１９９８年度 修士学位論文

BELLE 測定器

モニターシステムの製作

奈良女子大学大学院理学研究科

物理学専攻

高エネルギー物理学講座

横井　静香

1999 年 2 月

目次

はじめに **1**

1 B 中間子の物理 **4**

1.1 CP の破れ 4

1.2 小林・益川模型と B 中間子 5

1.3 B 中間子系での CP 非保存とその測定 7

1.3.1 B 中間子の生成と崩壊 7

1.3.2 $B^0 - \overline{B^0}$ 混合 8

1.3.3 B 中間子の崩壊における CP の破れ 10

1.3.4 時間測定の方法 11

2 B ファクトリー **13**

2.1 KEKB 加速器 13

2.1.1 電子・陽電子ビームのエネルギー 14

2.1.2 要求されるルミノシティ 15

2.2 BELLE 測定器 16

2.2.1 粒子崩壊点測定器 (SVD) 20

2.2.2 中央飛跡検出器 (CDC) 20

2.2.3 粒子識別装置 (ACC,TOF) 21

2.2.4 電磁カロリメータ (ECL) 21

2.2.5 K_L,μ 粒子検出器 (K_L, μ) 21

3 BELLE 測定器のモニターシステム **22**

3.1 測定器モニターシステムの概要 22

3.2 測定器モニターシステムの構造 26

3.2.1 測定器環境 26

3.2.2 電源電圧 40
3.3 測定器環境データの読みだし 47
3.3.1 HP3852A·HP3853A データロギングシステム 49
3.3.2 GPIB バス 50
3.3.3 SPARC ボード 51
3.3.4 ネットワークシェアードメモリー (NSM) 51

4 測定器モニターシステムの動作及び結果 55
4.1 測定器モニターシステムのソフトウェアのアルゴリズム 55
4.2 測定結果と動作実績 57

5 まとめ 62

A データ収集プログラム 64

図一覧

1.1 クォークの混合 5
1.2 ユニタリティ三角形 6
1.3 $B\overline{B}$ 混合のボックスダイアグラム 9

2.1 KEKB 加速器 15
2.2 BELLE 検出器 18
2.3 BELLE 検出器側面図 19

3.1 測定器モニターシステム全体図 24
3.2 BELLE-DAQ システム全体図 25
3.3 サーミスタ概観図 28
3.4 ヒュミター概観図 29
3.5 50m ケーブル取り付け時の湿度センサーのテスト・セットアップ . . 30
3.6 CsI 電磁カロリメータの全体像 33
3.7 CsI 結晶が並べられた様子 34
3.8 センサー取り付け位置（結晶） 35
3.9 コネクター BOX 36
3.10 温度・湿度モニター信号の流れ 37
3.11 セットアップ 37
3.12 回路図 38
3.13 38
3.14 TKO クレート配置図 41
3.15 TKO パッチパネル（モニターパネル） 42
3.16 コントロールパネル（スイッチパネル） 44
3.17 温度センサー取り付け位置 45
3.18 TKO クレート内部温度変化の様子（モジュール間） 45

3.19 TKO クレート内部温度変化の様子（フロントパネル） 46
3.20 TKO クレート内部温度変化の様子（クレート上部） 46
3.21 ５番目のスロットと１２番目のスロットとの温度比較 48
3.22 3852A のフロントパネル 49
3.23 3852A のリアパネル 49
3.24 リレーマルチプレクサのターミナルモジュール 53
3.25 GP スイッチ切りかえの様子 53
3.26 GPIB 構造図 54
3.27 NSM 構造図 54

4.1 データ収集プログラムの流れ（メインプログラム） 58
4.2 プログラムの流れ（インターロック） 59
4.3 測定結果 61

表一覧

2.1 各検出器の目的 17

3.1 HP3852A で測定される環境データ 27
3.2 結果 31
3.3 結果 1 35
3.4 温度センサー -1 39
3.5 温度センサー -2 39
3.6 温度センサー -3 39
3.7 湿度センサー 40
3.8 TKO クレート内部温度変化の様子 47

はじめに

現在、茨城県つくば市にある高エネルギー加速器研究機構（KEK）において、電子・陽電子を衝突させて大量のB中間子を造りだし、そこで起こる物理の研究を目的とした、「Bファクトリー」という大規模な素粒子実験が始まろうとしている。この実験の為の電子・陽電子衝突型加速器（KEKB）及び検出器（BELLE）の建設が開始されたのが1994年であり、そこから5年の建設期間をかけて、今年、1999年4月にいよいよ本実験を開始するところまで来ることができた。

このBファクトリーでは電子·陽電子の衝突によってbクォークと$\overline{b}$クォークの共鳴状態である$\Upsilon(4S)$状態を年間 10^8 個作り出すことを目標としている。$\Upsilon(4S)$状態はほぼ100 %の確率でB中間子と 反B中間子の対に崩壊するので、ここでは多量のB中間子を得ることが可能となる。Bファクトリーにおいてはこの多量のB中間子を用いて、B中間子の崩壊過程におけるCP対称性の破れの現象を研究することが最大の目的である。

CP対称性の破れとは、自然界において、物質を構成する粒子と反物質を構成する反粒子とのあいだの対称性、同等性が破れていることを意味している。このCP対称性の破れの現象は、1960年代にK中間子の系において初めて観測された。しかし、それから35年たった現在においてもK中間子以外の系ではいまだ観測されていない。素粒子の標準理論では、3世代あるクオーク間のミキシングを考え、そこに存在する位相の自由度がCP対称性の破れを生じると考えられている。これは1973年に小林、益川が始めて提唱したモデルである。これにもとづくと 10% 以上の大きなCP対称性の破れがB中間子系に予想されている。K中間子におけるCPの破れの大きさおおよそ 10^{-3} オーダーであることを考慮するとこれは非常に大きな破れである。従って、B中間子系のCP非保存現象の研究はこの標準理論の重要な検証であるとともに新しい物理の発見を導くおおきな可能性を秘めている。CP非保存現象は、また、なぜ我々の住むこの自然界（宇宙）は、物質のみでなりたっており、反粒子・反物質はほとんど存在しないのかの謎をひも解く重要な鍵であると考えられている。

B 中間子系における CP 非保存の有無を実際に実験で確認するためには、加速器および検出器両方において非常にすすんだ技術が要求されている。たとえば、大きな CP の非対称性が期待されている B 中間子の崩壊モードはその崩壊確率は 10^{-4} から 10^{-5} と非常に小さい。そのために非常に強いビームの強度 (ルミノシティー) が必要となる。必要とされるビーム電流は 1A のオーダーに達する。この強度はこれまでの電子 陽電子型加速器のビーム強度より 2 桁大きな値である。

また、検出器も高い分解能および検出効率を持ち、かつ大型で高性能であることが要求される。B 中間子と $\bar{B}$ 中間子系は平均 5 個の荷電粒子と 5 個の光子に崩壊する。この系における CP 非対称性を測定するには、観測された多量の粒子群の中から B 中間子を再構成し、かつ B 中間子の崩壊点を正確に決めることが必要である。そのため検出器には、いい運動量分解能、エネルギー分解能、位置分解能とともに、荷電粒子の種類の識別 (電子、ミュー粒子、π, K, p) や光子の能力が必要である。このような要求を 1 つの測定器で満たすことは不可能なので、BELLE と名付けられた装置は多くの異なる構成要素より形成されている。

