $E$ のイベントのシグナルは増幅器を通って、$\Delta E$ の精度で測定される。ADC に入力されるシグナルレベルを $V$、増幅器で規格化したシグナルレベルを $v$ とす

ると、

$$V = Gv \tag{A.14}$$

となる。増幅率が $\Delta G$ ずれた時、エネルギー分解能に及ぼす影響を誤差伝播として簡単に評価する。$\Delta V$ は式 (A.14) から、

$$\left(\frac{\Delta V}{V}\right)^2 \cong \left(\frac{\Delta G}{G}\right)^2 + \left(\frac{\Delta v}{v}\right)^2 \tag{A.15}$$

となり、測定されるエネルギー分解能に増幅率の変動が及ぼす影響は、電圧とエネルギーの関係が線形であることより、

$$\left(\frac{\Delta E}{E}\right)^2 = \left(\frac{\Delta V}{V}\right)^2 \tag{A.16}$$

$$\cong \left(\frac{\Delta v}{v}\right)^2 + \left(\frac{\Delta G}{G}\right)^2 \tag{A.17}$$

と書かれる。APD で $^{55}$Fe の X 線を 0 ℃で測定したとき、典型的なエネルギー分解能は 12 % である。ずれた増幅率 $\Delta G$ によってエネルギー分解能が 12.5 % まで悪化することを許容すると、

$$\left(\frac{\Delta G}{G}\right)^2 \cong \left(\frac{\Delta E}{E}\right)^2 - \left(\frac{\Delta v}{v}\right)^2 \tag{A.18}$$

$$\cong 12.5^2 - 12^2[\%^2] \tag{A.19}$$

$$\left(\frac{\Delta G}{G}\right) \cong 3.5[\%] \tag{A.20}$$

となる。温度変化による増幅率のずれは 5.2% まで許容されている。温度測定の分解能は式 (2.2) より、

$$\Delta T \cong 1.35[\mathrm{K}] \tag{A.21}$$

が必要となる。温度測定の精度は原理的に、ADC のビット数で決まる。測定温度範囲は −45 ℃ ∼30 ℃であるから、温度を 1.35[K] より良い分解能で測定するためには、ADC は 7 ビット以上のものである必要がある。

## A.5 H8 プログラム

以下に H8 マイコンに実装したプログラムを示す。

## iodefine.h

```
struct st_sam {                 /* struct DMAC  */
  void             *MAR;        /* MAR          */
  unsigned int      ETCR;       /* ETCR         */
  unsigned char     IOAR;       /* IOAR         */
  union {                       /* DTCR         */
    unsigned char BYTE;         /*  Byte Access */
    struct {                    /*  Bit  Access */
      unsigned char DTE :1;     /*    DTE       */
      unsigned char DTSZ:1;     /*    DTSZ      */
      unsigned char DTID:1;     /*    DTID      */
      unsigned char RPE :1;     /*    RPE       */
      unsigned char DTIE:1;     /*    DTIE      */
      unsigned char DTS :3;     /*    DTS       */
    }      BIT;
  }       DTCR;
};
struct st_fam {                 /* struct DMAC  */
  void             *MARA;       /* MARA         */
  unsigned int      ETCRA;      /* ETCRA        */
  char              wk1;
  union {                       /* DTCRA        */
    unsigned char BYTE;         /*  Byte Access */
    struct {                    /*  Bit  Access */
      unsigned char DTE  :1;    /*    DTE       */
      unsigned char DTSZ :1;    /*    DTSZ      */
      unsigned char SAID :1;    /*    SAID      */
      unsigned char SAIDE:1;    /*    SAIDE     */
      unsigned char DTIE :1;    /*    DTIE      */
      unsigned char DTS  :3;    /*    DTS       */
    }      BIT;
  }           DTCRA;
  void             *MARB;       /* MARB         */
  unsigned int      ETCRB;      /* ETCRB        */
  char              wk2;
  union {                       /* DTCRB        */
    unsigned char BYTE;         /*  Byte Access */
    struct {                    /*  Bit  Access */
      unsigned char DTME :1;    /*    DTME      */
      unsigned char      :1;    /*              */
      unsigned char DAID :1;    /*    DAID      */
      unsigned char DAIDE:1;    /*    DAIDE     */
      unsigned char TMS  :1;    /*    TMS       */
      unsigned char DTS  :3;    /*    DTS       */
    }      BIT;
  }           DTCRB;
};
struct st_flash {               /* struct FLASH */
  union {                       /* FLMCR        */
    unsigned char BYTE;         /*  Byte Access */
    struct {                    /*  Bit  Access */
      unsigned char VPP :1;     /*    VPP       */
      unsigned char VPPE:1;     /*    VPPE      */
      unsigned char     :2;     /*              */
      unsigned char EV  :1;     /*    EV        */
      unsigned char PV  :1;     /*    PV        */
      unsigned char E   :1;     /*    E         */
      unsigned char P   :1;     /*    P         */
    }      BIT;
  }         FLMCR;
  char             wk1;
  union {                       /* EBR1         */
    unsigned char BYTE;         /*  Byte Access */
    struct {                    /*  Bit  Access */
      unsigned char LB7:1;      /*    LB7       */
      unsigned char LB6:1;      /*    LB6       */
      unsigned char LB5:1;      /*    LB5       */
      unsigned char LB4:1;      /*    LB4       */
      unsigned char LB3:1;      /*    LB3       */
      unsigned char LB2:1;      /*    LB2       */
      unsigned char LB1:1;      /*    LB1       */
      unsigned char LB0:1;      /*    LB0       */
    }      BIT;
  }         EBR1;
  union {                       /* EBR2         */
    unsigned char BYTE;         /*  Byte Access */
    struct {                    /*  Bit  Access */
      unsigned char SB7:1;      /*    SB7       */
      unsigned char SB6:1;      /*    SB6       */
      unsigned char SB5:1;      /*    SB5       */
      unsigned char SB4:1;      /*    SB4       */
      unsigned char SB3:1;      /*    SB3       */
      unsigned char SB2:1;      /*    SB2       */
      unsigned char SB1:1;      /*    SB1       */
      unsigned char SB0:1;      /*    SB0       */
    }      BIT;
  }         EBR2;
  char             wk2[4];
  union {                       /* RAMCR        */
    unsigned char BYTE;         /*  Byte Access */
    struct {                    /*  Bit  Access */
      unsigned char FLER:1;     /*    FLER      */
      unsigned char     :3;     /*              */
      unsigned char RAMS:1;     /*    RAMS      */
      unsigned char RAM :3;     /*    RAM       */
    }      BIT;
  }         RAMCR;
};
struct st_itu {                   /* struct ITU   */
  union {                         /* TSTR         */
    unsigned char BYTE;           /*  Byte Access */
    struct {                      /*  Bit  Access */
      unsigned char     :3;       /*              */
      unsigned char STR4:1;       /*    STR4      */
      unsigned char STR3:1;       /*    STR3      */
      unsigned char STR2:1;       /*    STR2      */
      unsigned char STR1:1;       /*    STR1      */
      unsigned char STR0:1;       /*    STR0      */
    }      BIT;
  }           TSTR;
  union {                         /* TSNC         */
    unsigned char BYTE;           /*  Byte Access */
    struct {                      /*  Bit  Access */
      unsigned char      :3;      /*              */
      unsigned char SYNC4:1;      /*    SYNC4     */
      unsigned char SYNC3:1;      /*    SYNC3     */
      unsigned char SYNC2:1;      /*    SYNC2     */
      unsigned char SYNC1:1;      /*    SYNC1     */
      unsigned char SYNC0:1;      /*    SYNC0     */
    }      BIT;
  }           TSNC;
  union {                         /* TMDR         */
    unsigned char BYTE;           /*  Byte Access */
    struct {                      /*  Bit  Access */
      unsigned char     :1;       /*              */
      unsigned char MDF :1;       /*    MDF       */
      unsigned char FDIR:1;       /*    FDIR      */
      unsigned char PWM4:1;       /*    PWM4      */
      unsigned char PWM3:1;       /*    PWM3      */
      unsigned char PWM2:1;       /*    PWM2      */
      unsigned char PWM1:1;       /*    PWM1      */
      unsigned char PWM0:1;       /*    PWM0      */
    }      BIT;
  }           TMDR;
  union {                         /* TFCR         */
    unsigned char BYTE;           /*  Byte Access */
    struct {                      /*  Bit  Access */
      unsigned char     :2;       /*              */
      unsigned char CMD :2;       /*    CMD       */
      unsigned char BFB4:1;       /*    BFB4      */
      unsigned char BFA4:1;       /*    BFA4      */
      unsigned char BFB3:1;       /*    BFB3      */
      unsigned char BFA3:1;       /*    BFA3      */
    }      BIT;
  }           TFCR;
  char               wk[44];
  union {                         /* TOER         */
    unsigned char BYTE;           /*  Byte Access */
```

```
    struct {                    /*  Bit  Access */
      unsigned char     :2;     /*             */
      unsigned char EXB4:1;     /*    EXB4     */
      unsigned char EXA4:1;     /*    EXA4     */
      unsigned char EB3 :1;     /*    EB3      */
      unsigned char EB4 :1;     /*    EB4      */
      unsigned char EA4 :1;     /*    EA4      */
      unsigned char EA3 :1;     /*    EA3      */
    }      BIT;
  }          TOER;
  union {                       /* TOCR         */
    unsigned char BYTE;         /*  Byte Access */
    struct {                    /*  Bit  Access */
      unsigned char     :3;     /*             */
      unsigned char XTGD:1;     /*    XTGD     */
      unsigned char     :2;     /*             */
      unsigned char OLS4:1;     /*    OLS4     */
      unsigned char OLS3:1;     /*    OLS3     */
    }      BIT;
  }          TOCR;
};
struct st_itu0 {                /* struct ITU0  */
  union {                       /* TCR          */
    unsigned char BYTE;         /*  Byte Access */
    struct {                    /*  Bit  Access */
      unsigned char     :1;     /*             */
      unsigned char CCLR:2;     /*    CCLR     */
      unsigned char CKEG:2;     /*    CKEG     */
      unsigned char TPSC:3;     /*    TPSC     */
    }      BIT;
  }          TCR;
  union {                       /* TIOR         */
    unsigned char BYTE;         /*  Byte Access */
    struct {                    /*  Bit  Access */
      unsigned char    :1;      /*             */
      unsigned char IOB:3;      /*    IOB      */
      unsigned char    :1;      /*             */
      unsigned char IOA:3;      /*    IOA      */
    }      BIT;
  }          TIOR;
  union {                       /* TIER         */
    unsigned char BYTE;         /*  Byte Access */
    struct {                    /*  Bit  Access */
      unsigned char      :5;    /*             */
      unsigned char OVIE :1;    /*    OVIE     */
      unsigned char IMIEB:1;    /*    IMIEB    */
      unsigned char IMIEA:1;    /*    IMIEA    */
    }      BIT;
  }          TIER;
  union {                       /* TSR          */
    unsigned char BYTE;         /*  Byte Access */
    struct {                    /*  Bit  Access */
      unsigned char     :5;     /*             */
      unsigned char OVF :1;     /*    OVF      */
      unsigned char IMFB:1;     /*    IMFB     */
      unsigned char IMFA:1;     /*    IMFA     */
    }      BIT;
  }          TSR;
  unsigned int     TCNT;        /* TCNT         */
  unsigned int     GRA;         /* GRA          */
  unsigned int     GRB;         /* GRB          */
};                              /*              */
struct st_itu3 {                /* struct ITU3  */
  union {                       /* TCR          */
    unsigned char BYTE;         /*  Byte Access */
    struct {                    /*  Bit  Access */
      unsigned char     :1;     /*             */
      unsigned char CCLR:2;     /*    CCLR     */
      unsigned char CKEG:2;     /*    CKEG     */
      unsigned char TPSC:3;     /*    TPSC     */
    }      BIT;
  }          TCR;
  union {                       /* TIOR         */
    unsigned char BYTE;         /*  Byte Access */
    struct {                    /*  Bit  Access */
      unsigned char    :1;      /*             */
      unsigned char IOB:3;      /*    IOB      */
      unsigned char    :1;      /*             */
      unsigned char IOA:3;      /*    IOA      */
    }      BIT;
  }          TIOR;
  union {                       /* TIER         */
    unsigned char BYTE;         /*  Byte Access */
    struct {                    /*  Bit  Access */
      unsigned char      :5;    /*             */
      unsigned char OVIE :1;    /*    OVIE     */
      unsigned char IMIEB:1;    /*    IMIEB    */
      unsigned char IMIEA:1;    /*    IMIEA    */
    }      BIT;
  }          TIER;
  union {                       /* TSR          */
    unsigned char BYTE;         /*  Byte Access */
    struct {                    /*  Bit  Access */
      unsigned char     :5;     /*             */
      unsigned char OVF :1;     /*    OVF      */
      unsigned char IMFB:1;     /*    IMFB     */
      unsigned char IMFA:1;     /*    IMFA     */
    }      BIT;
  }          TSR;
  unsigned int     TCNT;        /* TCNT         */
  unsigned int     GRA;         /* GRA          */
  unsigned int     GRB;         /* GRB          */
  unsigned int     BRA;         /* BRA          */
  unsigned int     BRB;         /* BRB          */
};
struct st_tpc {                 /* struct TPC   */
  union {                       /* TPMR         */
    unsigned char BYTE;         /*  Byte Access */
    struct {                    /*  Bit  Access */
      unsigned char      :4;
      unsigned char G3NOV:1;    /*    G3NOV    */
      unsigned char G2NOV:1;    /*    G2NOV    */
      unsigned char G1NOV:1;    /*    G1NOV    */
      unsigned char GONOV:1;    /*    GONOV    */
    }      BIT;
  }          TPMR;
  union {                       /* TPCR         */
    unsigned char BYTE;         /*  Byte Access */
    struct {                    /*  Bit  Access */
      unsigned char G3CMS:2;    /*    G3CMS    */
      unsigned char G2CMS:2;    /*    G2CMS    */
      unsigned char G1CMS:2;    /*    G1CMS    */
      unsigned char GOCMS:2;    /*    GOCMS    */
    }      BIT;
  }          TPCR;
  union {                       /* NDERB        */
    unsigned char BYTE;         /*  Byte Access */
    struct {                    /*  Bit  Access */
      unsigned char B15:1;      /*    NDER15   */
      unsigned char B14:1;      /*    NDER14   */
      unsigned char B13:1;      /*    NDER13   */
      unsigned char B12:1;      /*    NDER12   */
      unsigned char B11:1;      /*    NDER11   */
      unsigned char B10:1;      /*    NDER10   */
      unsigned char B9 :1;      /*    NDER9    */
      unsigned char B8 :1;      /*    NDER8    */
    }      BIT;
  }          NDERB;
  union {                       /* NDERA        */
    unsigned char BYTE;         /*  Byte Access */
    struct {                    /*  Bit  Access */
      unsigned char B7:1;       /*    NDER7    */
      unsigned char B6:1;       /*    NDER6    */
      unsigned char B5:1;       /*    NDER5    */
      unsigned char B4:1;       /*    NDER4    */
      unsigned char B3:1;       /*    NDER3    */
      unsigned char B2:1;       /*    NDER2    */
      unsigned char B1:1;       /*    NDER1    */
      unsigned char B0:1;       /*    NDER0    */
    }      BIT;
```

```
  }           NDERA;
  union {                          /* NDRB (H'A4)  */
    unsigned char BYTE;            /*  Byte Access */
    struct {                       /*  Bit  Access */
      unsigned char B15:1;         /*    NDR15     */
      unsigned char B14:1;         /*    NDR14     */
      unsigned char B13:1;         /*    NDR13     */
      unsigned char B12:1;         /*    NDR12     */
      unsigned char B11:1;         /*    NDR11     */
      unsigned char B10:1;         /*    NDR10     */
      unsigned char B9 :1;         /*    NDR9      */
      unsigned char B8 :1;         /*    NDR8      */
    }      BIT;
  }           NDRB1;
  union {                          /* NDRA (H'A5)  */
    unsigned char BYTE;            /*  Byte Access */
    struct {                       /*  Bit  Access */
      unsigned char B7:1;          /*    NDR7      */
      unsigned char B6:1;          /*    NDR6      */
      unsigned char B5:1;          /*    NDR5      */
      unsigned char B4:1;          /*    NDR4      */
      unsigned char B3:1;          /*    NDR3      */
      unsigned char B2:1;          /*    NDR2      */
      unsigned char B1:1;          /*    NDR1      */
      unsigned char B0:1;          /*    NDR0      */
    }      BIT;
  }           NDRA1;
  union {                        /* NDRB (H'A6)  */
    unsigned char BYTE;          /*  Byte Access */
    struct {                     /*  Bit  Access */
      unsigned char    :4;       /*              */
      unsigned char B11:1;       /*    NDR11     */
      unsigned char B10:1;       /*    NDR10     */
      unsigned char B9 :1;       /*    NDR9      */
      unsigned char B8 :1;       /*    NDR8      */
    }      BIT;
  }           NDRB2;
  union {                        /* NDRA (H'A7)  */
    unsigned char BYTE;          /*  Byte Access */
    struct {                     /*  Bit  Access */
      unsigned char   :4;
      unsigned char B3:1;        /*    NDR3      */
      unsigned char B2:1;        /*    NDR2      */
      unsigned char B1:1;        /*    NDR1      */
      unsigned char B0:1;        /*    NDR0      */
    }      BIT;
  }           NDRA2;
};
union un_wdt {                   /* union WDT    */
  struct {                       /* Read  Access */
    union {                      /* TCSR         */
      unsigned char BYTE;        /*  Byte Access */
      struct {                   /*  Bit  Access */
unsigned char OVF :1;    /*    OVF       */
unsigned char WTIT:1;    /*    WT/IT     */
unsigned char TME :1;    /*    TME       */
unsigned char     :2;    /*              */
unsigned char CKS :3;    /*    CKS       */
      }      BIT;
    }       TCSR;
    unsigned char TCNT;          /* TCNT         */
    char          wk;
    union {                      /* RSTCSR       */
      unsigned char BYTE;        /*  Byte Access */
      struct {
unsigned char WRST :1;   /*    WSRT      */
unsigned char RSTOE:1;   /*    RSTOE     */
      }      BIT;
    }       RSTCSR;
  } READ;
  struct {                       /* Write Access */
    unsigned int  TCSR;          /* TCSR/TCNT    */
    unsigned int  RSTCSR;        /* RSTCSR       */
  } WRITE;
};
```

```
struct st_rfshc {                /* struct RFSHC */
  union {                        /* RFSHCR       */
    unsigned char BYTE;          /*  Byte Access */
    struct {                     /*  Bit  Access */
      unsigned char SRFMD :1;    /*    SRFMD     */
      unsigned char PSRAME:1;    /*    PSRAME    */
      unsigned char DRAME :1;    /*    DRAME     */
      unsigned char CASWE :1;    /*    CASWE     */
      unsigned char M9M8  :1;    /*    M9M8      */
      unsigned char RFSHE :1;    /*    RFSHE     */
      unsigned char       :1;    /*              */
      unsigned char RCYCE :1;    /*    RCYCE     */
    }      BIT;
  }         RFSHCR;
  union {                        /* RTMCSR       */
    unsigned char BYTE;          /*  Byte Access */
    struct {                     /*  Bit  Access */
      unsigned char CMF :1;      /*    CMF       */
      unsigned char CMIE:1;      /*    CMIE      */
      unsigned char CKS :3;      /*    CKS       */
    }      BIT;
  }         RTMCSR;
  unsigned char   RTCNT;         /* RTCNT        */
  unsigned char   RTCOR;         /* RTCOR        */
};
struct st_sci0 {                 /* struct SCI0  */
  union {                        /* SMR          */
    unsigned char BYTE;          /*  Byte Access */
    struct {                     /*  Bit  Access */
      unsigned char CA  :1;      /*    C/A       */
      unsigned char CHR :1;      /*    CHR       */
      unsigned char PE  :1;      /*    PE        */
      unsigned char OE  :1;      /*    O/E       */
      unsigned char STOP:1;      /*    STOP      */
      unsigned char MP  :1;      /*    MP        */
      unsigned char CKS :2;      /*    CKS       */
    }      BIT;
  }          SMR;
  unsigned char    BRR;          /* BRR          */
  union {                        /* SCR          */
    unsigned char BYTE;          /*  Byte Access */
    struct {                     /*  Bit  Access */
      unsigned char TIE :1;      /*    TIE       */
      unsigned char RIE :1;      /*    RIE       */
      unsigned char TE  :1;      /*    TE        */
      unsigned char RE  :1;      /*    RE        */
      unsigned char MPIE:1;      /*    MPIE      */
      unsigned char TEIE:1;      /*    TEIE      */
      unsigned char CKE :2;      /*    CKE       */
    }      BIT;
  }          SCR;
  unsigned char    TDR;          /* TDR          */
  union {                        /* SSR          */
    unsigned char BYTE;          /*  Byte Access */
    struct {                     /*  Bit  Access */
      unsigned char TDRE:1;      /*    TDRE      */
      unsigned char RDRF:1;      /*    RDRF      */
      unsigned char ORER:1;      /*    ORER      */
      unsigned char FER :1;      /*    FER       */
      unsigned char PER :1;      /*    PER       */
      unsigned char TEND:1;      /*    TEND      */
      unsigned char MPB :1;      /*    MPB       */
      unsigned char MPBT:1;      /*    MPBT      */
    }      BIT;
  }          SSR;
  unsigned char    RDR;          /* RDR          */
  union {                        /* SCMR         */
    unsigned char BYTE;          /*  Byte Access */
    struct {                     /*  Bit  Access */
      unsigned char     :4;      /*              */
      unsigned char SDIR:1;      /*    SDIR      */
      unsigned char SINV:1;      /*    SINV      */
      unsigned char     :1;      /*              */
      unsigned char SMIF:1;      /*    SMIF      */
    }      BIT;
```

```
  }          SCMR;
};
struct st_sci1 {                /* struct SCI1  */
  union {                       /* SMR          */
    unsigned char BYTE;         /*  Byte Access */
    struct {                    /*  Bit  Access */
      unsigned char CA  :1;     /*    C/A       */
      unsigned char CHR :1;     /*    CHR       */
      unsigned char PE  :1;     /*    PE        */
      unsigned char OE  :1;     /*    O/E       */
      unsigned char STOP:1;     /*    STOP      */
      unsigned char MP  :1;     /*    MP        */
      unsigned char CKS :2;     /*    CKS       */
    }      BIT;
  }          SMR;
  unsigned char    BRR;         /* BRR          */
  union {                       /* SCR          */
    unsigned char BYTE;         /*  Byte Access */
    struct {                    /*  Bit  Access */
      unsigned char TIE :1;     /*    TIE       */
      unsigned char RIE :1;     /*    RIE       */
      unsigned char TE  :1;     /*    TE        */
      unsigned char RE  :1;     /*    RE        */
      unsigned char MPIE:1;     /*    MPIE      */
      unsigned char TEIE:1;     /*    TEIE      */
      unsigned char CKE :2;     /*    CKE       */
    }      BIT;
  }          SCR;
  unsigned char    TDR;         /* TDR          */
  union {                       /* SSR          */
    unsigned char BYTE;         /*  Byte Access */
    struct {                    /*  Bit  Access */
      unsigned char TDRE:1;     /*    TDRE      */
      unsigned char RDRF:1;     /*    RDRF      */
      unsigned char ORER:1;     /*    ORER      */
      unsigned char FER :1;     /*    FER       */
      unsigned char PER :1;     /*    PER       */
      unsigned char TEND:1;     /*    TEND      */
      unsigned char MPB :1;     /*    MPB       */
      unsigned char MPBT:1;     /*    MPBT      */
    }      BIT;
  }          SSR;
  unsigned char    RDR;         /* RDR          */
};
struct st_smci {                /* struct SMCI  */
  union {                       /* SMR          */
    unsigned char BYTE;         /*  Byte Access */
    struct {                    /*  Bit  Access */
      unsigned char GM  :1;     /*    GM        */
      unsigned char CHR :1;     /*    CHR       */
      unsigned char PE  :1;     /*    PE        */
      unsigned char OE  :1;     /*    O/E       */
      unsigned char STOP:1;     /*    STOP      */
      unsigned char MP  :1;     /*    MP        */
      unsigned char CKS :2;     /*    CKS       */
    }      BIT;
  }          SMR;
  unsigned char    BRR;         /* BRR          */
  union {                       /* SCR          */
    unsigned char BYTE;         /*  Byte Access */
    struct {                    /*  Bit  Access */
      unsigned char TIE :1;     /*    TIE       */
      unsigned char RIE :1;     /*    RIE       */
      unsigned char TE  :1;     /*    TE        */
      unsigned char RE  :1;     /*    RE        */
      unsigned char MPIE:1;     /*    MPIE      */
      unsigned char TEIE:1;     /*    TEIE      */
      unsigned char CKE :2;     /*    CKE       */
    }      BIT;
  }          SCR;
  unsigned char    TDR;         /* TDR          */
  union {                       /* SSR          */
    unsigned char BYTE;         /*  Byte Access */
    struct {                    /*  Bit  Access */
      unsigned char TDRE:1;     /*    TDRE      */
      unsigned char RDRF:1;     /*    RDRF      */
      unsigned char ORER:1;     /*    ORER      */
      unsigned char ERS :1;     /*    ERS       */
      unsigned char PER :1;     /*    PER       */
      unsigned char TEND:1;     /*    TEND      */
      unsigned char MPB :1;     /*    MPB       */
      unsigned char MPBT:1;     /*    MPBT      */
    }      BIT;
  }          SSR;
  unsigned char    RDR;         /* RDR          */
  union {                       /* SCMR         */
    unsigned char BYTE;         /*  Byte Access */
    struct {                    /*  Bit  Access */
      unsigned char     :4;     /*              */
      unsigned char SDIR:1;     /*    SDIR      */
      unsigned char SINV:1;     /*    SINV      */
      unsigned char     :1;     /*              */
      unsigned char SMIF:1;     /*    SMIF      */
    }      BIT;
  }          SCMR;
};
struct st_p1 {                  /* struct P1    */
  unsigned char      DDR;       /* P1DDR        */
  char               wk;
  union {                       /* P1DR         */
    unsigned char BYTE;         /*  Byte Access */
    struct {                    /*  Bit  Access */
      unsigned char B7:1;       /*    Bit 7     */
      unsigned char B6:1;       /*    Bit 6     */
      unsigned char B5:1;       /*    Bit 5     */
      unsigned char B4:1;       /*    Bit 4     */
      unsigned char B3:1;       /*    Bit 3     */
      unsigned char B2:1;       /*    Bit 2     */
      unsigned char B1:1;       /*    Bit 1     */
      unsigned char B0:1;       /*    Bit 0     */
    }      BIT;
  }            DR;
};
struct st_p2 {                  /* struct P2    */
  unsigned char      DDR;       /* P2DDR        */
  char               wk1;
  union {                       /* P2DR         */
    unsigned char BYTE;         /*  Byte Access */
    struct {                    /*  Bit  Access */
      unsigned char B7:1;       /*    Bit 7     */
      unsigned char B6:1;       /*    Bit 6     */
      unsigned char B5:1;       /*    Bit 5     */
      unsigned char B4:1;       /*    Bit 4     */
      unsigned char B3:1;       /*    Bit 3     */
      unsigned char B2:1;       /*    Bit 2     */
      unsigned char B1:1;       /*    Bit 1     */
      unsigned char B0:1;       /*    Bit 0     */
    }      BIT;
  }            DR;
  char               wk2[20];
  union {                       /* P2PCR        */
    unsigned char BYTE;         /*  Byte Access */
    struct {                    /*  Bit  Access */
      unsigned char B7:1;       /*    Bit 7     */
      unsigned char B6:1;       /*    Bit 6     */
      unsigned char B5:1;       /*    Bit 5     */
      unsigned char B4:1;       /*    Bit 4     */
      unsigned char B3:1;       /*    Bit 3     */
      unsigned char B2:1;       /*    Bit 2     */
      unsigned char B1:1;       /*    Bit 1     */
      unsigned char B0:1;       /*    Bit 0     */
    }      BIT;
  }            PCR;
};
struct st_p3 {                  /* struct P3    */
  unsigned char      DDR;       /* P3DDR        */
  char               wk;        /*              */
  union {                       /* P3DR         */
    unsigned char BYTE;         /*  Byte Access */
    struct {                    /*  Bit  Access */
```

```
      unsigned char B7:1;           /*    Bit 7     */
      unsigned char B6:1;           /*    Bit 6     */
      unsigned char B5:1;           /*    Bit 5     */
      unsigned char B4:1;           /*    Bit 4     */
      unsigned char B3:1;           /*    Bit 3     */
      unsigned char B2:1;           /*    Bit 2     */
      unsigned char B1:1;           /*    Bit 1     */
      unsigned char B0:1;           /*    Bit 0     */
    }      BIT;
  }            DR;
};
struct st_p4 {                      /* struct P4    */
  unsigned char      DDR;           /* P4DDR        */
  char               wk1;
  union {                           /* P4DR         */
    unsigned char BYTE;             /*  Byte Access */
    struct {                        /*  Bit  Access */
      unsigned char B7:1;           /*    Bit 7     */
      unsigned char B6:1;           /*    Bit 6     */
      unsigned char B5:1;           /*    Bit 5     */
      unsigned char B4:1;           /*    Bit 4     */
      unsigned char B3:1;           /*    Bit 3     */
      unsigned char B2:1;           /*    Bit 2     */
      unsigned char B1:1;           /*    Bit 1     */
      unsigned char B0:1;           /*    Bit 0     */
    }      BIT;
  }            DR;
  char               wk2[18];
  union {                           /* P4PCR        */
    unsigned char BYTE;             /*  Byte Access */
    struct {                        /*  Bit  Access */
      unsigned char B7:1;           /*    Bit 7     */
      unsigned char B6:1;           /*    Bit 6     */
      unsigned char B5:1;           /*    Bit 5     */
      unsigned char B4:1;           /*    Bit 4     */
      unsigned char B3:1;           /*    Bit 3     */
      unsigned char B2:1;           /*    Bit 2     */
      unsigned char B1:1;           /*    Bit 1     */
      unsigned char B0:1;           /*    Bit 0     */
    }      BIT;
  }            PCR;
};
struct st_p5 {                      /* struct P5    */
  unsigned char      DDR;           /* P5DDR        */
  char               wk1;
  union {                           /* P5DR         */
    unsigned char BYTE;             /*  Byte Access */
    struct {                        /*  Bit  Access */
      unsigned char   :4;           /*               */
      unsigned char B3:1;           /*    Bit 3     */
      unsigned char B2:1;           /*    Bit 2     */
      unsigned char B1:1;           /*    Bit 1     */
      unsigned char B0:1;           /*    Bit 0     */
    }      BIT;
  }            DR;
  char               wk2[16];
  union {                           /* P5PCR        */
    unsigned char BYTE;             /*  Byte Access */
    struct {                        /*  Bit  Access */
      unsigned char   :4;           /*               */
      unsigned char B3:1;           /*    Bit 3     */
      unsigned char B2:1;           /*    Bit 2     */
      unsigned char B1:1;           /*    Bit 1     */
      unsigned char B0:1;           /*    Bit 0     */
    }      BIT;
  }            PCR;
};
struct st_p6 {                      /* struct P6    */
  unsigned char      DDR;           /* P6DDR        */
  char               wk;
  union {                           /* P6DR         */
    unsigned char BYTE;             /*  Byte Access */
    struct {                        /*  Bit  Access */
      unsigned char   :1;           /*               */
      unsigned char B6:1;           /*    Bit 6     */
      unsigned char B5:1;           /*    Bit 5     */
      unsigned char B4:1;           /*    Bit 4     */
      unsigned char B3:1;           /*    Bit 3     */
      unsigned char B2:1;           /*    Bit 2     */
      unsigned char B1:1;           /*    Bit 1     */
      unsigned char B0:1;           /*    Bit 0     */
    }      BIT;
  }            DR;
};
struct st_p7 {                      /* struct P7    */
  union {                           /* P7DR         */
    unsigned char BYTE;             /*  Byte Access */
    struct {                        /*  Bit  Access */
      unsigned char B7:1;           /*    Bit 7     */
      unsigned char B6:1;           /*    Bit 6     */
      unsigned char B5:1;           /*    Bit 5     */
      unsigned char B4:1;           /*    Bit 4     */
      unsigned char B3:1;           /*    Bit 3     */
      unsigned char B2:1;           /*    Bit 2     */
      unsigned char B1:1;           /*    Bit 1     */
      unsigned char B0:1;           /*    Bit 0     */
    }      BIT;
  }            DR;
};
struct st_p8 {                      /* struct P8    */
  unsigned char      DDR;           /* P8DDR        */
  char               wk;
  union {                           /* P8DR         */
    unsigned char BYTE;             /*  Byte Access */
    struct {                        /*  Bit  Access */
      unsigned char   :3;           /*               */
      unsigned char B4:1;           /*    Bit 4     */
      unsigned char B3:1;           /*    Bit 3     */
      unsigned char B2:1;           /*    Bit 2     */
      unsigned char B1:1;           /*    Bit 1     */
      unsigned char B0:1;           /*    Bit 0     */
    }      BIT;
  }            DR;
};
struct st_p9 {                      /* struct P9    */
  unsigned char      DDR;           /* P9DDR        */
  char               wk;
  union {                           /* P9DR         */
    unsigned char BYTE;             /*  Byte Access */
    struct {                        /*  Bit  Access */
      unsigned char   :2;           /*               */
      unsigned char B5:1;           /*    Bit 5     */
      unsigned char B4:1;           /*    Bit 4     */
      unsigned char B3:1;           /*    Bit 3     */
      unsigned char B2:1;           /*    Bit 2     */
      unsigned char B1:1;           /*    Bit 1     */
      unsigned char B0:1;           /*    Bit 0     */
    }      BIT;
  }            DR;
};
struct st_pa {                      /* struct PA    */
  unsigned char      DDR;           /* PADDR        */
  char               wk;
  union {                           /* PADR         */
    unsigned char BYTE;             /*  Byte Access */
    struct {                        /*  Bit  Access */
      unsigned char B7:1;           /*    Bit 7     */
      unsigned char B6:1;           /*    Bit 6     */
      unsigned char B5:1;           /*    Bit 5     */
      unsigned char B4:1;           /*    Bit 4     */
      unsigned char B3:1;           /*    Bit 3     */
      unsigned char B2:1;           /*    Bit 2     */
      unsigned char B1:1;           /*    Bit 1     */
      unsigned char B0:1;           /*    Bit 0     */
    }      BIT;
  }            DR;
};
struct st_pb {                      /* struct PB    */
  unsigned char      DDR;           /* PBDDR        */
  char               wk;
```

```
  union {                       /* PBDR         */
    unsigned char BYTE;         /*  Byte Access */
    struct {                    /*  Bit  Access */
      unsigned char B7:1;       /*    Bit 7     */
      unsigned char B6:1;       /*    Bit 6     */
      unsigned char B5:1;       /*    Bit 5     */
      unsigned char B4:1;       /*    Bit 4     */
      unsigned char B3:1;       /*    Bit 3     */
      unsigned char B2:1;       /*    Bit 2     */
      unsigned char B1:1;       /*    Bit 1     */
      unsigned char B0:1;       /*    Bit 0     */
    }      BIT;
  }        DR;
};
struct st_da {                  /* struct D/A   */
  union {                       /* DASTCR       */
    unsigned char BYTE;         /*  Byte Access */
    struct {                    /*  Bit  Access */
      unsigned char      :7;    /*              */
      unsigned char DASTE:1;    /*    DASTE     */
    }      BIT;
  }            DASTCR;
  char               wk[127];
  unsigned char      DADR0;     /* DADR0        */
  unsigned char      DADR1;     /* DADR1        */
  union {                       /* DACR         */
    unsigned char BYTE;         /*  Byte Access */
    struct {                    /*  Bit  Access */
      unsigned char DAOE1:1;    /*    DAOE1     */
      unsigned char DAOE0:1;    /*    DAOE0     */
      unsigned char DAE  :1;    /*    DAE       */
    }      BIT;
  }            DACR;
};
struct st_ad {                  /* struct A/D   */
  unsigned int       ADDRA;     /* ADDRA        */
  unsigned int       ADDRB;     /* ADDRB        */
  unsigned int       ADDRC;     /* ADDRC        */
  unsigned int       ADDRD;     /* ADDRD        */
  union {                       /* ADCSR        */
    unsigned char BYTE;         /*  Byte Access */
    struct {                    /*  Bit  Access */
      unsigned char ADF :1;     /*    ADF       */
      unsigned char ADIE:1;     /*    ADIE      */
      unsigned char ADST:1;     /*    ADST      */
      unsigned char SCAN:1;     /*    SCAN      */
      unsigned char CKS :1;     /*    CKS       */
      unsigned char CH  :3;     /*    CH        */
    }      BIT;
  }            ADCSR;
  union {                       /* ADCR         */
    unsigned char BYTE;         /*  Byte Access */
    struct {                    /*  Bit  Access */
      unsigned char TRGE:1;     /*    TRGE      */
    }      BIT;
  }            ADCR;
};
struct st_bsc {                 /* struct BSC   */
  union {                       /* CSCR         */
    unsigned char BYTE;         /*  Byte Access */
    struct {                    /*  Bit  Access */
      unsigned char CS7E:1;     /*    CS7E      */
      unsigned char CS6E:1;     /*    CS6E      */
      unsigned char CS5E:1;     /*    CS5E      */
      unsigned char CS4E:1;     /*    CS4E      */
    }      BIT;
  }           CSCR;
  char              wk1[140];
  union {                       /* ABWCR        */
    unsigned char BYTE;         /*  Byte Access */
    struct {                    /*  Bit  Access */
      unsigned char ABW7:1;     /*    ABW7      */
      unsigned char ABW6:1;     /*    ABW6      */
      unsigned char ABW5:1;     /*    ABW5      */
      unsigned char ABW4:1;     /*    ABW4      */
      unsigned char ABW3:1;     /*    ABW3      */
      unsigned char ABW2:1;     /*    ABW2      */
      unsigned char ABW1:1;     /*    ABW1      */
      unsigned char ABW0:1;     /*    ABW0      */
    }      BIT;
  }           ABWCR;
  union {                       /* ASTCR        */
    unsigned char BYTE;         /*  Byte Access */
    struct {                    /*  Bit  Access */
      unsigned char AST7:1;     /*    AST7      */
      unsigned char AST6:1;     /*    AST6      */
      unsigned char AST5:1;     /*    AST5      */
      unsigned char AST4:1;     /*    AST4      */
      unsigned char AST3:1;     /*    AST3      */
      unsigned char AST2:1;     /*    AST2      */
      unsigned char AST1:1;     /*    AST1      */
      unsigned char AST0:1;     /*    AST0      */
    }      BIT;
  }           ASTCR;
  union {                       /* WCR          */
    unsigned char BYTE;         /*  Byte Access */
    struct {                    /*  Bit  Access */
      unsigned char    :4;      /*              */
      unsigned char WMS:2;      /*    WMS       */
      unsigned char WC :2;      /*    WC        */
    }      BIT;
  }           WCR;
  union {                       /* WCER         */
    unsigned char BYTE;         /*  Byte Access */
    struct {                    /*  Bit  Access */
      unsigned char WCE7:1;     /*    WCE7      */
      unsigned char WCE6:1;     /*    WCE6      */
      unsigned char WCE5:1;     /*    WCE5      */
      unsigned char WCE4:1;     /*    WCE4      */
      unsigned char WCE3:1;     /*    WCE3      */
      unsigned char WCE2:1;     /*    WCE2      */
      unsigned char WCE1:1;     /*    WCE1      */
      unsigned char WCE0:1;     /*    WCE0      */
    }      BIT;
  }           WCER;
  char              wk2[3];
  union {                       /* BRCR         */
    unsigned char BYTE;         /*  Byte Access */
    struct {                    /*  Bit  Access */
      unsigned char A23E:1;     /*    A23E      */
      unsigned char A22E:1;     /*    A22E      */
      unsigned char A21E:1;     /*    A21E      */
      unsigned char     :4;     /*              */
      unsigned char BRLE:1;     /*    BRLE      */
    }      BIT;
  }           BRCR;
};
union un_divcr {                /* union DIVCR  */
  unsigned char BYTE;           /*  Byte Access */
  struct {                      /*  Bit  Access */
    unsigned char    :6;
    unsigned char DIV:2;        /*    DIV       */
  }      BIT;
};
union un_mstcr {                /* union MSTCR  */
  unsigned char BYTE;           /*  Byte Access */
  struct {                      /*  Bit  Access */
    unsigned char PSTOP :1;     /*    PSTOP     */
    unsigned char       :1;
    unsigned char _ITU  :1;     /*    MSTOP5    */
    unsigned char _SCI0 :1;     /*    MSTOP4    */
    unsigned char _SCI1 :1;     /*    MSTOP3    */
    unsigned char _DMAC :1;     /*    MSTOP2    */
    unsigned char _RFSHC:1;     /*    MSTOP1    */
    unsigned char _AD   :1;     /*    MSTOP0    */
  }      BIT;
};
union un_mdcr {                 /* union MDCR   */
  unsigned char BYTE;           /*  Byte Access */
  struct {                      /*  Bit  Access */
```

```
    unsigned char    :5;
    unsigned char MDS:3;          /*    MDS       */
  }      BIT;
};
union un_syscr {                  /* union SYSCR  */
  unsigned char BYTE;             /*  Byte Access */
  struct {                        /*  Bit  Access */
    unsigned char SSBY :1;        /*    SSBY      */
    unsigned char STS  :3;        /*    STS       */
    unsigned char UE   :1;        /*    UE        */
    unsigned char NMIEG:1;        /*    NMIEG     */
    unsigned char      :1;        /*              */
    unsigned char RAME :1;        /*    RAME      */
  }      BIT;
};
struct st_intc {                  /* struct INTC  */
  union {                         /* ISCR         */
    unsigned char BYTE;           /*  Byte Access */
    struct {                      /*  Bit  Access */
      unsigned char       :2;     /*              */
      unsigned char IRQ5SC:1;     /*    IRQ5SC    */
      unsigned char IRQ4SC:1;     /*    IRQ4SC    */
      unsigned char IRQ3SC:1;     /*    IRQ3SC    */
      unsigned char IRQ2SC:1;     /*    IRQ2SC    */
      unsigned char IRQ1SC:1;     /*    IRQ1SC    */
      unsigned char IRQ0SC:1;     /*    IRQ0SC    */
    }      BIT;
  }          ISCR;
  union {                         /* IER          */
    unsigned char BYTE;           /*  Byte Access */
    struct {                      /*  Bit  Access */
      unsigned char      :2;      /*              */
      unsigned char IRQ5E:1;      /*    IRQ5E     */
      unsigned char IRQ4E:1;      /*    IRQ4E     */
      unsigned char IRQ3E:1;      /*    IRQ3E     */
      unsigned char IRQ2E:1;      /*    IRQ2E     */
      unsigned char IRQ1E:1;      /*    IRQ1E     */
      unsigned char IRQ0E:1;      /*    IRQ0E     */
    }      BIT;
  }          IER;
  union {                         /* ISR          */
    unsigned char BYTE;           /*  Byte Access */
    struct {                      /*  Bit  Access */
      unsigned char      :2;      /*              */
      unsigned char IRQ5F:1;      /*    IRQ5F     */
      unsigned char IRQ4F:1;      /*    IRQ4F     */
      unsigned char IRQ3F:1;      /*    IRQ3F     */
      unsigned char IRQ2F:1;      /*    IRQ2F     */
      unsigned char IRQ1F:1;      /*    IRQ1F     */
      unsigned char IRQ0F:1;      /*    IRQ0F     */
    }      BIT;
  }          ISR;
  char             wk;
  union {                         /* IPRA         */
    unsigned char BYTE;           /*  Byte Access */
    struct {                      /*  Bit  Access */
      unsigned char _IRQ0 :1;     /*    IRQ0      */
      unsigned char _IRQ1 :1;     /*    IRQ1      */
      unsigned char _IRQ23:1;     /*    IRQ2,IRQ3 */
      unsigned char _IRQ45:1;     /*    IRQ4,IRQ5 */
      unsigned char _WDT  :1;     /*    WDT,RFSHC */
      unsigned char _ITU0 :1;     /*    ITU0      */
      unsigned char _ITU1 :1;     /*    ITU1      */
      unsigned char _ITU2 :1;     /*    ITU2      */
    }      BIT;
  }          IPRA;
  union {                         /* IPRB         */
    unsigned char BYTE;           /*  Byte Access */
    struct {                      /*  Bit  Access */
      unsigned char _ITU3:1;      /*    ITU3      */
      unsigned char _ITU4:1;      /*    ITU4      */
      unsigned char _DMAC:1;      /*    DMAC      */
      unsigned char      :1;      /*              */
      unsigned char _SCI0:1;      /*    SCI0      */
      unsigned char _SCI1:1;      /*    SCI1      */
      unsigned char _AD  :1;      /*    A/D       */
    }      BIT;
  }          IPRB;
};                                                /*              */
#define DMAC0A  (*(volatile struct st_sam   *)0xFFFF20) /* DMAC 0A Addr */
#define DMAC0B  (*(volatile struct st_sam   *)0xFFFF28) /* DMAC 0B Addr */
#define DMAC1A  (*(volatile struct st_sam   *)0xFFFF30) /* DMAC 1A Addr */
#define DMAC1B  (*(volatile struct st_sam   *)0xFFFF38) /* DMAC 1B Addr */
#define DMAC0   (*(volatile struct st_fam   *)0xFFFF20) /* DMAC 0  Addr */
#define DMAC1   (*(volatile struct st_fam   *)0xFFFF30) /* DMAC 1  Addr */
#define ITU     (*(volatile struct st_itu   *)0xFFFF60) /* ITU   Address*/
#define ITU0    (*(volatile struct st_itu0  *)0xFFFF64) /* ITU0  Address*/
#define ITU1    (*(volatile struct st_itu0  *)0xFFFF6E) /* ITU1  Address*/
#define ITU2    (*(volatile struct st_itu0  *)0xFFFF78) /* ITU2  Address*/
#define ITU3    (*(volatile struct st_itu3  *)0xFFFF82) /* ITU3  Address*/
#define ITU4    (*(volatile struct st_itu3  *)0xFFFF92) /* ITU4  Address*/
#define TPC     (*(volatile struct st_tpc   *)0xFFFFA0) /* TPC   Address*/
#define WDT     (*(volatile union  un_wdt   *)0xFFFFA8) /* WDT   Address*/
#define RFSHC   (*(volatile struct st_rfshc *)0xFFFFAC) /* RFSHC Address*/
#define SCI0    (*(volatile struct st_sci0  *)0xFFFFB0) /* SCI0  Address*/
#define SCI1    (*(volatile struct st_sci1  *)0xFFFFB8) /* SCI1  Address*/
#define SMCI0   (*(volatile struct st_smci  *)0xFFFFB0) /* SMCI0 Address*/
#define P1      (*(volatile struct st_p1    *)0xFFFFC0) /* P1    Address*/
#define P2      (*(volatile struct st_p2    *)0xFFFFC1) /* P2    Address*/
#define P3      (*(volatile struct st_p3    *)0xFFFFC4) /* P3    Address*/
#define P4      (*(volatile struct st_p4    *)0xFFFFC5) /* P4    Address*/
#define P5      (*(volatile struct st_p5    *)0xFFFFC8) /* P5    Address*/
#define P6      (*(volatile struct st_p6    *)0xFFFFC9) /* P6    Address*/
#define P7      (*(volatile struct st_p7    *)0xFFFFCE) /* P7    Address*/
#define P8      (*(volatile struct st_p8    *)0xFFFFCD) /* P8    Address*/
#define P9      (*(volatile struct st_p9    *)0xFFFFD0) /* P9    Address*/
#define PA      (*(volatile struct st_pa    *)0xFFFFD1) /* PA    Address*/
#define PB      (*(volatile struct st_pb    *)0xFFFFD4) /* PB    Address*/
#define DA      (*(volatile struct st_da    *)0xFFFF5C) /* D/A   Address*/
#define AD      (*(volatile struct st_ad    *)0xFFFFE0) /* A/D   Address*/
#define BSC     (*(volatile struct st_bsc   *)0xFFFF5F) /* BSC   Address*/
#define FLASH   (*(volatile struct st_flash *)0xFFFF40) /* FLASH Address*/
#define DIVCR   (*(volatile union  un_divcr *)0xFFFF5D) /* DIVCR Address*/
#define MSTCR   (*(volatile union  un_mstcr *)0xFFFF5E) /* MSTCR Address*/
#define MDCR    (*(volatile union  un_mdcr  *)0xFFFFF1) /* MDCR  Address*/
#define SYSCR   (*(volatile union  un_syscr *)0xFFFFF2) /* SYSCR Address*/
#define INTC    (*(volatile struct st_intc  *)0xFFFFF4) /* INTC  Address*/
```