CP 非保存現象の詳細な研究にはこのような大型の検出器を１０年以上の長期に渡って安定に運転し続けることが必要となる。そのためには測定器の状態を絶えず監視しモニターすることが非常に大切である。例えば、潮解性のある CsI カウンターは一定の湿度を保たなければならず、常時湿度を監視しておく必要がある。また、熱によって読みだしのプレアンプが破損するなどの異常が起きないないよう温度を測定する必要がある。さらに異常が発見された時にも可能な限り制御するシステムを構築することが求められる。

私は、BELLE の一員として、このような 測定器の環境 (温度・湿度・ガス圧など) をモニターし、必要ならばそのコントロールを行なうシステム (測定器モニターシステム) の開発及び製作にここ 2 年間携わってきた。このモニターは温度・湿度・ガス圧などを BELLE 測定器全体でおよそ 1,200 点について常時モニターすることができるシステムである。私はモニターの実行と制御を行なうシステムを製作するとともに、各測定に接地されたモニターの動作チェックおよび、全体が正常に起動していることの動作確認を行なってきた。システム全体はほぼ完成し、いまは測定器の環境を常時モニターしながら今年 1999 年 4 月から始まる本実験を待っている所である。

本論文では、この環境モニターシステムの詳細および性能について報告する。本論文の構成は以下の通りである。第一章で B 中間子の物理について説明し、第二章でそれを実現するために建設されてきた、B ファクトリー加速器および BELLE 測

定器について説明する。第三章で 私が主に担当してきた、BELLE 測定器のモニターシステムの構成、及びさまざまな研究結果と製作過程・結果を示す。第四章ではモニター用ソフトウェアの構成と実際の性能について報告する。最後に全体のまとめを第五章で行なう。

第 1 章

B 中間子の物理

1.1 CP の破れ

自然界にはいろいろな対称性が存在する。その中には厳密になり立っているもの、また限られた条件でしか成立しないものがある。荷電共役（C）変換、空間反転（P）変換、時間反転（T）変換を組み合わせた CPT 変換のもとでは今までわかっている物理法則は全て不変である。しかし、個々の変換の対称性は保証されていない。自然界に 4 つある相互作用のうち重力相互作用、電磁相互作用、強い相互作用はこれら各々の変換に対する対称性を守っている。しかし、弱い相互作用による反応では荷電共役（C）変換、空間反転（P）変換に対して対称性が破れていることが確認されている。C 変換の破れの典型的な例は、π 中間子の崩壊である。崩壊 $\pi^+ \to \mu^+ + \nu_\mu$ に荷電共役変換を施したら、この変換はスピンの向きを変えないので、左巻きの反ニュートリのが出てこなければならなくなる。右巻きのスピンを持つニュートリノや、左巻きの反ニュートリノは（ほとんど）存在しないことが実験からわかっているので、ニュートリノが関わる崩壊は、C 変換に対して不変ではない。P 変換については、リーとヤンの予言に従い、1957 年にウーがコバルト原子核のベータ崩壊を使って実証した。

ところが、これら両方を同時に行う CP 変換を考えると、左巻きニュートリノは右巻き反ニュートリノに変わるので実験事実に抵触せず、この CP における不変性は破られていないと考えられてきた。

しかし、1964 年クリステンソン達が、中性 K 中間子の崩壊において、CP 対称性が破れていることを実験的に明らかにした [1]。そして 1973 年、小林・益川は「クォークのフレーバー間に世代間混合があって、少なくとも 3 世代 6 種類のクォークがあれば CP の破れを導く複素位相が意味のあるパラメーターとして残ることができる」

という描像を発表した [2]。さらに、CP の破れは B 中間子系で大きく見えるであろうことが三田 · カーターにより指摘された [3]。

次節では、B 中間子の崩壊における CP の破れが、小林 · 益川模型の枠組の中でどのように説明されるかについて述べる。

1.2 小林 · 益川模型と B 中間子

クォークは、その質量固有状態が弱い相互作用の固有状態とは異なる。つまり、弱い相互作用の固有状態は質量の固有状態の線型結合で書くことが出来、このことをクォークの混合と呼ぶ。そのため、弱い相互作用のうち、荷電カレント相互作用においてはクォークが他の世代のクォークに遷移する（図 1.1）。

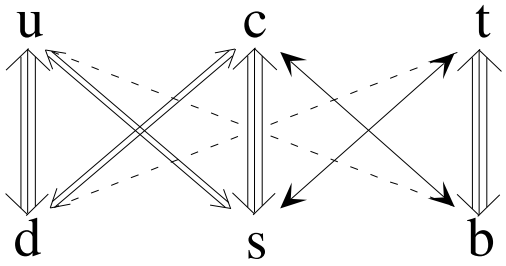


図 1.1: クォークの混合

この混合の割合を表すのが、カビボ · 小林 · 益川 (CKM) 行列である。

$$V = \begin{pmatrix} V_{ud} & V_{us} & V_{ub} \\ V_{cd} & V_{cs} & V_{cb} \\ V_{td} & V_{ts} & V_{tb} \end{pmatrix}$$

ヴォルフェンシュタイン表示 [4] では、CKM 行列は

$$V = \begin{pmatrix} 1 - \frac{\lambda^2}{2} & \lambda & \lambda^3 A(\rho - i\eta) \\ -\lambda & 1 - \frac{\lambda^2}{2} & \lambda^2 A \\ \lambda^3 A(1 - \rho - i\eta) & -\lambda^2 A & 1 \end{pmatrix}$$

と表される。この表示においては、CP を破る複素位相はパラメータ η に現れる。

これまでに得られている実験結果から、各パラメータには次のような制限が与えられている。

$$\lambda = 0.221 \pm 0.002$$

$$A = 0.798 \pm 0.082$$

$$\sqrt{(1-\rho)^2+\eta^2} = 0.33 \pm 0.068$$

これらの制限では、ρ と η に対し各々単独の値は決定されていない。そこで、以下に ρ と η を決定する上で B 中間子の崩壊過程の観測が大きな寄与をすることを示す。

CKM 行列は $V^\dagger V = 1$ をみたすユニタリー行列であるので、特に b クォークの関与する部分を抜き出すと次の式が得られる。

$$V_{ub}^* V_{ud} + V_{cb}^* V_{cd} + V_{tb}^* V_{td} = 0$$

これは複素平面上で三角形を描きユニタリティ三角形と呼ばれる (図 1.2) 。こ

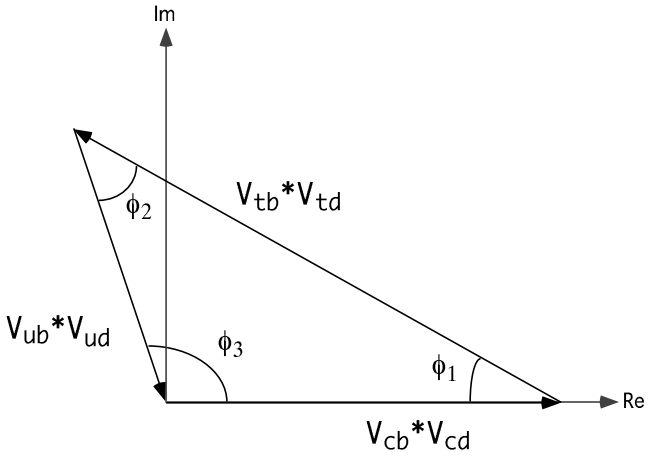


図 1.2: ユニタリティ三角形

の三角形の頂点の実部が ρ、虚部が η であるから、この三角形を測定から強く制限することが ρ、η を決定することと等価である。言い替えれば、ユニタリティ三角形の辺の長さと角の大きさを測定し、この三角形が矛盾なく成立するか否かを確認することが示されれば小林・益川模型の実験的検証である。