## sbrk.h

```
/* size of area managed by sbrk */
#define HEAPSIZE 0x420
```

## stacksct.h

```
#pragma stacksize 0x200  /* Do not modify this line. */
```

## usbio.h

```
#ifndef __USBIO_H__
#define __USBIO_H__

/*

AKI-H8-USB IORg[

2002.4 (C)C.I.M

AKI-H8-USBM[‘BfoCXhCoMB
```

```
Oobt@Busbstart()TCY
wBAAKI-H8-USB{256oCg
AxB
A'\0'IB'\0'USBMB
MAXbhsB
^COB

*/

/*-------------------------------------------------------
  USBXbhJn

  int buffsize      USBMobt@TCY
  int timer       Xbh[v^C}[(ms)
  USBpoB
  obt@AUSBMXbhJnB
  sA-1B
  AUSBXbhdNB
  VXenOAbv
  -------------------------------------------------------*/
extern int usbstart(int buffsize,int timer);


/*-------------------------------------------------------
  USBXbhI
  USBXbh~AnhN[YB
-------------------------------------------------------*/
extern void stopusb(void);


/*-------------------------------------------------------
  USBXbh

  l 0 = XbhN
  1 = XbhN
  2 = Xbh~
  -------------------------------------------------------*/
extern int getusbactive(void);


/*-------------------------------------------------------
  USB2
  l bit0 ... XbhN
  bit1 ... 1 (\)
  bit2 ... 1=READ Handle Open
  bit3 ... 1=WRITE Handle Open
  NAOxfA
  USBAOx3
  -------------------------------------------------------*/
extern int getusbstat(void);


/*-------------------------------------------------------
  USBXbhJE^
  [vJEgAbvB'FbNpB
  -------------------------------------------------------*/
extern  int getusbcount(void);

/*-------------------------------------------------------
  f[^M
  char *buff Mf[^|C^
  int size f[^TCY
  l       MTCY

  USBTCYf[^MB
  -------------------------------------------------------*/
extern int write_buff(char *buff,int size);


/*-------------------------------------------------------
  f[^M
  char *buff obt@
  int size obt@TCY
  l MTCY

  USBMAMobt@f[^B
-------------------------------------------------------*/
extern int read_buff(char *buff,int size);


/*-------------------------------------------------------
  Mf[^TCY
  l       MTCY

  USBMobt@cTCYB
  -------------------------------------------------------*/
extern int get_inbufflen(void);

/*-------------------------------------------------------
  Mobt@f[^TCY
  l Mobt@cTCY

  Mobt@USBMf[^TCYB
-------------------------------------------------------*/
extern int get_outbufflen(void);


extern  int usbreadcnt;
extern  int usbwritecnt;

#endif
```

# USBN9602.H

```
/*=========================================================
N9604 Address
=========================================================*/
#define USB9602R (*(volatile unsigned char *)0x400003)
#define USB9602D (*(volatile unsigned char *)0x400001)

/*=========================================================
N9604 Define
=========================================================*/
#define USB_CLKDIV 0x04
/* CLKOUT = 48MHz/4 = 12MHz */


/* USB1.ONGXg */
#define USB_GET_STATUS 0
#define USB_CLEAR_FEATURE 1
#define USB_SET_FEATURE 3
#define USB_SET_ADDRESS 5
#define USB_GET_DESCRIPTOR 6
#define USB_SET_DESCRIPTOR 7
#define USB_GET_CONFIGURATION 8
#define USB_SET_CONFIGURATION 9
#define USB_GET_INTERFACE 10
#define USB_SET_INTERFACE 11
#define USB_SYNCH_FRAME 12

/* fBXNv^ */
#define USB_DEVICE 1
#define USB_CONFIGURATION 2
#define USB_XSTRING 3
#define USB_INTERFACE 4
#define USB_ENDPOINT 5
#define USB_HID 0x21
#define USB_HIDREPORT 0x22
#define USB_HIDPHYSICAL 0x23

/* HIDNGXg */
#define USB_GET_REPORT 0x01
#define USB_GET_IDLE 0x02
#define USB_GET_PROTOCOL 0x03
#define USB_SET_REPORT 0x09
```

```
#define USB_SET_IDLE 0x0A
#define USB_SET_PROTOCOL 0x0B

/*=====================================================
N9604 Register
=====================================================*/
#define USB_MCNTRL 0x00 /*Main control register*/
#define USB_CCONF 0x01 /*Clk. config. register*/
#define USB_TCR 0x02 /*Xcvr config. register*/
#define USB_RID 0x03 /*Rev. ID register*/
#define USB_FAR 0x04 /*Func address register*/
#define USB_NFSR 0x05 /*Node func st register*/
#define USB_MAEV 0x06 /*Main event   register*/
#define USB_MAMSK 0x07 /*Main mask    register*/
#define USB_ALTEV 0x08 /*Alt. event   register*/
#define USB_ALTMSK 0x09 /*ALT  mask    register*/
#define USB_TXEV 0x0A /*TX   event   register*/
#define USB_TXMSK 0x0B /*TX   mask    register*/
#define USB_RXEV 0x0C /*RX   event   register*/
#define USB_RXMSK 0x0D /*RX   mask    register*/
#define USB_NAKEV 0x0E /*NAK  event   register*/
#define USB_NAKMSK 0x0F /*NAK  mask    register*/
#define USB_FWEV 0x10 /*FIFO warning register*/
#define USB_FWMSK 0x11 /*FIFO warning mask*/
#define USB_FNH 0x12 /*Frame nbr hi register*/
#define USB_FNL 0x13 /*Frame nbr lo register*/
#define USB_DMACNTRL 0x14 /*DMA  control register*/
#define USB_EPC0 0x20 /*Endpoint0  register*/
#define USB_TXD0 0x21 /*TX data  register 0 */
#define USB_TXS0 0x22 /*TX status register 0*/
#define USB_TXC0 0x23 /*TX command register0*/
#define USB_RXD0 0x25 /*RX data register 0*/
#define USB_RXS0  0x26 /*RX status register 0*/
#define USB_RXC0 0x27 /*RX command register 0*/
#define USB_EPC1 0x28 /*Endpoint1  register */
#define USB_TXD1 0x29 /*TX data    register 1 */
#define USB_TXS1 0x2A /*TX status  register 1 */
#define USB_TXC1 0x2B /*TX command register 1 */
#define USB_EPC2 0x2C /*Endpoint2  register   */
#define USB_RXD1 0x2D /*RX data    register 1 */
#define USB_RXS1 0x2E /*RX status  register 1 */
#define USB_RXC1 0x2F /*RX command register 1 */
#define USB_EPC3 0x30 /*Endpoint3  register   */
#define USB_TXD2 0x31 /*TX data    register 2 */
#define USB_TXS2 0x32 /*TX status  register 2 */
#define USB_TXC2 0x33 /*TX command register 2 */
#define USB_EPC4 0x34 /*Endpoint4  register   */
#define USB_RXD2 0x35 /*RX data    register 2 */
#define USB_RXS2 0x36 /*RX status  register 2 */
#define USB_RXC2 0x37 /*RX command register 2 */
#define USB_EPC5 0x38 /*Endpoint5  register   */
#define USB_TXD3 0x39 /*TX data    register 3 */
#define USB_TXS3 0x3A /*TX status  register 3 */
#define USB_TXC3 0x3B /*TX command register 3 */
#define USB_EPC6 0x3C /*Endpoint6  register   */
#define USB_RXD3 0x3D /*RX data    register 3 */
#define USB_RXS3 0x3E /*RX status  register 3 */
#define USB_RXC3 0x3F /*RX command register 3 */

/*---------- MCNTRL bits -------------*/
#define USB_SRST 0x01 /*software reset */
#define USB_DBG 0x02 /*debug mode     */
#define USB_VGE 0x04 /*voltage regulator enable*/
#define USB_NAT 0x08 /*node attached  */
#define USB_INT_DIS 0x00 /*interrupts disabled    */
#define USB_INT_L_O 0x40 /*act lo ints, open drain*/
#define USB_INT_H_P 0x80 /*act hi ints, push pull */
#define USB_INT_L_P 0xC0 /*act lo ints, push pull */

/*------------ FAR bits --------------*/
#define USB_AD_EN 0x80 /*address enable */

/*------------ NFSR bits -------------*/
#define USB_RST_ST 0x00 /*reset       state */
#define USB_RSM_ST 0x01 /*resume      state */
#define USB_OPR_ST 0x02 /*operational state */
#define USB_SUS_ST 0x03 /*suspend     state */

/*-------- MAEV, MAMSK bits ----------*/
#define USB_WARN 0x01 /*warning bit has been set*/
#define USB_ALT 0x02 /*alternate event         */
#define USB_TX_EV 0x04 /*transmit event          */
#define USB_FRAME 0x08 /*SOF packet received     */
#define USB_NAK 0x10 /*NAK event               */
#define USB_ULD 0x20 /*unlock locked detected  */
#define USB_RX_EV 0x40 /*receive event          */
#define USB_INTR_E 0x80 /*master interrupt enable */

/*-------- ALTEV, ALTMSK bits --------*/
#define USB_EOP 0x08 /*end of packet */
#define USB_SD3 0x10 /*3 ms suspend  */
#define USB_SD5 0x20 /*5 ms suspend  */
#define USB_RESET_A  0x40 /*reset  detected */
#define USB_RESUME_A 0x80 /*resume detected */

/*--------- TXEV, TXMSK bits ---------*/
#define USB_TXFIFO0 0x01 /*TX_DONE, FIFO 0 */
#define USB_TXFIFO1 0x02 /*TX_DONE, FIFO 1 */
#define USB_TXFIFO2 0x04 /*TX_DONE, FIFO 2 */
#define USB_TXFIFO3 0x08 /*TX_DONE, FIFO 3 */
#define USB_TXUDRN0 0x10 /*TX_URUN, FIFO 0 */
#define USB_TXUDRN1 0x20 /*TX_URUN, FIFO 1 */
#define USB_TXUDRN2 0x40 /*TX_URUN, FIFO 2 */
#define USB_TXUDRN3 0x80 /*TX_URUN, FIFO 3 */


/*--------- RXEV, RXMSK bits ---------*/
#define USB_RXFIFO0 0x01 /*RX_DONE, FIFO 0 */
#define USB_RXFIFO1 0x02 /*RX_DONE, FIFO 1 */
#define USB_RXFIFO2 0x04 /*RX_DONE, FIFO 2 */
#define USB_RXFIFO3 0x08 /*RX_DONE, FIFO 3 */
#define USB_RXOVRN0 0x10 /*RX_OVRN, FIFO 0 */
#define USB_RXOVRN1 0x20 /*RX_OVRN, FIFO 1 */
#define USB_RXOVRN2 0x40 /*RX_OVRN, FIFO 2 */
#define USB_RXOVRN3 0x80 /*RX_OVRN, FIFO 3 */

/*------- NAKEV, NAKMSK bits ---------*/
#define USB_NAK_I0 0x01 /*IN  NAK, FIFO 0 */
#define USB_NAK_I1 0x02 /*IN  NAK, FIFO 1 */
#define USB_NAK_I2 0x04 /*IN  NAK, FIFO 2 */
#define USB_NAK_I3 0x08 /*IN  NAK, FIFO 3 */
#define USB_NAK_O0 0x10 /*OUT NAK, FIFO 0 */
#define USB_NAK_O1 0x20 /*OUT NAK, FIFO 1 */
#define USB_NAK_O2 0x40 /*OUT NAK, FIFO 2 */
#define USB_NAK_O3 0x80 /*OUT NAK, FIFO 3 */

/*--------- EPCX bits ----------------*/
#define USB_EP_EN 0x10 /*enables endpt.  (1-6)   */
#define USB_ISO 0x20 /*set for isochr. (1-6)   */
#define USB_DEF 0x40 /*force def. adr (0 only) */
#define USB_STALL 0x80 /*force stall handshakes  */

/*---------- TXCx bits ---------------*/
#define USB_TX_EN 0x01 /*transmit enable       */
#define USB_TX_LAST 0x02 /*last data in FIFO     */
#define USB_TX_TOGL 0x04 /*specifies PID used    */
#define USB_FLUSH 0x08 /*flushes all FIFO data */
#define USB_IGNIOS 0x80 /*       */

/*----------- TXSx bits --------------*/
#define USB_TX_DONE 0x20 /*transmit done         */
#define USB_ACK_STAT 0x40 /*ack status of xmission*/

/*------------ RXCx bits -------------*/
#define USB_RX_EN 0x01 /*receive  enable    */
#define USB_IGN_OUT 0x02 /*ignore out tokens  */
#define USB_IGN_SETUP 0x04 /*ignore setup tokens*/

/*------------- RXS0 bits ------------*/
#define USB_RX_LAST 0x10 /*indicates RCOUNT valid  */
```

```
#define USB_RX_TOGL 0x20 /*last pkt was DATA1 PID  */
#define USB_SETUP_RX 0x40 /*setup packet received   */
#define USB_RX_ERR 0x80 /*last packet had an error*/
```