各辺の長さを決定する上で重要なのは $|V_{ub}|$、$|V_{cb}|$、そして $|V_{td}|$ である。$|V_{ub}|$ は B 中間子のセミレプトニック崩壊のうち、b クォークが c クォークに遷移する場合には運動学的に禁止される高運動量レプトンと、$B \to \rho\ell\nu$ なる崩壊過程の観測により決定することが出来る。V_{cb} は $B \to D^*\ell\nu$ なる崩壊過程の観測、$|V_{td}|$ は $B^0 - \overline{B^0}$ 混合の大きさの測定により、各々決定できる。

三つの内角 $\phi_1 = arg\left(\frac{V_{cd}V_{cb}^*}{V_{td}V_{tb}^*}\right)$、$\phi_2 = arg\left(\frac{V_{ud}V_{ub}^*}{V_{td}V_{tb}^*}\right)$、$\phi_3 = arg\left(\frac{V_{cd}V_{cb}^*}{V_{ud}V_{ub}^*}\right)$ は、B 中間子の崩壊における CP 非対称度の測定により決定される。

次節で述べるように、B 中間子の CP 非保存は K 中間子や D 中間子の場合に比べて二桁ほど大きいことが期待される。しかし、CP の固有状態への崩壊確率が 10^{-4} ～ 10^{-5} と小さいことが、その観測を難しくしている。そこで、10^7 個から 10^8 におよぶ大量の B 中間子を生成し、その崩壊過程を観測することが必要となる。

1.3 B 中間子系での CP 非保存とその測定

小林・益川模型を検証するには、既に測定の実例があるユニタリティ三角形の辺の長さのみならず、三つの内角（ϕ_1、ϕ_2、ϕ_3）をも決定し、これらを総合して三角形が成立しうるか否かを確認する必要がある。ユニタリティ三角形の三つの内角を決定するためには、実験的には CP 非対称度と呼ばれる量を測定する [5]。

そこで以下では、B 中間子の生成と崩壊の過程、CP 非対称度の測定原理と ϕ_1、ϕ_2、ϕ_3 との関係を説明する。

1.3.1 B 中間子の生成と崩壊

電子・陽電子衝突において、重心系エネルギーを $10.58GeV$ に設定すると $\Upsilon(4S)$ と呼ばれる共鳴状態を生成できる。$\Upsilon(4S)$ はほぼ 100 % B 中間子対に崩壊するので、B 中間子の研究に非常に適している。Υ は b クォーク、反 b クォーク対から成る共鳴状態で、1977 年にフェルミ研究所における $400GeV$ のエネルギーを持った陽子ビームを用いた固定標的実験で発見された。この実験では、入射陽子と標的の原子核の間の反応から生成した μ 粒子対の不変質量分布から基底状態の $1S$ 状態の $\Upsilon(1S)$ のみならず、動経方向の励起状態がさらに二つ（$\Upsilon(2S)$ と $\Upsilon(3S)$）確認された。

その後、コーネル大学の電子・陽電子衝突型加速器 CESR と CLEO 測定器を用いた実験によって Υ の性質は詳細に調べられ、現在では 5 番目の動経方向励起状態 $\Upsilon(5S)$ まで、少なくとも合計五つの Υ の存在が確認されている。B 中間子対の質

量は $\Upsilon(3S)$ の質量と $\Upsilon(4S)$ の質量の間にあるので、Υ からの B 中間子を得るには $\Upsilon(4S)$ または $\Upsilon(5S)$ を生成すればよいのであるが、後者は生成断面積が低いことと、b クォーク・$\overline{s}$ クォークからなる B_s 中間子・$\overline{B_s}$ 中間子対も生成するために実験条件が複雑になるため、通常 $\Upsilon(4S)$ を用いる。

CLEO グループは、終状態の粒子を全て測定器でとらえて再構成した B 中間子生成事象から B 中間子の質量を測定した。$B^{\pm}$、B^0、$\overline{B^0}$ の質量平均値として、

$$M_B = 5272.3 \pm 1.5 \pm 2.0 MeV/c^2$$

が得られている。

B 中間子の崩壊における CP 非保存においては、時間と共に B^0 が $\overline{B^0}$ に遷移したり、あるいは逆に $\overline{B^0}$ が B^0 に転換したりする B^0–$\overline{B^0}$ 混合という現象が深く関わっている。従って次節ではこの B^0 – $\overline{B^0}$ 混合について述べる。

1.3.2 B^0 – $\overline{B^0}$ 混合

中性 B 中間子（B^0）は時間と共に反 B 中間子（$\overline{B^0}$）に変化する。B 中間子が B^0 と $\overline{B^0}$ のどちらであったかは、セミレプトニック崩壊の生成物を観測することによって同定できる。それは B^0 と $\overline{B^0}$ はそれぞれ電荷の違うレプトンを含む終状態に崩壊するからである。

$$B^0 \to l^+ X$$

$$\overline{B^0} \to l^- X$$

B 中間子は混合により $B\overline{B}$ 間を振動するが、$\Upsilon(4S)$ 上では B 中間子対以外の粒子の生成がない為 B^0 と $\overline{B^0}$ の間に厳密にボーズ・アインシュタイン統計が成り立つので、$\Upsilon(4S)$ から生じた B 中間子対の場合、一方が B の瞬間はもう一方が必ず $\overline{B}$ になっている。そのため、一方が崩壊した瞬間にもう一方が B、$\overline{B}$ のどちらであったかということも判る。そこで、この瞬間を $t = 0$ として、崩壊確率を測定する。

$t = 0$ で 100% B^0 状態であったものが、時間とともに変化して、時刻 t において B^0 である確率と $\overline{B^0}$ である確率はそれぞれ、

$$I(B^0) = \frac{1}{4}[e^{-\Gamma_1 t} + e^{-\Gamma_2 t} + 2e^{-\Gamma t} cos\Delta m t]$$

$$I(\overline{B^0}) = \frac{1}{4}[e^{-\Gamma_1 t} + e^{-\Gamma_2 t} - 2e^{-\Gamma t} cos\Delta m t]$$

ここで、$\Gamma = (\Gamma_1 + \Gamma_2)/2$ とおいている。これら二式を積分して比較する。

$$N(B^0) = \int_0^\infty I(B^0)dt = \frac{1}{4}\left[\frac{1}{\Gamma_1} + \frac{1}{\Gamma_2} + \frac{2\Gamma}{\Gamma^2 + (\Delta m)^2}\right]$$

$$N(\overline{B^0}) = \int_0^\infty I(\overline{B^0})dt = \frac{1}{4}\left[\frac{1}{\Gamma_1} + \frac{1}{\Gamma_2} + \frac{2\Gamma}{\Gamma^2 - (\Delta m)^2}\right]$$