## cute.c

```
#ifdef __cplusplus
extern "C" {
#endif
void abort(void);
#ifdef __cplusplus
}
#endif


#include <machine.h>
#include "iodefine.h"
#include "USB.C"

void send_SCI0(unsigned char data);
void send_SCI0_modC0(unsigned char data);
void send_SCI1(unsigned char data);
unsigned char receive_SCI0(void);
unsigned char receive_SCI1(void);
int write_buff2(char *p,int size);
void sci0_init(void);
void sci1_init(void);
void itu0_init(void);
void itu1_init(void);
void itu2_init(void);
void itu3_init(void);
void led_init(void);
void da1_init();
void P1_init(void);
void P2_init(void);
void P4_init(void);
void P5_init(void);
void counter_init(void);
void counter_ICclear(void);
void send_data(void);
void send_data_viaUSB(void);
void data_request(void);
void request_HK(void);
void request_CounterAa(void);
void request_CounterAb(void);
void request_CounterBa(void);
void request_CounterBb(void);
void request_All(void);
void request_All_viaUSB(void);
void request_AllwithGain(void);
void request_AllwithGain_viaUSB(void);
void hv_set(unsigned char data);
void gradual_hv_set_ON(unsigned char setValue);
void gradual_hv_set_OFF();
void gradual_hv_set_core(unsigned char setValue);
void gradual_hv_set_org(unsigned char setValue);
void wati_a_minite (int i);
void eth_set(unsigned char setValue);
unsigned char get_temperature();
void show_temperature();
void gain_set_30(void);
void gain_set_50(void);
void tmp_cont(void);
float ADvaluetoTemp(unsigned char advalue);
float TemptoHVno156and158Square(float temp,unsigned char gain);
float TemptoHVno156and158Cube(float temp,unsigned char gain);
float TemptoHV(float temp,float gain);
unsigned char HVtoADvalue(float hv);
float spl3252_gain_HV_m40 (float g);
float spl3252_gain_HV_m30 (float g);
float spl3252_gain_HV_m20 (float g);
float spl3252_gain_HV_m10 (float g);
float spl3252_gain_HV_0 (float g);
float spl3252_gain_HV_10 (float g);
float spl3252_gain_HV_20 (float g);
float spl3252_gain_HV_30 (float g);
float spl3252_gain_HV_40 (float g);
void tmp_cont_on(void);
void tmp_cont_off(void);
void apd_on_a(void);
void apd_on_b(void);
unsigned int value_of_counterA(void);
unsigned int value_of_counterB(void);
void reset_counterA(void);
void reset_counterB(void);
void reset_counterAll(void);

void USB_init(void);
void H8init(void);
void toUSBofficeAdministrator (char buff[64]);

void PA_init(void);
void CheckPort(void);

void visual_scaler_START(void);
void visual_scaler_STOP(void);
void visual_scaler_RESET(void);
void KEK_send_data(void);
void KEK_reset(void);
void KEK_start(void);

void tmp_log(void);
void tmp_log_on(void);
void tmp_log_off(void);

unsigned int i;
unsigned int iTemp;
unsigned int iHV;
unsigned char tmp_cont_isOn;
unsigned char counter1_IC;
unsigned char counter2_IC;
unsigned int counter1_ovf;
unsigned int counter2_ovf;

unsigned char hv_value_atnow;
unsigned char temperature_atnow;
unsigned char threshold_atnow;
unsigned char selAPD_atnow;
unsigned char counter1_IC_atnow;
unsigned int counter1_atnow;
unsigned int counter1_ovf_atnow;
unsigned char counter2_IC_atnow;
unsigned int counter2_atnow;
unsigned int counter2_ovf_atnow;
unsigned char gain_atnow;

float volMax = 396.5;
float AVref = 2.973;
unsigned char expected_hvValue;
unsigned char gradual_hv_set_isOn;

unsigned char tmp_log_isON;

void main(void)
{
  int cnt;
  static char buff[64];

  gain_atnow = 30;
  H8init();
  sci0_init();
  itu0_init();
  itu3_init();
  led_init();
  da1_init();
```

```
  P5_init();
  P2_init();
  counter_init();
  USB_init();

  while (1) {
    if( get_inbufflen() )
      {
cnt = read_buff(buff,64);
toUSBofficeAdministrator(buff);
      }
    ;
  }
}

void send_SCI0(unsigned char data)
{
  while (SCI0.SSR.BIT.TDRE==0) {} //TDR
  SCI0.TDR = data;
  SCI0.SSR.BIT.TDRE = 0; //tONAAMJn
}

void send_SCI0_modC0(unsigned char data)
{
  if (0xc0==data) {
    send_SCI0(0xdb);send_SCI0(0xdc);
  } else if (0xdb==data) {
    send_SCI0(0xdb);send_SCI0(0xdd);
  } else {
    send_SCI0(data);
  }
}

void send_SCI1(unsigned char data)
{
  while (SCI1.SSR.BIT.TDRE==0) {} //TDR
  SCI1.TDR = data;
  SCI1.SSR.BIT.TDRE = 0; //tONAAMJn
}

unsigned char receive_SCI0(void)
{
  unsigned char data;
  while (SCI0.SSR.BIT.RDRF == 0) {}//RDRf[^
  data = SCI0.RDR;//Mf[^WX^dataRs[
  SCI0.SSR.BIT.RDRF = 0;//MIARDRNA
  return data;
}

unsigned char receive_SCI1(void)
{
  unsigned char data;
  while (SCI1.SSR.BIT.RDRF == 0) {}//RDRf[^
  data = SCI1.RDR;//Mf[^WX^dataRs[
  SCI1.SSR.BIT.RDRF = 0;//MIARDRNA
  return data;
}

int write_buff2(char *p,int size)
{
  int i;
  i = write_buff(p,size);
  SendTX1();
  return(i);
}

void sci0_init(void)
{
  int i;
  SCI0.SCR.BYTE = 0x00;
  SCI0.SMR.BYTE = 0x00;
  SCI0.BRR = 51; //9600bps
  for (i=0;i<1000;i++) {}
  SCI0.SCR.BYTE = 0x70; //Tx,RxLAML
  SCI0.SSR.BYTE &= 0x80; // G[tONA
}

void sci1_init(void)
{
  int i;
  SCI1.SCR.BYTE = 0x00;
  SCI1.SMR.BYTE = 0x00;
  SCI1.BRR = 51; //9600bps
  for (i=0;i<1000;i++) {}
  SCI1.SCR.BYTE = 0x70; //Tx,RxLAML
  SCI1.SSR.BYTE &= 0x80; // G[tONA
}

void itu0_init(void)
{
  ITU0.TCR.BYTE = 0xa3; //GRA compare match, phi/8
  ITU0.GRA = 50000; // 25ms
  ITU0.TIER.BIT.IMIEA = 1;//IMFA
  ITU.TSTR.BIT.STR0 = 1; //TCNT0JE^ON
}

void itu1_init(void)
{
  counter1_ovf = 0;
  ITU1.TCR.BYTE = 0xac; //GRA compare match, TCLKAJEg
  ITU1.GRA = 0xffff; //
  ITU1.TIER.BIT.IMIEA = 1;//IMFA
  ITU1.TIER.BIT.OVIE = 1; //OVF
  ITU.TSTR.BIT.STR1 = 1; //TCNT1JE^ON
}

void itu2_init(void)
{
  counter2_ovf = 0;
  ITU2.TCR.BYTE = 0xad; //GRA compare match, TCLKBJEg
  ITU2.GRA = 0xffff; //
  ITU2.TIER.BIT.IMIEA = 1;//IMFA
  ITU2.TIER.BIT.OVIE = 1; //OVF
  ITU.TSTR.BIT.STR2 = 1; //TCNT2JE^ON
}

void itu3_init(void)
{
  ITU3.TCR.BYTE = 0xa3; //GRA compare match, phi/8
  ITU3.GRA = 50000; // 25ms
  ITU3.TIER.BIT.IMIEA = 1;//IMFA
  ITU.TSTR.BIT.STR3 = 1; //TCNT3JE^ON
}

void led_init(void)
{
  PB.DDR = 0xff; // ledopJ
}

void da1_init(void)
{
  DA.DACR.BYTE = 0x9f;//‘lPDA
}

void P1_init(void)
{
  P1.DDR = 0xff; // P1opJ
}

void P2_init(void)
{
  P2.DDR = 0x00; // P2pJ
}

void P4_init(void)
{
  P4.DDR = 0xff; // P4opJ
}

void P5_init(void)
```

```
{
  P5.DDR = 0xff; // P5opJ
}

void PA_init(void)
{
  PA.DDR = 0x00; // PApJ
}

void counter_init(void)
{
  P2_init();
  P4_init();
  PB.DR.BIT.B3 = 1; // RSTA become High
  PB.DR.BIT.B4 = 1; // RSTB become High
  itu1_init();
  itu2_init();
}

void counter_ICclear(void)
{
  PB.DR.BIT.B3 = 0; // RSTA become High
  PB.DR.BIT.B4 = 0; // RSTB become High
}

void send_data(void)
{
  int countA, countB;
  countA = value_of_counterA();
  countB = value_of_counterB();

  send_SCI0_modC0(counter1_ovf>>8);
  send_SCI0_modC0(counter1_ovf);
  send_SCI0_modC0(countA>>8);
  send_SCI0_modC0(countA);
  send_SCI0_modC0(counter1_IC);

  send_SCI0_modC0(counter2_ovf>>8);
  send_SCI0_modC0(counter2_ovf);
  send_SCI0_modC0(countB>>8);
  send_SCI0_modC0(countB);
  send_SCI0_modC0(counter2_IC);
}

void send_data_viaUSB(void)
{
  int countA, countB;
  unsigned char buff[10];

  countA = value_of_counterA();
  countB = value_of_counterB();

  buff[0] = counter1_ovf>>8;
  buff[1] = counter1_ovf;
  buff[2] = countA>>8;
  buff[3] = countA;
  buff[4] = counter1_IC;

  buff[5] = counter2_ovf>>8;
  buff[6] = counter2_ovf;
  buff[7] = countB>>8;
  buff[8] = countB;
  buff[9] = counter2_IC;

  write_buff2(buff,64); /* USBo */

}

void data_request(void)
{
  get_temperature();
  counter1_atnow = value_of_counterA();
  counter1_IC_atnow = counter1_IC;
  counter1_ovf_atnow = counter1_ovf;
  counter2_atnow = value_of_counterB();
  counter2_IC_atnow = counter2_IC;
  counter2_ovf_atnow = counter2_ovf;

  send_SCI0(0xc0);send_SCI0(0x03);
  send_SCI0(1);
  send_SCI0(0xc0);
}

void request_HK(void)
{
  unsigned char lastTerm
    = tmp_cont_isOn + 2*threshold_atnow + 8*selAPD_atnow;
  send_SCI0(0xc0);send_SCI0(0x03);
  send_SCI0(1);
  send_SCI0_modC0(temperature_atnow);
  send_SCI0_modC0(hv_value_atnow);
  send_SCI0_modC0(lastTerm);
  send_SCI0(0xc0);
}

void request_HK_viaUSB(void)
{
  unsigned char buff[4];
  char lastTerm
    = tmp_cont_isOn + 2*threshold_atnow + 8*selAPD_atnow;
  buff[0] = 1;
  buff[1] = temperature_atnow;
  buff[2] = hv_value_atnow;
  buff[3] = lastTerm;
  write_buff2(buff,64); /* USBo */
}

void request_CounterAa(void)
{
  send_SCI0(0xc0);send_SCI0(0x03);
  send_SCI0(1);
  send_SCI0_modC0(counter1_ovf_atnow>>8);
  send_SCI0_modC0(counter1_ovf_atnow);
  send_SCI0_modC0(counter1_atnow>>8);
  send_SCI0(0xc0);
}

void request_CounterAa_viaUSB(void)
{
  unsigned char buff[4];
  buff[0] = 1;
  buff[1] = counter1_ovf_atnow>>8;
  buff[2] = counter1_ovf_atnow;
  buff[3] = counter1_atnow>>8;
  write_buff2(buff,64); /* USBo */

}

void request_CounterAb(void)
{
  send_SCI0(0xc0);send_SCI0(0x03);
  send_SCI0(1);
  send_SCI0_modC0(counter1_atnow);
  send_SCI0_modC0(counter1_IC_atnow);
  send_SCI0(0xc0);
}

void request_CounterAb_viaUSB(void)
{
  unsigned char buff[3];
  buff[0] = 1;
  buff[1] = counter1_atnow;
  buff[2] = counter1_IC_atnow;
  write_buff2(buff,64); /* USBo */
}

void request_CounterBa(void)
{
  send_SCI0(0xc0);send_SCI0(0x03);
```

```
  send_SCI0(1);
  send_SCI0_modC0(counter2_ovf_atnow>>8);
  send_SCI0_modC0(counter2_ovf_atnow);
  send_SCI0_modC0(counter2_atnow>>8);
  send_SCI0(0xc0);
}

void request_CounterBa_viaUSB(void)
{
  unsigned char buff[4];
  buff[0] = 1;
  buff[1] = counter2_ovf_atnow>>8;
  buff[2] = counter2_ovf_atnow;
  buff[3] = counter2_atnow>>8;
  write_buff2(buff,64); /* USBo */
}

void request_CounterBb(void)
{
  send_SCI0(0xc0);send_SCI0(0x03);
  send_SCI0(1);
  send_SCI0_modC0(counter2_atnow);
  send_SCI0_modC0(counter2_IC_atnow);
  send_SCI0(0xc0);
}

void request_CounterBb_viaUSB(void)
{
  unsigned char buff[3];
  buff[0] = 1;
  buff[1] = counter2_atnow;
  buff[2] = counter2_IC_atnow;
  write_buff2(buff,64); /* USBo */
}

void request_All(void)
{
  get_temperature();

  send_SCI0(0xc0);send_SCI0(0x03);
  send_SCI0_modC0(1);
  send_SCI0_modC0(temperature_atnow);
  send_SCI0_modC0(hv_value_atnow);
  send_SCI0_modC0(tmp_cont_isOn);
  send_SCI0_modC0(threshold_atnow);
  send_SCI0_modC0(selAPD_atnow);
  send_data();
  send_SCI0(0xc0);
}

void request_All_viaUSB(void)
{
  unsigned char buff[16];
  data_request();
  buff[0]  = 1;
  buff[1]  = temperature_atnow;
  buff[2]  = hv_value_atnow;
  buff[3]  = tmp_cont_isOn;
  buff[4]  = threshold_atnow;
  buff[5]  = selAPD_atnow;
  buff[6]  = counter1_ovf_atnow>>8;
  buff[7]  = counter1_ovf_atnow;
  buff[8]  = counter1_atnow>>8;
  buff[9]  = counter1_atnow;
  buff[10] = counter1_IC_atnow;
  buff[11] = counter2_ovf_atnow>>8;
  buff[12] = counter2_ovf_atnow;
  buff[13] = counter2_atnow>>8;
  buff[14] = counter2_atnow;
  buff[15] = counter2_IC_atnow;
  write_buff2(buff,64); /* USBo */
}

void request_AllwithGain(void)
{
  get_temperature();

  send_SCI0(0xc0);send_SCI0(0x03);
  send_SCI0_modC0(1);
  send_SCI0_modC0(temperature_atnow);
  send_SCI0_modC0(hv_value_atnow);
  send_SCI0_modC0(tmp_cont_isOn);
  send_SCI0_modC0(threshold_atnow);
  send_SCI0_modC0(selAPD_atnow);
  send_data();
  send_SCI0_modC0(gain_atnow);
  send_SCI0(0xc0);
}

void request_AllwithGain_viaUSB(void)
{
  unsigned char buff[17];
  data_request();
  buff[0]  = 1;
  buff[1]  = temperature_atnow;
  buff[2]  = hv_value_atnow;
  buff[3]  = tmp_cont_isOn;
  buff[4]  = threshold_atnow;
  buff[5]  = selAPD_atnow;
  buff[6]  = counter1_ovf_atnow>>8;
  buff[7]  = counter1_ovf_atnow;
  buff[8]  = counter1_atnow>>8;
  buff[9]  = counter1_atnow;
  buff[10] = counter1_IC_atnow;
  buff[11] = counter2_ovf_atnow>>8;
  buff[12] = counter2_ovf_atnow;
  buff[13] = counter2_atnow>>8;
  buff[14] = counter2_atnow;
  buff[15] = counter2_IC_atnow;
  buff[16] = gain_atnow;
  write_buff2(buff,64); /* USBo */
}

void hv_set(unsigned char data)
{
  DA.DADR1 = data; //HV15V*data/256
  hv_value_atnow = data;
  send_SCI0(0xc0);send_SCI0(0x03);
  send_SCI0(1);
  send_SCI0_modC0(data);
  send_SCI0(0xc0);
}

void gradual_hv_set_ON(unsigned char setValue) {
  gradual_hv_set_isOn = 1;
  expected_hvValue = setValue;
  send_SCI0(0xc0);send_SCI0(0x03);send_SCI0(1);send_SCI0(0xc0);
}

void gradual_hv_set_OFF() {
  gradual_hv_set_isOn = 0;
  send_SCI0(0xc0);send_SCI0(0x03);send_SCI0(1);send_SCI0(0xc0);
}

void gradual_hv_set_core(unsigned char setValue)
{
  unsigned char i;
  unsigned char hvValue;
  send_SCI0(hv_value_atnow);
  hvValue = hv_value_atnow;
  if (setValue>hvValue) {
    if (0xff!=hvValue) hv_set(hvValue+1);
  } else if (setValue==hvValue) {
    gradual_hv_set_OFF();
  } else {
    if (0x00!=hvValue) hv_set(hvValue-1);
  }
}

void gradual_hv_set_org(unsigned char setValue)
```

```
{
  unsigned char i;
  unsigned char hvValue;
send_SCI0(hv_value_atnow);
hvValue = hv_value_atnow;
if (setValue>hvValue) {
  for (i=hvValue;i<=setValue;i++) {
    hv_set(i);
    wati_a_minite(40);
    if (i==0xff) break;
  }
} else {
  for (i=hvValue;i>=setValue;i--) {
    hv_set(i);
    wati_a_minite(40);
    if (i==0x00) break;
  }
}
}

void wati_a_minite (int i)
{
  int j,k;
  for (j=0;j<i;j++) {
    for (k=0;k<0xffff;k++) {}
  }
}

void eth_set(unsigned char setValue)
{
  send_SCI0(0xc0);send_SCI0(0x03);
  send_SCI0(1);send_SCI0(setValue);//for debug
  send_SCI0(0xc0);

  if  (0x30==setValue) {
    PB.DR.BIT.B0 = 0;
    PB.DR.BIT.B1 = 0;
    threshold_atnow = 0x30;
  } else if (0x31==setValue) {
    PB.DR.BIT.B0 = 1;
    PB.DR.BIT.B1 = 0;
    threshold_atnow = 0x31;
  } else if (0x32==setValue) {
    PB.DR.BIT.B0 = 0;
    PB.DR.BIT.B1 = 1;
    threshold_atnow = 0x32;
  } else if (0x33==setValue) {
    PB.DR.BIT.B0 = 1;
    PB.DR.BIT.B1 = 1;
    threshold_atnow = 0x33;
  }
}

unsigned char get_temperature()
{
  unsigned char advalue;
  AD.ADCSR.BYTE = 0x28; //AN0 P[h
  while (AD.ADCSR.BIT.ADF==0) {}//ADI
  advalue = AD.ADDRA >> 8;// ADMadvaluei[
  temperature_atnow = advalue;
  return advalue;
}

void show_temperature()
{
  send_SCI0(0xc0);send_SCI0(0x03);
  send_SCI0_modC0(get_temperature());
  send_SCI0(0xc0);
}

void gain_set_30()
{
  gain_atnow = 30;
  send_SCI0(0xc0);send_SCI0(0x03);send_SCI0(1);send_SCI0(0xc0);
}

void gain_set_50()
{
  gain_atnow = 50;
  send_SCI0(0xc0);send_SCI0(0x03);send_SCI0(1);send_SCI0(0xc0);
}

void tmp_cont(void)
{
  unsigned char advalue;
  unsigned char set_value;
  float hv,temp;
  unsigned char gain;

  AD.ADCSR.BYTE = 0x28; //AN0 P[h
  while (AD.ADCSR.BIT.ADF==0) {}//ADI
  advalue = AD.ADDRA >> 8;// ADMadvaluei[

  gain = gain_atnow; //QCX\
  send_SCI0(advalue);
  temp = ADvaluetoTemp(advalue);
  //hv = TemptoHV(temp,gain);
  if (0==selAPD_atnow) {
    hv = TemptoHVno156and158Cube(temp,gain);
  } else if (1==selAPD_atnow) {
    hv = TemptoHVno156and158Cube(temp,gain);
  } else {
  }
  send_SCI0(hv); //for debug
  if (hv>volMax) {
    set_value = 255;
  } else {
    set_value = HVtoADvalue(hv);
  }
  send_SCI0(set_value); //for debug
  gradual_hv_set_ON(set_value);
}

float ADvaluetoTemp(unsigned char advalue) {
  float vcc, voltage, temperature;
  voltage = AVref*advalue/255.;
  temperature = 26.64*voltage -50.06;
  return temperature;
}

float TemptoHVno156and158Square(float temp,unsigned char gain) {
  float a,b,c;
  if (30==gain) {
    a = 0.000538331030088785;
    b = 0.822084459400044;
    c = 358.574658065267;
  } else if (50==gain) {
    a = 0.00129006950996322;
    b = 0.855832896884502;
    c = 381.75799253569;
  }
  return (a*temp*temp + b*temp + c);
}

float TemptoHVno156and158Cube(float temp,unsigned char gain) {
  float a,b,c,d;
  if (30==gain) {
    a = 6.32991458881554e-05;
    b = 0.00221138086702532;
    c = 0.793148134673951;
    d = 358.219729722205;
  } else if (50==gain) {
    a = 0.000106110429459764;
    b = 0.00416349348841085;
    c = 0.813957164388571;
    d = 381.346230494575;
  }
  return (a*temp*temp*temp + b*temp*temp + c*temp + d);
}
```

```
float TemptoHV(float temp,float gain) {
  float facA,facB;
  if (40<=temp) {
  } else if (30<=temp && temp <40) {
    facB = (temp-30)/10.;
    facA = (40-temp)/10.;
    return
      (facA*spl3252_gain_HV_30(gain) + facB*spl3252_gain_HV_40(gain));
  } else if (20<=temp && temp <30) {
    facB = (temp-20)/10.;
    facA = (30-temp)/10.;
    return
      (facA*spl3252_gain_HV_20(gain) + facB*spl3252_gain_HV_30(gain));
  } else if (10<=temp && temp <20) {
    facB = (temp-10)/10.;
    facA = (20-temp)/10.;
    return
      (facA*spl3252_gain_HV_10(gain) + facB*spl3252_gain_HV_20(gain));
  } else if (0<=temp && temp <10) {
    facB = (temp-0)/10.;
    facA = (10-temp)/10.;
    return
      (facA*spl3252_gain_HV_0(gain) + facB*spl3252_gain_HV_10(gain));
  } else if (-10<=temp && temp <0) {
    facB = (temp-(-10))/10.;
    facA = (0-temp)/10.;
    return
      (facA*spl3252_gain_HV_m10(gain) + facB*spl3252_gain_HV_0(gain));
  } else if (-20<=temp && temp <-10) {
    facB = (temp-(-20))/10.;
    facA = (-10-temp)/10.;
    return
      (facA*spl3252_gain_HV_m20(gain) + facB*spl3252_gain_HV_m10(gain));
  } else if (-30<=temp && temp <-20) {
    facB = (temp-(-30))/10.;
    facA = (-20-temp)/10.;
    return
      (facA*spl3252_gain_HV_m30(gain) + facB*spl3252_gain_HV_m20(gain));
  } else if (-40<=temp && temp <-30) {
    facB = (temp-(-40))/10.;
    facA = (-30-temp)/10.;
    return
      (facA*spl3252_gain_HV_m40(gain) + facB*spl3252_gain_HV_m30(gain));
  } else if (temp < -40) {
  } else {
  }
}

unsigned char HVtoADvalue(float hv) {
  return (255.*hv/volMax);
}

float spl3252_gain_HV_m40 (float g) {
  return (10.3946*g*g*g -93.5907*g*g +322.044*g -130.031);
}

float spl3252_gain_HV_m30 (float g) {
  return (20.1367*g*g*g -150.939*g*g +432.398*g -163.054);
}

float spl3252_gain_HV_m20 (float g) {
  return (16.3257*g*g*g -136.618*g*g +421.263*g -129.814);
}

float spl3252_gain_HV_m10 (float g) {
  return (40.0891*g*g*g -256.881*g*g +613.734*g -191.697);
}

float spl3252_gain_HV_0 (float g) {
  return (72.4753*g*g*g -388.269*g*g +785.809*g -229.786);
}

float spl3252_gain_HV_10 (float g) {
  return (110.885*g*g*g -511.824*g*g +915.251*g -239.177);
}

float spl3252_gain_HV_20 (float g) {
  return (252.616*g*g*g -912.847*g*g +1289.92*g -317.87);
}

float spl3252_gain_HV_30 (float g) {
  return (195.301*g*g*g -829.402*g*g +1287.4*g -296.657);
}

float spl3252_gain_HV_40 (float g) {
  return (250.777*g*g*g -980.164*g*g +1440.53*g -302.337);
}

void tmp_cont_on(void) {
  send_SCI0(0xff);//for debug
  tmp_cont_isOn = 1;
  send_SCI0(0xc0);send_SCI0(0x03);send_SCI0(1);send_SCI0(0xc0);
}

void tmp_cont_off(void)
{
  send_SCI0(0x00);//for debug
  tmp_cont_isOn = 0;
  send_SCI0(0xc0);send_SCI0(0x03);send_SCI0(1);send_SCI0(0xc0);
}

void apd_on_a(void)
{
  send_SCI0(0x0a);//for debug
  PB.DR.BIT.B2 = 0;
  selAPD_atnow = 0;
  send_SCI0(0xc0);send_SCI0(0x03);send_SCI0(1);send_SCI0(0xc0);
}

void apd_on_b(void)
{
  send_SCI0(0x0b);//for debug
  PB.DR.BIT.B2 = 1;
  selAPD_atnow = 1;
  send_SCI0(0xc0);send_SCI0(0x03);send_SCI0(1);send_SCI0(0xc0);
}

unsigned int value_of_counterA(void)
{
  unsigned int countVal;
  unsigned int upperVal;
  unsigned char lowerVal;
  unsigned char q1,q2,q3,q4;
  q1 = P8.DR.BIT.B0;
  q2 = P8.DR.BIT.B1;
  q3 = P8.DR.BIT.B3;
  q4 = P8.DR.BIT.B4;
  upperVal = ITU1.TCNT;
  lowerVal = q1 + 2*q2 + 4*q3 + 8*q4;
  countVal = upperVal;
  counter1_IC = lowerVal;
  return countVal;
}

unsigned int value_of_counterB(void)
{
  unsigned int countVal;
  unsigned int upperVal;
  unsigned char lowerVal;
  unsigned char q1,q2,q3,q4;
  upperVal = ITU2.TCNT;
  q1 = PA.DR.BIT.B2;
  q2 = PA.DR.BIT.B3;
  q3 = PA.DR.BIT.B4;
  q4 = PA.DR.BIT.B5;
  lowerVal = q1 + 2*q2 + 4*q3 + 8*q4;
  countVal = upperVal;
  counter2_IC = lowerVal;
  return countVal;
}
```

```
void reset_counterA(void)
{
  send_SCI0(0x00);//for debug
  PB.DR.BIT.B3 = 0; // RSTA become Low
  wati_a_minite(1);
  PB.DR.BIT.B3 = 1; // RSTA become High
  counter1_IC = 0;
  ITU1.TCNT = 0;
  counter1_ovf = 0;

  send_SCI0(0xc0);send_SCI0(0x03);
  send_SCI0(1);
  send_SCI0(0xc0);
}

void reset_counterB(void)
{
  send_SCI0(0x00);//for debug
  PB.DR.BIT.B4 = 0; // RSTB become Low
  wati_a_minite(1);
  PB.DR.BIT.B4 = 1; // RSTB become High
  counter2_IC = 0;
  ITU2.TCNT = 0;
  counter2_ovf = 0;

  send_SCI0(0xc0);send_SCI0(0x03);
  send_SCI0(1);
  send_SCI0(0xc0);
}

void reset_counterAll(void)
{
  reset_counterA();
  reset_counterB();
}

void USB_init(void)
{
  InitUSB();
  INTC.ISCR.BYTE &= (-1^0x20);
  INTC.IER.BYTE |= 0x20;
  set_imask_ccr(0);
}

void H8init(void)
{
  BSC.ABWCR.BYTE = 0x06; /* 8bit BUS MODE */
  P1.DDR = 0xff;         /* all OUT*/
  P6.DDR = 0xff; /* all OUT*/
  P8.DDR = 0x04;          /* IN except P82 */
  PB.DDR = 0xff;          /* all OUT*/
  P5.DDR = 0xff;
  AD.ADCSR.BYTE = 0x33;  /*auto scan mode*/
  PA.DDR = 0x00;
}

void toUSBofficeAdministrator (char buff[64])
{
  unsigned char data;
  unsigned char redundantCommand;
  unsigned char value;
  char answer[64];
  int k;
for (k=0;k<64;k++) {
  answer[k] = 0;
}
answer[0] = 1;

data = buff[0];

if (data==0x99) {
PB.DR.BYTE = 0x00; //LED all ON
write_buff2(answer,64);/* USBo */
} else if (data==0x31) {
PB.DR.BYTE = 0xff; //LED all OFF
send_SCI0(1);
write_buff2(answer,64);/* USBo */
} else if (0x32==data) {
redundantCommand = buff[1];
if (0x32==redundantCommand) {
value = buff[2];
hv_set(value);
write_buff2(answer,64);/* USBo */
}
} else if (0x33==data) {
redundantCommand = buff[1];
if (0x33==redundantCommand) {
value = buff[2];
gradual_hv_set_ON(value);
write_buff2(answer,64);/* USBo */
}
} else if (0x34==data) {
tmp_cont_on();
write_buff2(answer,64);/* USBo */
} else if (0x35==data) {
tmp_cont_off();
write_buff2(answer,64);/* USBo */
} else if (0x36==data) {
apd_on_a();
write_buff2(answer,64);/* USBo */
} else if (0x37==data) {
apd_on_b();
write_buff2(answer,64);/* USBo */
} else if (0x38==data) {
value = buff[1];
eth_set(value);
write_buff2(answer,64);/* USBo */
} else if (0x39==data) {
reset_counterA();
write_buff2(answer,64);/* USBo */
} else if (0x3a==data) {
reset_counterB();
write_buff2(answer,64);/* USBo */
} else if (0x3b==data) {
reset_counterAll();
write_buff2(answer,64);/* USBo */
} else if (0x40==data) {
data_request();
} else if (0x41==data) {
request_HK();
request_HK_viaUSB();
} else if (0x42==data) {
request_CounterAa();
request_CounterAa_viaUSB();
} else if (0x43==data) {
request_CounterAb();
request_CounterAb_viaUSB();
} else if (0x44==data) {
request_CounterBa();
request_CounterBa_viaUSB();
} else if (0x45==data) {
request_CounterBb();
request_CounterBb_viaUSB();
} else if (0x46==data) {
gain_set_30();
write_buff2(answer,64);/* USBo */
} else if (0x47==data) {
gain_set_50();
write_buff2(answer,64);/* USBo */
} else if (0x50==data) {
request_All();
request_All_viaUSB();
} else if (0x51==data) {
request_AllwithGain();
request_AllwithGain_viaUSB();
} else if (0x70==data) {
send_SCI0(0xc0);send_SCI0(0x03);
send_data();
send_SCI0(0xc0);
```

```
send_data_viaUSB();
write_buff2(answer,64);/* USBo */
} else if (0x90==data) {
gradual_hv_set_OFF();
write_buff2(answer,64);/* USBo */
} else if (0xa0==data) {
tmp_log_on();
write_buff(answer,64);/* USBo */
} else if (0xa1==data) {
tmp_log_off();
write_buff(answer,64);/* USBo */
}

}


void CheckPort(void) {
char q1,q2,q3,q4,q5,q6,q7,q8;
q1 = P8.DR.BIT.B0;
q2 = P8.DR.BIT.B1;
q3 = P8.DR.BIT.B3;
q4 = P8.DR.BIT.B4;
q5 = PA.DR.BIT.B2;
q6 = PA.DR.BIT.B3;
q7 = PA.DR.BIT.B4;
q8 = PA.DR.BIT.B5;
send_SCI0(q1);
send_SCI0(q2);
send_SCI0(q3);
send_SCI0(q4);
send_SCI0(q5);
send_SCI0(q6);
send_SCI0(q7);
send_SCI0(q8);
}


void visual_scaler_START(void) {
PB.DR.BIT.B5 = 0;
PB.DR.BIT.B5 = 1;
PB.DR.BIT.B5 = 0;
send_SCI0(1);
}

void visual_scaler_STOP(void) {
PB.DR.BIT.B6 = 0;
PB.DR.BIT.B6 = 1;
PB.DR.BIT.B6 = 0;
send_SCI0(1);
}

void visual_scaler_RESET(void) {
PB.DR.BIT.B7 = 0;
PB.DR.BIT.B7 = 1;
PB.DR.BIT.B7 = 0;
send_SCI0(1);
}

void KEK_send_data(void) {
send_data();
visual_scaler_STOP();
}

void KEK_reset(void) {
reset_counterAll();
visual_scaler_RESET();
}

void KEK_start(void) {
reset_counterAll();
visual_scaler_START();
}

void tmp_log(void) {
request_All();
request_All_viaUSB();
}

void tmp_log_on(void) {
tmp_log_isON = 1;
}

void tmp_log_off(void) {
tmp_log_isON = 0;
}

void abort(void)
{
send_SCI0(0);
}
```

## intprg.c

```
extern unsigned int i;
extern unsigned int iTemp;
extern unsigned int iHV;
extern unsigned char tmp_cont_isOn;
extern unsigned char gradual_hv_set_isOn;
extern char expected_hvValue;
extern unsigned int counter1_ovf;
extern unsigned int counter2_ovf;
extern unsigned char tmp_log_isON;

#include "iodefine.h"
#include    <machine.h>
#pragma section IntPRG

__interrupt(vect=7) void INT_NMI(void) {/* sleep(); */}
__interrupt(vect=8) void INT_TRAP1(void) {/* sleep(); */}
__interrupt(vect=9) void INT_TRAP2(void) {/* sleep(); */}
__interrupt(vect=10) void INT_TRAP3(void) {/* sleep(); */}
__interrupt(vect=11) void INT_TRAP4(void) {/* sleep(); */}
__interrupt(vect=12) void INT_IRQ0(void) {/* sleep(); */}
__interrupt(vect=13) void INT_IRQ1(void) {/* sleep(); */}
__interrupt(vect=14) void INT_IRQ2(void) {/* sleep(); */}
__interrupt(vect=15) void INT_IRQ3(void) {/* sleep(); */}
__interrupt(vect=16) void INT_IRQ4(void) {/* sleep(); */}
__interrupt(vect=17) void INT_IRQ5(void) {
usb_int();
}

__interrupt(vect=20) void INT_WOVI(void) {/* sleep(); */}
__interrupt(vect=21) void INT_CMI(void) {/* sleep(); */}
__interrupt(vect=24) void INT_IMIA0(void) {
if (0 == (iTemp%50)) {
if (1==tmp_cont_isOn) tmp_cont();
if (1==tmp_log_isON) tmp_log();
}
ITU0.TSR.BIT.IMFA = 0;//IMFAtONA
iTemp++;
}
__interrupt(vect=25) void INT_IMIB0(void) {/* sleep(); */}
__interrupt(vect=26) void INT_OVI0(void) {
; //do nothing. OVF interution is not permitted.
}

__interrupt(vect=28) void INT_IMIA1(void) {
ITU1.TSR.BIT.IMFA = 0;//IMFAtONA
}
__interrupt(vect=29) void INT_IMIB1(void) {/* sleep(); */}
__interrupt(vect=30) void INT_OVI1(void) {
ITU1.TSR.BIT.OVF = 0;//OVFtONA
counter1_ovf++;
}
```

```
__interrupt(vect=32) void INT_IMIA2(void) {
ITU2.TSR.BIT.IMFA = 0;//IMFAtONA
}
//  vector 33 IMIB2
__interrupt(vect=33) void INT_IMIB2(void) {/* sleep(); */}
//  vector 34 OVI2
__interrupt(vect=34) void INT_OVI2(void) {
ITU2.TSR.BIT.OVF = 0;//OVFtONA
counter2_ovf++;
}
__interrupt(vect=36) void INT_IMIA3(void) {
  if (0 == (iHV%100)) {
    if (1==gradual_hv_set_isOn)
      gradual_hv_set_core(expected_hvValue);
  }
  ITU3.TSR.BIT.IMFA = 0;//IMFAtONA
  iHV++;
}
__interrupt(vect=37) void INT_IMIB3(void) {/* sleep(); */}
__interrupt(vect=38) void INT_OVI3(void) {/* sleep(); */}
__interrupt(vect=40) void INT_IMIA4(void) {/* sleep(); */}
__interrupt(vect=41) void INT_IMIB4(void) {/* sleep(); */}
__interrupt(vect=42) void INT_OVI4(void) {/* sleep(); */}
__interrupt(vect=44) void INT_DEND0A(void) {/* sleep(); */}
__interrupt(vect=45) void INT_DEND0B(void) {/* sleep(); */}
__interrupt(vect=46) void INT_DEND1A(void) {/* sleep(); */}
__interrupt(vect=47) void INT_DEND1B(void) {/* sleep(); */}
__interrupt(vect=52) void INT_ERI0(void) {
SCI0.SSR.BYTE &= 0xc7; // clear error flag
}
__interrupt(vect=53) void INT_RXI0(void) {
unsigned char data;
unsigned char redundantCommand;
unsigned char value;

data = receive_SCI0();
if (data==0x30) {
PB.DR.BYTE = 0x00; //LED all ON
} else if (data==0x31) {
PB.DR.BYTE = 0xff; //LED all OFF
} else if (0x32==data) {
redundantCommand = receive_SCI0();
if (0x32==redundantCommand) {
value = receive_SCI0();
hv_set(value);
} else {
value = receive_SCI0();
}
} else if (0x33==data) {
redundantCommand = receive_SCI0();
if (0x33==redundantCommand) {
value = receive_SCI0();
gradual_hv_set_ON(value);
} else {
value = receive_SCI0();
}
} else if (0x34==data) {
tmp_cont_on();
} else if (0x35==data) {
tmp_cont_off();
} else if (0x36==data) {
apd_on_a();
} else if (0x37==data) {
apd_on_b();
} else if (0x38==data) {
value = receive_SCI0();
eth_set(value);
} else if (0x39==data) {
reset_counterA();
} else if (0x3a==data) {
reset_counterB();
} else if (0x3b==data) {
reset_counterAll();
} else if (0x40==data) {
data_request();
} else if (0x41==data) {
request_HK();
} else if (0x42==data) {
request_CounterAa();
} else if (0x43==data) {
request_CounterAb();
} else if (0x44==data) {
request_CounterBa();
} else if (0x45==data) {
request_CounterBb();
} else if (0x46==data) {
gain_set_30();
} else if (0x47==data) {
gain_set_50();
} else if (0x50==data) {
request_All();
}else if (0x51==data) {
request_AllwithGain();
} else if (0x60==data) {
show_temperature();
} else if (0x70==data) {
send_SCI0(0xc0);send_SCI0(0x03);
send_data();
send_SCI0(0xc0);
} else if (0x80==data) {
send_SCI0(0xc0);send_SCI0(0x03);
CheckPort();
send_SCI0(0xc0);
} else if (0x90==data) {
gradual_hv_set_OFF();
} else if (0x81==data) {
send_SCI0(0xc0);send_SCI0(0x03);
KEK_send_data();
send_SCI0(0xc0);
} else if (0x82==data) {
send_SCI0(0xc0);send_SCI0(0x03);
KEK_reset();
send_SCI0(0xc0);
} else if (0x83==data) {
send_SCI0(0xc0);send_SCI0(0x03);
KEK_start();
send_SCI0(0xc0);
} else if (0x84==data) {
send_SCI0(0xc0);send_SCI0(0x03);
visual_scaler_START();
send_SCI0(0xc0);
} else if (0x85==data) {
send_SCI0(0xc0);send_SCI0(0x03);
visual_scaler_STOP();
send_SCI0(0xc0);
} else if (0x86==data) {
send_SCI0(0xc0);send_SCI0(0x03);
visual_scaler_RESET();
send_SCI0(0xc0);
} else if (0xa0==data) {
send_SCI0(0xc0);send_SCI0(0x03);
tmp_log_on();
send_SCI0(0xc0);
} else if (0xa1==data) {
send_SCI0(0xc0);send_SCI0(0x03);
tmp_log_off();
send_SCI0(0xc0);
}
}
__interrupt(vect=54) void INT_TXI0(void) {/* sleep(); */}
__interrupt(vect=55) void INT_TEI0(void) {/* sleep(); */}
__interrupt(vect=56) void INT_ERI1(void) {
SCI1.SSR.BYTE &= 0xc7; // clear error flag
}
__interrupt(vect=57) void INT_RXI1(void) {}
__interrupt(vect=58) void INT_TXI1(void) {/* sleep(); */}
__interrupt(vect=59) void INT_TEI1(void) {/* sleep(); */}
__interrupt(vect=60) void INT_ADI(void) {/* sleep(); */}
```