ここで、B^0 と $\overline{B^0}$ の数の比をとると、

$$r = \frac{N(\overline{B^0})}{N(B^0)} = \frac{(\Delta m)^2 + \left(\frac{\Delta\Gamma}{2}\right)^2}{2\Gamma^2 + (\Delta m)^2 - \left(\frac{\Delta\Gamma}{2}\right)}$$

となる。標準理論が示唆するように、$\Delta\Gamma \sim 0$ をとり、質量差 Δm と平均の崩壊幅 $\Gamma = (\Gamma_1 + \Gamma_2)/2$ の比を

$$x = \frac{\Delta m}{\Gamma}$$

とおくと、パラメータ r は簡単な形で書けるようになり、

$$r = \frac{x^2}{2 + x^2}$$

と表される。$r = 0$ で B^0–$\overline{B^0}$ の混合はなく、$r = 1$ で完全混合となる。

B^0 と $\overline{B^0}$ は、それぞれボトム数 +1 と -1 を持つ。B^0 と $\overline{B^0}$ 混合は、ボトム数を 2 だけ変える、高次の弱い相互作用において起こる。b クォークと s クォークからなる B_s 中間子も同様に B_s–$\overline{B_s}$ 混合をおこすので、これらをまとめてダイアグラムを図 1.3 に示す。

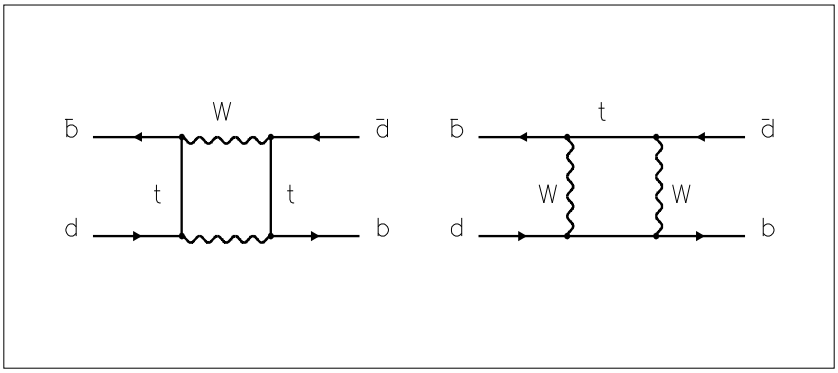


図 1.3: $B\overline{B}$ 混合のボックスダイアグラム

B^0–$\overline{B^0}$ 混合の x を x_d、B_s–$\overline{B_s}$ のそれを x_s とおくと、標準理論では、図 1.3 のグラフを計算して、

$$x_d \propto |V_{td}^* V_{tb}|^2$$

$$x_s \propto |V_{ts}^* V_{tb}|^2$$

であることがわかる。したがって、x_s と x_d を実験で測定し、その比を用いて

$$\frac{x_s}{x_d} = |\frac{V_{ts}}{V_{td}}|^2$$

の関係から、小林・益川の行列要素 V_{ts}/V_{td} を求めることができる。

1.3.3 B 中間子の崩壊における CP の破れ

B 中間子が CP の固有状態に崩壊するとき、$B^0 - \overline{B^0}$ 混合の寄与によって、大きな CP 非対称度が現れることが期待される。

一般に、CP 非保存を観測するには、その反応が二つの振幅を持ち、それらが干渉することが必要である。B が CP の固有状態 f に崩壊する振幅と、B が混合により $\overline{B}$ に変わって f に崩壊する振幅との間に干渉が生じ、そこに CP 非対称度が観測可能な量として現れる。

今、f を CP の固有状態とすると、$\overline{f} = \pm f$ であって、B^0、$\overline{B^0}$ とも f に崩壊できる。B^0 と $\overline{B^0}$ の間には混合があるので、状態は時間とともに振動する。時刻 $t = 0$ で B^0、$\overline{B^0}$ であったものは時刻 t のとき、

$$|B^0(t)> = a_+(t)|B^0> + \eta_B e^{i\phi_M} a_-|\overline{B^0}>$$

$$|\overline{B^0}(t)> = \eta_B^{-1} e^{-i\phi_M} a_-|B^0> + a_+|\overline{B^0}>$$

となる。ここで ϕ_M は混合の位相、$a_\pm$ は $\Delta\mu = \Delta m - i\Delta\Gamma/2$ として

$$a_\pm = e^{-imt} e^{-\frac{\Gamma}{2}t} \cdot \begin{cases} cos\left(\frac{\Delta\mu}{2}t\right) \\ i \cdot sin\left(\frac{\Delta\mu}{2}t\right) \end{cases}$$

である。ただし、$\Delta\mu \sim \Delta m$、$\eta_B = 1$ の近似を用いることができる。

ここで、もう一つパラメータを導入する。

$$\frac{A(\overline{B} \to f)}{A(B \to f)} = \pm\rho e^{i\phi_D}$$

ρ は二つの振幅の比の実数部分で、ϕ_D が位相部分である。

$B^0(\overline{B^0} \to)f$ の崩壊が、ただひとつの振幅に支配される場合、$\rho = 1$ となり、最初 B^0、$\overline{B^0}$ としてつくられた B 中間子は時間とともに

$$\Gamma(B^0(t) \to f) = \Gamma(B^0 \to f)e^{-\Gamma t}1 + \alpha_f sin\Delta mt$$

$$\Gamma(\overline{B^0}(t) \to f) = \Gamma(\overline{B^0} \to f)e^{-\Gamma t}1 - \alpha_f sin\Delta mt$$

と変化する。α_f は CP を破る位相と次のような関係にある。

$$\alpha_f = -(\pm)sin[\phi_M + \phi_D]$$

カッコ内の符号は $\overline{f} = \pm f$ に対応する。

B 中間子の崩壊において、相棒のクォークが何ら影響を与えないとすると、これは単にボトムクォークの自由な崩壊になって、B^0 と $\overline{B^0}$ の崩壊振幅の比は、小林·益川の行列要素を使って、

$$\frac{A(\overline{B^0} \to f)}{A(B^0) \to f} = (\pm)\frac{V_{q'b}V^*_{q'q}}{V^*_{q'b}V_{q'q}}$$

と表される。ここで、$q\prime = (u, c)$、$q = (d, s)$ である。ヴォルフェンシュタインの表示では、$V_{q'q}$ には位相がないので、$V_{ub} = |V_{ub}|e^{-i\delta}$ とすると、ϕ_D はボトムクォークの崩壊モードにしたがって、

$$\phi_D = \begin{cases} -2\delta & b \to u\text{変換の場合} \\ 0 & b \to c\text{変換の場合} \end{cases}$$

となることがわかる。混合の位相 ϕ_M は、上述のボックスダイアグラム (1.3) でトップクォークがもっとも大きな寄与を与えることから、$V_{td} = |V_{td}|e^{-i\phi}$ として、

$$\phi_M = \begin{cases} -2\phi & B_d\text{中間子の場合} \\ 0 & B_s\text{中間子の場合} \end{cases}$$