## dbsct.c

```
#pragma section $DSEC
static const struct {
  char *rom_s;
  /* Start address of the initialized data section in ROM */

  char *rom_e;
  /* End address of the initialized data section in ROM   */

  char *ram_s;
  /* Start address of the initialized data section in RAM */

}DTBL[]= {
  {__sectop("D"), __secend("D"), __sectop("R")},
};
#pragma section $BSEC
static const struct {
  char *b_s;
  /* Start address of non-initialized data section */

  char *b_e;
  /* End address of non-initialized data section */
}BTBL[]= {
  {__sectop("B"), __secend("B")},
};
```

## sbrk.c

```
#include <stdio.h>
#include "sbrk.h"
#pragma pack 2
static  union {
short  dummy ;
char heap[HEAPSIZE]; /* sbrk  */
 }heap_area ;
#pragma unpack

static  char  *brk=(char *)&heap_area;

/*********************************************************/
/*  sbrk:f[^o                                */
/*       ^[lFtAhXij */
/*                   -1                          isj */
/*********************************************************/
extern  char  *sbrk(int size)   /* tTCY */
{
      char  *p ;
      if (brk+size>heap_area.heap+HEAPSIZE)
*/ ‘FbN */

  return (char *)-1 ;
      p=brk ;            /* t     */
      brk += size ;      /* IAhXXV */
      return p ;
}
```

## resetprg.c

```
#include <machine.h>
#include <_h_c_lib.h>
#include "stacksct.h"

extern void main(void);

#pragma section ResetPRG

__entry(vect=0) void PowerON_Reset(void)
{
 set_imask_ccr(0);
_INITSCT();
set_imask_ccr(0);
main();
sleep();
}
```