となることがわかる。これで求まる角度 ϕ、δ は、ユニタリティ三角形の ϕ_1、ϕ_2 に対応するものである。

1.3.4 時間測定の方法

B 中間子の寿命は約 2ps と非常に短いため、直接 B 中間子上の時間を測定するのは不可能である。そこで B 中間子が運動しているときに崩壊点の位置を測定することによって、B 中間子上の時間を測定する。

一方、$\Upsilon(4S)$ は B 中間子対が発生し得るしきい値よりもわずかに高い質量しか持っていないために、$\Upsilon(4S)$ から生成した B 中間子は $\Upsilon(4S)$ に対しほぼ静止している。この場合、B 中間子が生成から崩壊までに進む距離は $20\mu m$ 程度にすぎず、これを上回る位置分解能で崩壊点測定を行なう検出器を用意するのは困難である。

そこで、電子・陽電子衝突によって $\Upsilon(4S)$ を生成するとき、電子ビームと陽電子ビームのエネルギーを異なる値に設定してやると、実験室系で $\Upsilon(4S)$ がビーム軸方向に運動する。これによって B 中間子もビーム軸方向に運動することとなり、ローレンツブーストによって実験室系での寿命が延びる効果も得られるので、既存の検出器技術の位置分解能で B 中間子上の時間を充分な精度で得られる程度に B 中間子の飛行距離を伸ばすことが出来る。例えば電子ビームと陽電子ビームのエネルギーをそれぞれ 8GeV と 3.5GeV に設定すれば、B 中間子が崩壊するまでの飛行距離を $150\mu m$ 程度まで伸ばすことが出来る。

以上をまとめると、$\Upsilon(4S)$ を用いた B 中間子の CP 非保存の研究には、非対称エネルギーによる電子・陽電子衝突によって $10^7 \sim 10^8$ におよぶ大量の B 中間子を発生させ、その崩壊事象を効率良く検出することが必要となる。

そこで次章では、この条件を実現するための実験施設である B ファクトリーについて詳しく説明する。

第 2 章

B ファクトリー

b クォークの研究は、今日の素粒子物理学の最も重要なテーマの一つであり、第一章で述べた CP の破れの観測と小林・益川理論の検証を行なううえで非常に重要である。そこで、b クォークを含む中間子である B 中間子を大量に生成し、その崩壊過程を観測するために、大強度電子・陽電子衝突型加速器と終状態の粒子の諸々の物理量を高分解能で観測する汎用測定器を組み合わせた実験施設が提案された。これを、B ファクトリー（「B の工場」）と呼ぶ。

1994 年 4 月、茨城県つくば市にある高エネルギー加速器研究機構（KEK）において建設が開始された B ファクトリーは、現在その準備の最終段階に入っており、1999 年 4 月に実験を開始する予定である。

この加速器実験施設 B ファクトリーにおいて、高強度の電子・陽電子ビームを衝突させる加速器を KEKB 加速器と呼び、そこから生じた粒子の検出を行なう測定器を、BELLE 測定器と呼ぶ。この章では、KEKB 加速器、BELLE 検出器について詳しく述べる [6]。

2.1 KEKB 加速器

KEKB 加速器は、8GeV の電子と 3.5GeV の陽電子を衝突させ、B 中間子と反 B 中間子の対を大量に造り出すことを目的とする、非対称エネルギー・2 リング型の電子陽電子衝突型加速器である。

またビーム強度の尺度であるルミノシティは、$10^{34}cm^{-2}s^{-1}$ を目標に設計されており、これは達成すれば世界最高の数値となる。

以下の小節では非対称エネルギーかつ大強度の加速器でなくてはならない理由とその達成方法について述べる。

2.1.1 電子・陽電子ビームのエネルギー

これまでの電子・陽電子衝突実験が等しいエネルギーの電子と陽電子を衝突させる対称エネルギーの 1 リング型衝突型加速器であったのに対し、KEKB 加速器は電子と陽電子のエネルギーが異なる非対称エネルギー・2 リング型の衝突型加速器であることを特徴とする。

電子・陽電子衝突の重心系エネルギーを上げていくと、相当するしきい値を越える度に新たなクォーク対生成が始まる。しきい値から更にエネルギーを増加させていくと、この新たなクォーク対からなる共鳴状態がいくつか観測され、そこでは事象の発生頻度が非常に高くなる。これらの共鳴状態は運動学的に許されれば新たに生じた重いクォークを含んだ中間子の対に崩壊する。この重心系エネルギーを 10.58GeV のピークに合わせると、b と $\overline{b}$ クォークの 4 番目の共鳴状態 $\Upsilon(4S)$ が出来、これはほぼ 100% の割合で、B 中間子・反 B 中間子の対（$B\overline{B}$ 対）に崩壊する。

前章で既に述べたように、崩壊点の位置を測定することによって B 中間子が崩壊するまでの時間を計るには、電子と陽電子のビームのエネルギーを違えて、非対称衝突させねばならない。双方のビームのエネルギーの差が大きいほど B 中間子上の時間測定の精度を上げることが出来るが、このビームのエネルギーの非対称度を大きくしすぎると大強度のビーム衝突を得るのは難しくなる。

そこで、重心系エネルギーを 10.58GeV とし、ビーム強度を損なわぬ範囲で B 中間子上の時間が充分な精度で測定可能となるエネルギーとして、KEKB 加速器では電子と陽電子ビームのエネルギーを各々 8GeV・3.5GeV とする。（重心系でのエネルギーは $E^* = 2\sqrt{E_{e^-}E_{e^+}}$ で与えられる。）エネルギーの高い方のリングに電子を蓄積する理由は、電子の蓄積リングで問題となるイオントラッピング現象（残留ガスがイオン化されて生じたイオンが電子の軌道の周辺に捕捉されることにより、電子ビームの運動が撹乱される現象）がエネルギーの低いほど深刻であるためである。

エネルギーの等しい電子と陽電子は加速器の中で同一の軌道を反対方向に周回するため、トリスタンの様な対称エネルギー型の衝突型加速器においては一つのリングを用意すればよい。しかし KEKB のような非対称エネルギー型の衝突型加速器では、電子と陽電子は異なったリング中に蓄積されなければならず、2 リングが必要となる。

実際には、既存の直径 1km· 周長 3km のトリスタンのトンネルの中に、電子を蓄積するリングと陽電子を蓄積するリングの二つを並べて設置する。効率を上げるた

めに、線形加速器で最終エネルギーまで加速された電子と陽電子は中間加速器を経由せずに直接主リングに入射され、それぞれのリングの中を反対方向に周回する。最終的には各々のリングが、$60cm$ 間隔で走る約 5000 個のバンチ（電子や陽電子が数千億個集まって出来た塊）を蓄積する予定である。二つのリングは 2 ヶ所で交差するが、そのうちの 1 ヶ所は衝突点で、後述する BELLE 測定器が置かれる場所である。衝突点ではバンチを強く収束して交差させる。もう一方の交差点では、リングを上下にすれ違わせ衝突を起こさせないようにする。

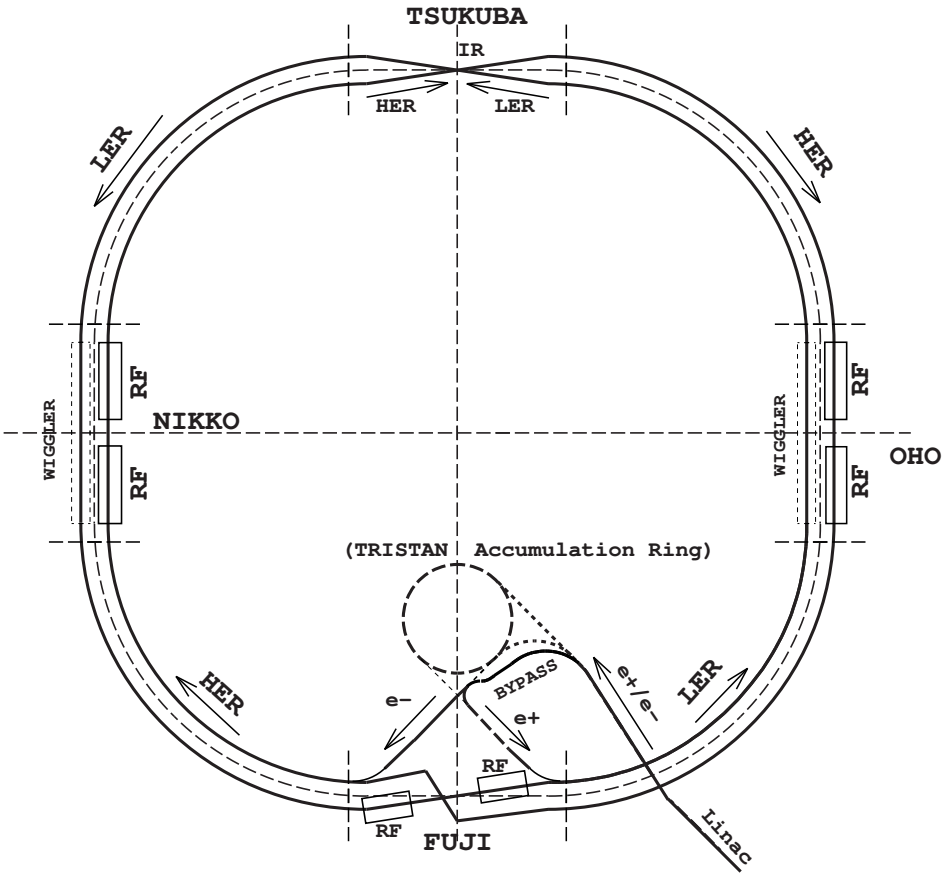


図 2.1: KEKB 加速器

2.1.2 要求されるルミノシティ

衝突型加速器のビーム強度の指標となるのはルミノシティとよばれる量である。KEKB で目標としている $10^{34}cm^{-2}s^{-1}$ というルミノシティの値は、トリスタンの 250 倍、現在世界最高である CESR の 50 倍であり、具体的には毎秒 12 対の $B\bar{B}$ 対を生成するビーム強度である。ルミノシティ $\mathcal{L}$ は、断面積 σ を持つ反応の発生頻度 R との間に $R = \mathcal{L}\sigma$ となる関係にある。B 中間子が CP 非対称度の測定に使える特定の

終状態に崩壊する確率は非常に低く、$10^{-4} \sim 10^{-5}$ 程度であるので、必要な事象の数を十分得るには大量の B 中間子対（約 10^8 個）が必要である。そのため加速器は非常に大きな値のルミノシティを達成せねばならない。

衝突型加速器において、ルミノシティは次式で与えられる。

$$\mathcal{L} = 2.2 \times 10^{34} \xi (1 + r) \left(\frac{E \cdot I}{\beta_y^*} \right)_{\pm}$$

ここで、

E : ビームのエネルギー [GeV]
I : 蓄積電流 [A]
ξ : ヒームビームチューンシフト (衝突時に働く、ビームビーム力の強さ)
r : 衝突点における垂直方向のビームサイズを水平方向のビームサイズで割った値
β_y^* : 衝突点で垂直方向にどれだけビームを絞るかを表すパラメータ [cm]

である。$\pm$ はこの式が電子にも陽電子にも成り立つことを示す。

電子リングにおいてはビームは非常に偏平であり、r の値は $0.01 - 0.03$ と小さく無視してよい。よってルミノシティ L を大きくするには ξ と電流 I を大きくし、ビームサイズ β_y^* を小さくする必要がある。KEKB では ξ を 0.05 と比較的大きな値を仮定し、かつ β_y^* を $1cm$ まで小さくするが、それでも目指すルミノシティに対して必要な電流は、電子リングでは $1.1A$、陽電子リングでは $2.6A$ という大電流となる。一つのバンチが担うことの出来る電流は 数mA 程度なので、このような大きな電流値を得るには非常に多くのバンチを蓄積しなければならない。KEKB では、最終的に各リングに 5000 個のバンチを蓄積させる。

2.2 BELLE 測定器

BELLE 測定器では、電子・陽電子の衝突によって生まれた様々な粒子の崩壊点および粒子の種類を測定する。効率良く CP の破れを検出するために、測定器には以下のことが必要とされる。

1. B 中間子の崩壊点（バーテックス）を少なくとも平均崩壊長の 2 分の 1 よりも良い精度で測定すること

2. $\pi^{\pm}$、$K^{\pm}$、$e^{\pm}$、$\mu^{\pm}$ といった多岐に及ぶ終状態の荷電粒子の運動量を少なくとも 0.5 %の分解能で測定し、かつその種類を正しく判別する識別能力を持つこと

3. π^0、γ を伴う崩壊をした B 中間子を再構成するための高いエネルギー分解能と良好な位置分解能をもったカロリメータを持つこと

4. 効率良く興味ある事象を選別して取り込むトリガーと高速のデータ収集システムを持つこと

衝突ビーム型の実験、とりわけ B ファクトリーの精密実験では、あらゆる種類の粒子を検出できる「汎用型測定器」が要求される。BELLE 測定器では、それぞれの目的を担った検出器が層構造になって、全体の検出器システムをなしている。

それぞれの目的とそれを果たす検出器を、ビームパイプに近いものから順に表 2.1 に示す [7]。

表 2.1: 各検出器の目的

目的	検出器名
B 中間子崩壊点の検出	シリコンバーテックス検出器 (SVD)
ルミノシティモニター	超前方カロリメータ (EFC)
荷電粒子の飛跡測定	中央ドリフトチェンバー (CDC)
粒子識別	中央ドリフトチェンバー (CDC) エアロジェルチェレンコフカウンター (ACC) 飛行時間測定器 (TOF/TSC)
電磁シャワーによるエネルギー測定	CsI 電磁カロリメータ (ECL)
K_L 粒子と μ 粒子の検出	K_L,μ 検出器 (KLM)