## USB.C

```
#include <stdio.h>
#include <string.h>
#include "usbn9602.h"

static void RegisterSet();
static void ResetUSB();
static void WakeupUSB();

static void rx0();
static void rx1();
static void tx0();
static void tx1();
static void nako0();
static void nako1();
static void naki0();
static void naki1();

static void clrfeature();
static void setfeature();
static void getdescriptor();
static void send_desc_sub(void *ptr,int size);
static void send_desc();
static void getstatus();
static void setconfiguration();

static void SetStallUSB(int adr);
static void ClearStallUSB(int adr);
static void FlushRXC(int no);
static void FlushTXC(int no);
static void TxTogle(int no);

static void WriteUSB(int adr,int data);
static unsigned char ReadUSB(int adr);
static int ReadUSBBurst(int adr,int adr2,char *buff,int cnt);
static int WriteUSBBurst(int adr,int adr2,char *buff,int cnt);

/*---------------------------------*/
static int SendTX1();

/*---------------------------------*/
int get_inbufflen(void);
void init_usbbuff(void);
int write_inbuff(char *p,int size);
int get_outbufflen(void);
int write_buff(char *p,int size);
int read_outbuff(char *p,int size);
/*---------------------------------*/
void IRQ5_disable();
void IRQ5_enable();

static unsigned char usbevent;
/* USBCxg*/
static unsigned char SETADDR;
/* AhXZbg*/
static unsigned char configno;
```

```
/* RtBO[VNO*/
static unsigned char usbbuff[64];
/* obt@*/
static unsigned char rx1buff[64];
static unsigned char rx2buff[64];
static unsigned char STALLD;
/* ECP*/
static unsigned char DATA0_1;
/* USB_TXTGLtO*/
static char senddesc;
/* 1 = fBXNv^M */
static int desc_size;
/* fBXNv^MTCY */
static char *desc_ptr;
/* fBXNv^|C^ */


static const unsigned char
epctbl[8] =
{USB_EPC0,USB_EPC1,USB_EPC2,USB_EPC3,USB_EPC4,USB_EPC5,USB_EPC6,USB_EPC0};
static int txcreg[4] = {USB_TXC0,USB_TXC1,USB_TXC2,USB_TXC3};
static int rxcreg[4] = {USB_RXC0,USB_RXC1,USB_RXC2,USB_RXC3};

/*-------------------------------------------------------------*/
/*-------------------------------------------------------------*/
static const unsigned char dev_desc[] = {
0x12,       /* length of this desc.*/
0x01,       /* foCXEfBXNv^ 1*/
0x00,0x01,    /* USB Version 1.0*/
0x00,       /* device class NX*/
0x00,       /* device subclass*/
0x00,       /* device protocol*/
0x08,       /* EP0pPbgTCY*/
0xfe,0xff,    /* vendor ID Tv*/
0x10,0x00,    /* product ID*/
0x01,0x00,    /* revision ID*/
0x01,       /* index of manuf. string*/
0x01,       /* index of prod.  string*/
0x02,       /* index of ser. # string*/
0x01       /* bNumConfigurations*/
};

/* RtBO[VfBXNv^ */
static const unsigned char cfg_desc[] = {
0x09,       /* length of this desc.*/
0x02,       /* RtBO[VEfBXNv^*/
9+9+7*3,
/* C^[tF[X^
   Gh|CgfBXNv^v CFG + IF + EP*3 */
0x00,       /**/
0x01,       /* C^[tF[X 1*/
0x01,       /* RtBO[V 1*/
0x00,       /* index of config. string*/
0x80,       /* attr.: self powered D6=d*/
300,       /* ;max power (100 mA)*/
0x09,       /* length of this desc.*/
0x04,       /* INTERFACE descriptor*/
0x00,       /* interface number*/
0x00,       /* alternate setting*/
0x03,       /* # of (non 0) endpoints*/
0x00,       /* interface class*/
0x00,       /* interface subclass*/
0x00,       /* interface protocol*/
0x03,       /* index of intf. string*/

/*static const unsigned char endp_desc[] = {*/
/* pipe 0 */
7,       /* length of this desc.*/
5,       /* ENDPOINT descriptor */
0x81,       /* address (IN)*/
0x02,       /* attributes  (BULK)*/
0x40,0x00,    /* max packet size (64)*/
255,       /* interval (ms)*/
7,       /* length of this desc.*/
5,       /* ENDPOINT descriptor*/
0x02,       /* address (OUT)*/
0x02,       /* attributes  (BULK)*/
0x40,0x00,    /* max packet size (64)*/
255,       /* interval (ms)*/
7,       /* length of this desc.*/
5,       /* ENDPOINT descriptor*/
0x83,       /* address (IN)*/
0x02,       /* attributes  (BULK)*/
0x40,0x00,    /* max packet size (64)*/
255,       /* interval (ms)*/
};


static const char lang_data[] = {
  4,3,9,4       /* LANGID (English)*/
};

static const char mfg_str[] = {
  18,3,
  'U',0,'S',0,'B',0,
  ' ',0,'T',0,'E',0,'S',0,'T',0,
};


static const char nbr_str[] = {
  8,3,
  '1',0,'.',0,'0',0,
};

static const char int_str[] = {
  34,3,
  'U',0,'S',0,'B',0,' ',0,
  'T',0,'E',0,'S',0,'T',0,
  ' ',0,'P',0,'R',0,'O',0,'G',0,'R',0,'A',0,'M',0,
};

static void wait(int c)
{
  int i,j;
  for(j=0;j<c;j++)
    {
      for(i=0;i<0x682;i++)
{
}
    }
}

/*-----------------------------------------------*/
/* USB */
void InitUSB()
{
  init_usbbuff();
  ResetUSB();
  RegisterSet();
  WakeupUSB();
}

static void RegisterSet()
{
  STALLD = 0;
  senddesc = 0;
  DATA0_1 = 0;
  SETADDR = 0;

  WriteUSB(USB_FAR,USB_AD_EN+0); /* AhX */
  WriteUSB(USB_EPC0,USB_EP_EN); /* EP0 佳醗勿 */
  WriteUSB(USB_NAKMSK,USB_NAK_00); /* NAK MASK 霈   */
  WriteUSB(USB_TXMSK,USB_TXFIFO0+USB_TXFIFO1+USB_TXFIFO2+USB_TXFIFO3);
  /* TX MASK 霈   */
  WriteUSB(USB_RXMSK,USB_RXFIFO0+USB_RXFIFO1+USB_RXFIFO2+USB_RXFIFO3);
  /* RX MASK 霈   */
  WriteUSB(USB_ALTMSK,USB_SD3+USB_RESET_A); /* ALT MASK 霈   */
  WriteUSB(USB_MAMSK,USB_INTR_E+USB_RX_EV+USB_NAK+USB_TX_EV+USB_ALT);
  /* MAIN MASK 霈   */
  FlushTXC(0);
```

```
  FlushRXC(1);
  FlushTXC(1);
  WriteUSB(USB_TXC1,0);
  WriteUSB(USB_RXC1,0);
  WriteUSB(USB_RXC0,USB_RX_EN); /* RX0 佳醚勿 */
}

static void ResetUSB()
{
  WriteUSB(USB_MCNTRL,USB_SRST+USB_VGE); /* USBZbg 3.3V */
  wait(100);        /* 100msec */
  WriteUSB(USB_MCNTRL,USB_INT_L_P+USB_VGE);
  /* active low push pull */

  WriteUSB(USB_CCONF,USB_CLKDIV-1); /* 48MHz/4 = 12MHz */
}

static void WakeupUSB()
{
  WriteUSB(USB_NFSR,USB_OPR_ST); /*  */
  WriteUSB(USB_MCNTRL,USB_INT_L_P+USB_NAT+USB_VGE);
  /* USB 棉拔 */

}

/* USB|[gf[^\ */
/* [hAXe[^XWX^ */

/*-------------------------------------------------------------*/
/* USB */
#ifdef __GNUC__
void usb_int()  __attribute__ ((interrupt_handler));
#endif
void usb_int()
{
  unsigned char nakevent,rxevent,txevent,altevent;
  char reg;


  usbevent = ReadUSB(USB_MAEV);

  if( usbevent & USB_NAK )
    {
      nakevent = ReadUSB(USB_NAKEV);
      if( nakevent & USB_NAK_O0 )
{
  nako0();
}
      if( nakevent & USB_NAK_O1 )
{
  nako1();
}
      else if( nakevent & USB_NAK_I0 )
{
  naki0();
}
      else if( nakevent & USB_NAK_I1 )
{
  naki1();
}
    }
  else if( usbevent & USB_RX_EV )
    {
      rxevent = ReadUSB(USB_RXEV);
      if( rxevent & USB_RXFIFO0 )
{
  rx0();
}
      else if( rxevent & USB_RXFIFO1 )
{
  rx1();
}
    }
  else if( usbevent & USB_TX_EV )
    {
      txevent = ReadUSB(USB_TXEV);
      if( txevent & USB_TXFIFO0 )
{
  tx0();
}
      else if( txevent & USB_TXFIFO1 )
{
  tx1();
}
    }
  else if( usbevent & USB_ALT )
    {
      altevent = ReadUSB(USB_ALTEV);
      if( altevent & USB_RESET_A )
{
  /* Zbg */
  RegisterSet();
  WakeupUSB();
}
      else if( altevent & USB_SD3 )
{
  /* TXyh */
  WriteUSB(USB_ALTMSK,USB_RESUME_A+USB_RESET_A);
  /* ALTMSK 霜  */

  WriteUSB(USB_NFSR,USB_SUS_ST);
  /* 桧爪雹庸烟爪*/

}
      else if( altevent & USB_RESUME_A )
{
  /* W[ */
  WriteUSB(USB_ALTMSK,USB_SD3+USB_RESET_A);
  /* ALTMSK 霜  */

  WriteUSB(USB_NFSR,USB_OPR_ST);
  /* 桧爪\*/
}
}
}

/*=======================================================
  RXCxg
=======================================================*/
/* RX0(system) */
/*
 NGXgR[h
 0 byte
 D7 ... f[^ 0=zXg->foCX , 1=foCX->zXg
 D6-D5 ... ^Cv
 0:W , 1:NX , 2:x_ , 3:\
 D4-D0 ... M
 0:foCX , 1:C^[tFCX , 2:Gh|Cg , 3:
 1 byte
 NGXg
 2 byte
 value
 2 byte
 index
 2 byte
 length
*/
static void rx0()
{
  unsigned char rxstat;
  rxstat = ReadUSB(USB_RXS0);
  if( rxstat & USB_SETUP_RX )
    {
      ReadUSBBurst(USB_RXD0,USB_RXS0,(char*)usbbuff,8);
      FlushRXC(0);
      FlushTXC(0);
      ClearStallUSB(USB_EPC0);
      if( (usbbuff[0] & 0x60) == 0 )
{
  /* WNGXg */
```

```
  switch( usbbuff[1] )
    {
    case USB_CLEAR_FEATURE :
      clrfeature();
      break;
    case USB_GET_CONFIGURATION :
      WriteUSB(USB_TXD0,configno);
      break;
    case USB_GET_DESCRIPTOR :
      getdescriptor();
      break;
    case USB_GET_STATUS :
      getstatus();
      break;
    case USB_GET_INTERFACE :
      WriteUSB(USB_TXD0,0);
      break;
    case USB_SET_ADDRESS :
      WriteUSB(USB_EPC0,USB_DEF);
      SETADDR = usbbuff[2]|USB_AD_EN;
      WriteUSB(USB_FAR,SETADDR);
      break;
    case USB_SET_CONFIGURATION :
      setconfiguration();
      break;
    case USB_SET_FEATURE :
      setfeature();
      break;
    case USB_SET_INTERFACE :
      if( usbbuff[2] != 0 )
SetStallUSB(USB_EPC0);
      break;
    default :
      /* ‘ */
      SetStallUSB(USB_EPC0);
      break;
    }
}
      else if( (usbbuff[0] &0x60 ) ==  0x20 )
{
  /* NXNGXg */
  SetStallUSB(USB_EPC0);
}
      else if( (usbbuff[0] &0x60 ) ==  0x40 )
{
  /* x_NGXg */
  SetStallUSB(USB_EPC0);
}
      else
{
  /* ‘ */
  SetStallUSB(USB_EPC0);
}
      DATA0_1 |= 1;
      /* SETUP 濮洌 LWDATA1M */

      TxTogle(0);
    }
  else
    {
      if( senddesc )
{
  senddesc = 0;
}
      FlushTXC(0);
      WriteUSB(USB_RXC0,USB_RX_EN);
    }
}

/*-----------------------------------------------------------*/
/* RX1 M */
static void rx1()
{
  int cnt;
  unsigned char rxstat;

  rxstat = ReadUSB(USB_RXS1);/* RX1Xe[^X */
  if( !(rxstat & (USB_SETUP_RX|USB_RX_ERR)) )
    /* SETUP,ERROROpPbg */

    {
      /* zXgf[^M */
      cnt = ReadUSBBurst(USB_RXD1,USB_RXS1,(char*)rx1buff,64);
      /* FIFOf[^ */

      write_inbuff((char*)rx1buff,cnt);
      /* Oobt@ */

    }
  FlushRXC(1);       /* obt@tbV */
  WriteUSB(USB_RXC1,USB_RX_EN);  /* M */
}

/*-----------------------------------------------------------*/
/* RX2 M(not use) */
static void rx2()
{
unsigned char rxstat;
rxstat = ReadUSB(USB_RXS2);
}

/*============================================================
TXCxg
============================================================*/
/* TX0 MI */
static void tx0()
{
  unsigned char txstat;
  txstat = ReadUSB(USB_TXS0);
  if( (txstat & USB_ACK_STAT) && (txstat & USB_TX_DONE) )
    {
      /* ok */
      FlushTXC(0);
      if( senddesc )
{
  send_desc();
  TxTogle(0); /* TX0M */
}
      else
{
  WriteUSB(USB_RXC0,USB_RX_EN); /* RX0M */
}
    }
}

/*-----------------------------------------------------------*/
/* TX1MI */
static void tx1()
{
  unsigned char txstat;
  txstat = ReadUSB(USB_TXS1);
  if( (txstat & USB_ACK_STAT) && (txstat & USB_TX_DONE) )
    {
      /*
 MIMf[^M
 Mf[^AHOST0oCg
      */
    }
  else
    {
      /*
 TCYsA

      */
    }
}

/*============================================================
  NAKCxg
============================================================*/
/*
```

```
  NAKCxgAG[M
  NAKOENABLEAO
*/
static void nako0()
{
}
static void nako1()
{
}
static void naki0()
{
}

static void naki1()
{
}


/*=========================================================
WNGXg
=========================================================*/
/* I@ */
static void clrfeature()
{
  if( (usbbuff[0] & 3) == 2 )
    {
      /* Gh|Cgf[^ */
      if((usbbuff[3]&7) != 0 )
ClearStallUSB(epctbl[usbbuff[3]&7]);
      STALLD &= -1 ^ (1<<(usbbuff[3]&7));
    }
}
static void setfeature()
{
  if( (usbbuff[0]&3) == 2 ) /* ENDPOINT */
    {
      /* Gh|Cgf[^ */
      if((usbbuff[3]&7) != 0 )
SetStallUSB(epctbl[usbbuff[3]&7]);
      STALLD |= (1<<(usbbuff[3]&7));
    }
}

/*-----------------------------------------------------------*/
/* fBXNv^ */
static void getdescriptor()
{
  DATA0_1 &= 0xfe;
  switch( usbbuff[3] )
    {
    case USB_DEVICE :
      send_desc_sub((void*)dev_desc,dev_desc[0]);
      break;
    case USB_CONFIGURATION :
      {
send_desc_sub((void *)cfg_desc,cfg_desc[2]);
break;
      }
    case USB_XSTRING :
      {
switch( usbbuff[2] )
  {
  case 0 :
    send_desc_sub((void *)lang_data,lang_data[0]);
    break;
  case 1 :
    send_desc_sub((void *)mfg_str,mfg_str[0]);
    break;
  case 2 :
    send_desc_sub((void *)nbr_str,nbr_str[0]);
    break;
  case 3 :
    send_desc_sub((void *)int_str,int_str[0]);
    break;
  }
break;
      }
    default :
      {
      }
    }
}

static void send_desc_sub(void *ptr,int size)
{
  desc_size = (usbbuff[7] << 8) + usbbuff[6];
  if( desc_size > size ) desc_size = size;
  /* Mvobt@f[^ */
  desc_ptr = ptr;
  senddesc = 1;  /* fBXNv^MtO */
  send_desc();
}

static void send_desc()
{
  int sz;
  sz = 8;
  if( desc_size == 0 ) return;
  if( desc_size <= 8 ) sz = desc_size;
  sz = WriteUSBBurst(USB_TXD0,USB_TXS0,desc_ptr,sz);
  desc_size -= sz;
  desc_ptr += sz;
  if( desc_size == 0 ) senddesc = 0;
}
/*-----------------------------------------------------------*/
/* Xe[^X */
static void getstatus()
{
  int data,ep;
  data = usbbuff[0]&3;
  if( (data == 0) || (data == 1) )   /* DEVICE,INTERFACE */
    {
      WriteUSB(USB_TXD0,0);
      WriteUSB(USB_TXD0,0);
    }
  else if( data== 2)    /* Gh|Cg */
    {
      ep = usbbuff[3]&7;
      /* epSTALLM */
      if( STALLD & (1<<ep) ) WriteUSB(USB_TXD0,1);
      else WriteUSB(USB_TXD0,0);
    }
  else
    {
      WriteUSB(USB_TXD0,0);
    }
}

/*-----------------------------------------------------------*/
static void setconfiguration()
{
  configno = usbbuff[2];
  if( configno == 0 )
    {
      WriteUSB(USB_EPC1,0); /* EPC1gps */
      WriteUSB(USB_EPC2,0); /* EPC2gps */
      WriteUSB(USB_EPC3,0); /* EPC3gps */
      WriteUSB(USB_EPC4,0); /* EPC4gps */
      WriteUSB(USB_EPC5,0); /* EPC5gps */
      WriteUSB(USB_EPC6,0); /* EPC6gps */
    }
  else
    {
      STALLD = 0;
      FlushTXC(1);
      WriteUSB(USB_EPC1,USB_EP_EN+01);
      /* EPC1 隷爪攴 1 腸醗勿 */

      WriteUSB(USB_TXC1,USB_TX_EN|USB_TX_LAST);
      /* TX1M */
```

```
      FlushRXC(1);
      WriteUSB(USB_EPC2,USB_EP_EN+02);
      /* EPC2 隸爪攴 2 腸醜勿 */

      WriteUSB(USB_RXC1,USB_RX_EN); /* RX1M */
      /*
USB_TX_LASTAREADXg[B
Af[^OREADObyteB
hostNGXgsA
NGXgXg[B
      */
    }
}

/*=========================================================
  p[‘
 =========================================================*/
/* STALLZbgNA */
static void SetStallUSB(int adr)
{
  WriteUSB(adr,ReadUSB(adr) | 0x80);
}

static void ClearStallUSB(int adr)
{
  WriteUSB(adr,ReadUSB(adr)&0x7f);
}

/* FIFOtbV */
static void FlushRXC(int no)
{
  WriteUSB(rxcreg[no],USB_FLUSH);
}

static void FlushTXC(int no)
{
  int d;
  d = ReadUSB(txcreg[no]);
  d |= USB_FLUSH;
  WriteUSB(txcreg[no],d);
}

/* MItOZbg */
/* reg = USB_TXC0‘6 */
static void TxTogle(int no)
{
  unsigned char d;
  d = USB_TX_EN;
  if( DATA0_1 & (1<<no) ) d |= USB_TX_TOGL;
  else d &= ~USB_TX_TOGL;
  d |= USB_TX_LAST;
  WriteUSB(txcreg[no],d);
  DATA0_1 ^= (1<<no);
}

/*-------------------------------------------------*/
/* USBAhX */
static unsigned char ReadUSB(int adr)
{
  USB9602R = (unsigned char)adr;
  return( USB9602D );
}
/* USBAhX */
static void WriteUSB(int adr,int data)
{
  USB9602R = (unsigned char)adr;
  USB9602D = (unsigned char)data;
}


/* o[Xgg] */
static int ReadUSBBurst(int adr,int adr2,char *buff,int cnt)
{
  int i;
  int rcnt;
  USB9602R = (unsigned char)adr;
  for(rcnt=0,i=0;i<cnt;i++)
    {
      if( (ReadUSB(adr2) & 0xf) == 0 ) break;

      USB9602R = (unsigned char)adr;
      *buff = USB9602D;
      buff++;
      rcnt++;
    }
  return(rcnt);
}

static int WriteUSBBurst(int adr,int adr2,char *buff,int cnt)
{
  int i,scnt;
  for(scnt=0,i=0;i<cnt;i++)
    {
      if( (ReadUSB(adr2) & 0x1f) == 0 ) break;

      USB9602R = (unsigned char)adr;
      USB9602D = *buff;
      buff++;
      scnt++;
    }
  return(scnt);
}

/*=========================================================
  TvvO
=========================================================*/
/* TX1M[‘ */
/*
  USBIMAA
  TX1MIobt@(outbuff)f[^MB
  AHOSTIREADsB
  O^C~OEPC1FIFOobt@TCYB
*/
static int SendTX1()
{
  int c,cnt,sz,i;

  cnt = 0;
  sz = read_outbuff((char*)rx2buff,64);  /* FIFO64byte */
  FlushTXC(1);  /* Mobt@tbV */
  if( sz != 0 )
    {
      cnt = WriteUSBBurst(USB_TXD1,USB_TXS1,(char*)rx2buff,sz);
      /* o[Xg] */
    }
  TxTogle(1); /* MI */
return( cnt ); /* Mf[^ sz==cnt */
}

/*-----------------------------------------------------------*/
/*
  Mobt@
  inbuffHOSTf[^obt@
  outbuffHOSTf[^p
  TvOobt@B
  A256oCgmBobt@
  CB
*/
#define USBBUFFLEN 256     /* obt@TCY */
static int inpos,inlen;   /* obt@uATCY */
static int outpos,outlen; /* oobt@uATCY */
static char inbuff[USBBUFFLEN]; /* Oobt@ */
static char outbuff[USBBUFFLEN]; /* oOobt@ */

/*-----------------------------------------------------------*/
/* obt@*/
/*-----------------------------------------------------------*/
void init_usbbuff()
{
```

```
inpos = inlen = 0;
outpos = outlen = 0;
}

/*-------------------------------------------------------------*/
/* HOSTMobt@ */
/* char   *p obt@|C^ */
/* int size TCY */
/* l TCY */
/*-------------------------------------------------------------*/
int write_inbuff(char *p,int size)
{
int i;
IRQ5_disable();

for(i=0;i<size;i++)
{
     if( inlen >= USBBUFFLEN ) break;
     inbuff[inpos] = *p;
     inpos = (inpos + 1)%USBBUFFLEN;
     inlen++;
        p++;
    }
IRQ5_enable();
return(i);
}
/*-------------------------------------------------------*/
/* Mobt@ */
/* char      *p obt@|C^        */
/* int      size TCY */
/* l             TCY        */
/*-------------------------------------------------------*/
int write_buff(char *p,int size)
{
int i;
//INTC.IER &= (-1^0x20);    /* IRQ5 Disable */
IRQ5_disable();
for(i=0;i<size;i++)
{
     if( outlen >= USBBUFFLEN ) break;
     outbuff[outpos%USBBUFFLEN] = *p;
     outpos = (outpos + 1)%USBBUFFLEN;
     outlen++;
        p++;
    }
//INTC.IER |= 0x20;     /* IRQ5 Enable */
IRQ5_enable();
return(i);
}

/*-------------------------------------------------------*/
/* Mobt@*/
/* char *p obt@|C^ */
/* int size obt@TCY */
/* l         TCY */
/*-------------------------------------------------------*/
int read_outbuff(char *p,int size)
{
  int i;
  IRQ5_disable();
  for(i=0;outlen>0;i++)
    {
      if( i >= size ) break;
      p[i] = outbuff[ (USBBUFFLEN+outpos-outlen)%USBBUFFLEN ];
      outlen--;
    }
  IRQ5_enable();
  return(i);
}
/*-------------------------------------------------------*/
/* Mobt@*/
/* char *p obt@|C^ */
/* int size obt@TCY */
/* l TCY        */
/*-------------------------------------------------------*/
int read_buff(char *p,int size)
{
  int i;
  //INTC.IER &= (-1^0x20); /* IRQ5 Disable */
  IRQ5_disable();
  for(i=0;inlen>0;i++)
    {
      if( i >= size ) break;
      p[i] = inbuff[ (USBBUFFLEN+inpos-inlen)%USBBUFFLEN ];
      inlen--;
    }
  //INTC.IER |= 0x20; /* IRQ5 Enable */
  IRQ5_enable();
  return(i);
}
/*-----------------------------------------------------*/
/* Mobt@TCY*/
/*-----------------------------------------------------*/
int get_inbufflen()
{
return( inlen%USBBUFFLEN );
}

/*-----------------------------------------------------*/
/*          Mobt@TCY*/
/*-----------------------------------------------------*/
int get_outbufflen()
{
return( outlen%USBBUFFLEN );
}

void IRQ5_disable()
{
INTC.IER.BIT.IRQ5E = 0;/* IRQ5 Disable */
}

void IRQ5_enable()
{
INTC.IER.BIT.IRQ5E = 1;/* IRQ5 Enable */
}
```

# 謝辞

本研究を行うに当たり、多くの方々の御世話になりました。指導教官の河合誠之教授、そして片岡 惇助手に深謝致します。お二人は、宇宙物理学についてまったく知識のない私に対し指導して下さりました。また渡辺靖志教授、石野宏和助手をはじめ、研究室の皆様にも日々の生活で非常に御世話になりました。

Cute-1.7プロジェクトチームの博士研究員の古徳さん、博士課程1年の谷津さん、修士課程2年の倉本さん、そして松永三郎教授をはじめ松永研究室の皆様には、非常に御世話になりました。古徳さんには、実験屋としての心構え、そして論文執筆に当たり細かく指導して頂きました。谷津さんには、実験を行う上での多くの助言を頂きました。そして倉本さんには、右も左も分からなかった私に対し、日々の実験から生活まで常にやさしく接して下さり、そして指導して下さいました。御迷惑ばかりお掛けしてしまいましたが、ともに実験を行うことができたことをとてもうれしく思います。

最後に、本研究に携わることができ、Cute-1.7 打ち上げという素晴しい場に立ち会えたことを関係者各位に深く感謝申し上げます。