BELLE 検出器の全体図を図 2.2、2.3 に示す。非対称エネルギーでの衝突実験であるので、検出器も非対称に作られる。

以下、各々の検出器について説明する [8]。

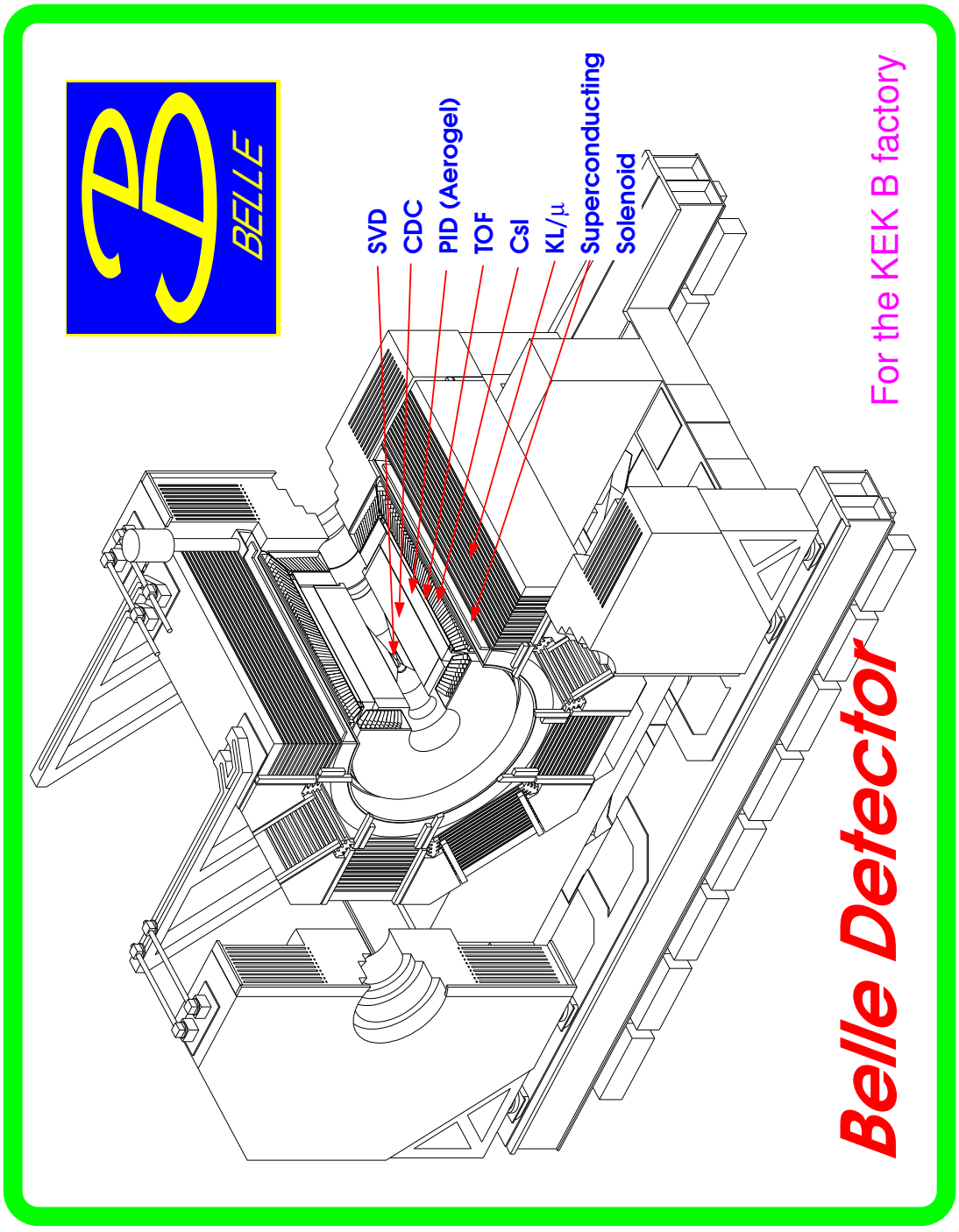


図 2.2: BELLE 検出器

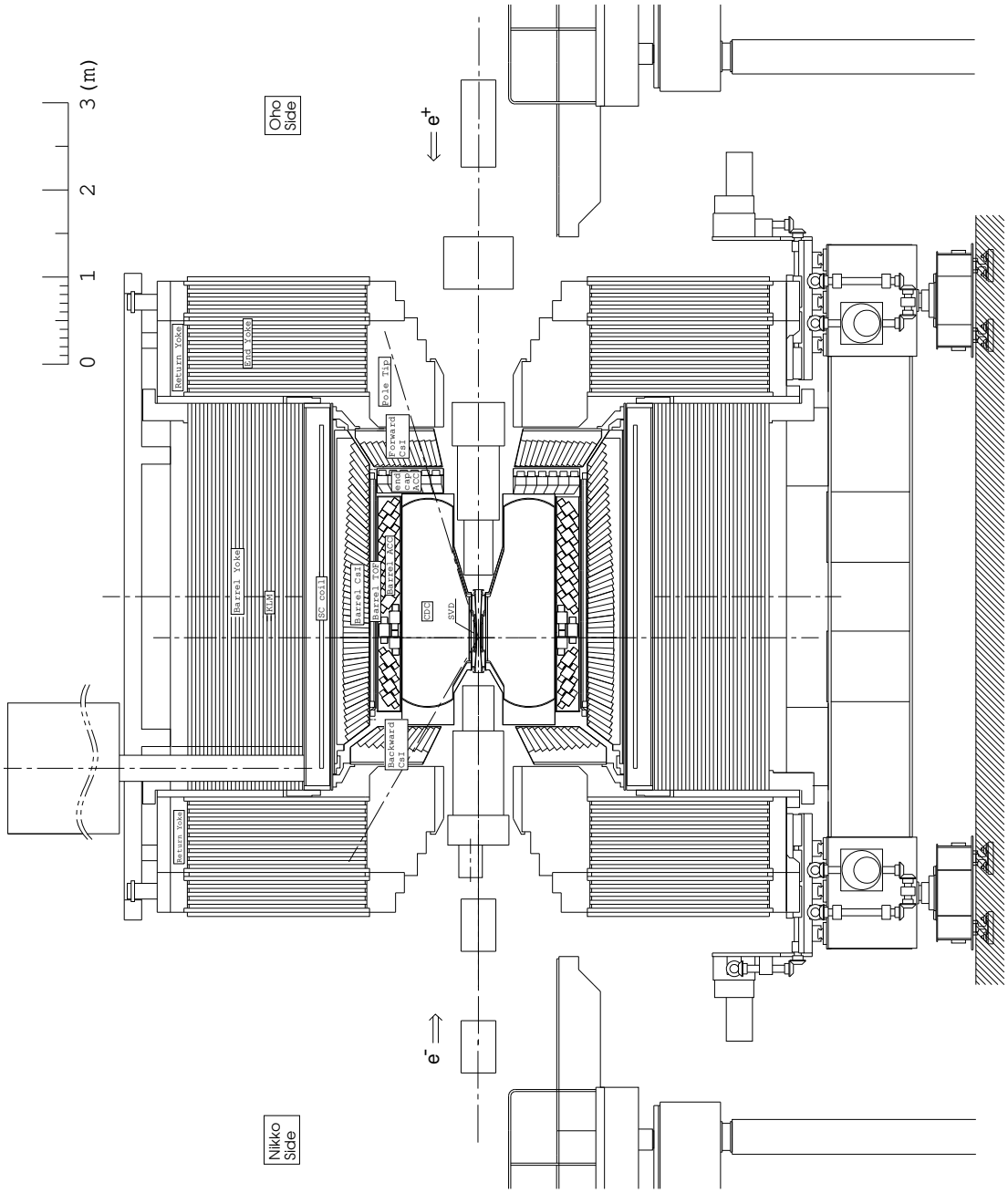


図 2.3: BELLE 検出器側面図

2.2.1 粒子崩壊点測定器 (SVD)

B 中間子の生成点と崩壊点を区別するのに充分な分解能をもち、その距離を測定することが可能な高性能のバーテックス検出器（SVD）は、測定器系全体の心臓部である。KEKB で作られる B 中間子は、崩壊までに約 150 μm 走る。これに対し SVD では B メソンの崩壊点を 80μm の精度で測定可能である。非常に短い距離を精度よく測定する測定器であるため、出来る限りビームパイプ内の反応点に近付けて設置される。測定器の最内側層のビーム軸からの半径は 20mm である。SVD は全部で 3 層から成り、各層の両面にはシリコンマイクロストリップ検出器（DSSD）が内側から順に 8,10,14 枚配置される。シリコンストリップ検出器とは、厚さ 300μm のシリコン板に幅 6μm の電極を 25μm 間隔に貼りつけたものである。この上下面の電極に逆バイアス電圧をかけ、電極間を荷電粒子が通過する際に生じる電子とホールの対を各電極に集め信号として読みだし、飛跡を測定する。

2.2.2 中央飛跡検出器 (CDC)

荷電粒子は、磁場中で運動量に応じた螺旋型の飛跡を描く。その飛跡をドリフトチェンバーで再構成することによって粒子の運動量を測定する。CDC は多線式のドリフトチェンバーであり、その構造は、ガス（He:50%,C_2H_6:50%）中に細い電極線（ワイヤー）を多数張ったものである [9]。荷電粒子が測定器の物質から受ける多重散乱などの影響による運動量分解能の悪化を極力小さくするために、ガスやワイヤーは物質量の小さいものが用いられている。ワイヤーには二種類あり、陽極となるアノードワイヤー（金メッキタングステン製、直径 30μm）と、ドリフト電場形成のための陰極となるフィールドワイヤー（アルミニウム合金製、直径 120μm）とがある。この陽極ワイヤーに正電圧をかけて、荷電粒子が通過する際ガス粒子との反応で生じる電子を集め信号として読み出す。位置分解能は 130μm である。CDC には約 40 層の陽極ワイヤーが張られ、各々の陽極ワイヤーは陰極ワイヤーで囲まれセルと呼ばれる単位を形成する。ϕ 方向に一周分のセルをレイヤーと呼び、レイヤーは z 軸に並行な軸性レイヤーとそれに対して 4° の角度をつけたステレオレイヤーがある。CDC はガス中での電離による荷電粒子のエネルギー損失を測ることで荷電粒子の識別を行う。目標とされる運動量分解能は P_t≥100MeV/c に対し σ_{P_t}/P_t〜$0.5\sqrt{1+(P_t)^2}$% である。ここで P_t は運動量のビーム軸に垂直な成分である。

2.2.3 粒子識別装置 (ACC,TOF)

粒子識別、特に π 中間子と K 中間子の識別を目的とする検出器である。ACC、TOF では粒子の速度を測定し、CDC で得られた運動量と合わせて粒子の質量を求める。ACC はしきい値型チェレンコフ検出器である。荷電粒子が物質中をその物質中での光の伝搬速度を越えて通過する際生じるチェレンコフ光を検出し、粒子の速度を測定する。ラディエーター物質には非常に屈折率の小さい (〜 1.01) エアロジェルが用いられるため、およそ運動量 4GeV/c まで π 中間子と K 中間子の識別が出来る。

低運動量 ($< 1.2GeV/c$) 粒子の識別には TOF （飛行時間差測定器）が適用される。厚さ 5cm、長さ 300cm のプラスチックシンチレーターを用い、反応からの時間差を測る。1.2m の飛行距離に対して、100ps の精度をもつ。また信号はトリガーにも用いられる。

2.2.4 電磁カロリメータ (ECL)

電子、光子のエネルギーを測定する。π^0 の崩壊から生じる光子のエネルギーは約 20MeV 〜 4GeV に及ぶ。また、電子・陽電子の弾性散乱（Bhabha 散乱）においては、電子は最大 8GeV のエネルギーを持つ。このような光子・電子の広いエネルギー領域に対応した測定器である。良いエネルギー分解能を得るためにシンチレーターとして CsI （Tl）結晶を用いる。

奈良女子大学高エネルギー物理学研究室では、この CsI 電磁カロリメータの開発・製作の共同研究に取り組んだ。

2.2.5 K_L,μ 粒子検出器 (K_L, μ)

中性で長寿命の K_L 中間子と μ 粒子を検出する。前者は中性のため飛跡を残さず、かつ電磁シャワーを作らない。また後者は貫通力が高いという特徴を持つ。そのため、この検出器は最も外側に設置されている。

K_L 粒子は、厚さ 5cm の鉄の吸収板（14 層）と厚さ 4cm のカウンター（14 or 15 層）が互いに重なった構造の検出器の比較的内側で、鉄と反応を起こして検出される。μ 粒子は貫通力が高いため、外側にまで貫く飛跡として観測される。

第 3 章

BELLE 測定器のモニターシステム

3.1 測定器モニターシステムの概要

本論文で述べる測定器モニターシステムとは、BELLE 測定器各所の温度・湿度・ガス圧等の測定器環境の状況を連続してモニターするシステムのことである。以下、BELLE 測定器環境のモニターシステムを測定器モニターシステムと呼ぶことにする。測定器環境のデータを測定する目的は次の 2 点である。

(1) 測定器の状況を監視し、測定器の冷却システムの停止による温度上昇などに代表される異常を感知したときには警告信号 (アラーム) を発する。必要ならば緊急停止を行なう。

(2) 測定器自体が正常に動作していたとしても、その特性はこれら「環境」に依存するため測定データの較正に必要である。

このため計 1163 点の環境情報を測定・記録する。

今回、我々はこの BELLE 測定器モニターシステムの製作をおこなった。この章では測定器モニターシステムの仕組みと実装について詳しく述べる。

測定器モニターシステムは、図 3.1のようなシステムになっている。

測定器に設置されたセンサーは、測定器環境のデータを抵抗、電圧の形で出力する。これらは約 50m のケーブルを通じてエレクトロニクスハット内の HP3852A データロギングシステムで数値化される。数値化されたデータは、HP3852A 内の出力バッファーにためられ、GPIB バスを通じて制御計算機である SPARC ボードがそのデータを受信する。SPARC ボードは受信したデータをネットワークファイルシステム (NFS) でマウントしたサーバー計算機のハードディスクに書き込む。それとともに、ネットワークシェアードメモリー (NSM) というソフトウェアを通じて BELLE データ収集システムを構成する他の CPU へ随時読み出したデータを配布できる。

NSM と測定器モニターシステムの相関図を図 3.2に示す。この図からも分かるように、NSM は測定器モニターの他に、エレクトロニクスハットのモニター、SVD モニター、HV コントロール・モニターなどのモニター＆コントロールのシステムと、DAQ（データ収集システム）が互いに情報交換するための TCP ／ IP ベースのネットワークアプリケーションである。NSM についての詳しい説明は後の 3.4.5 にて述べる。

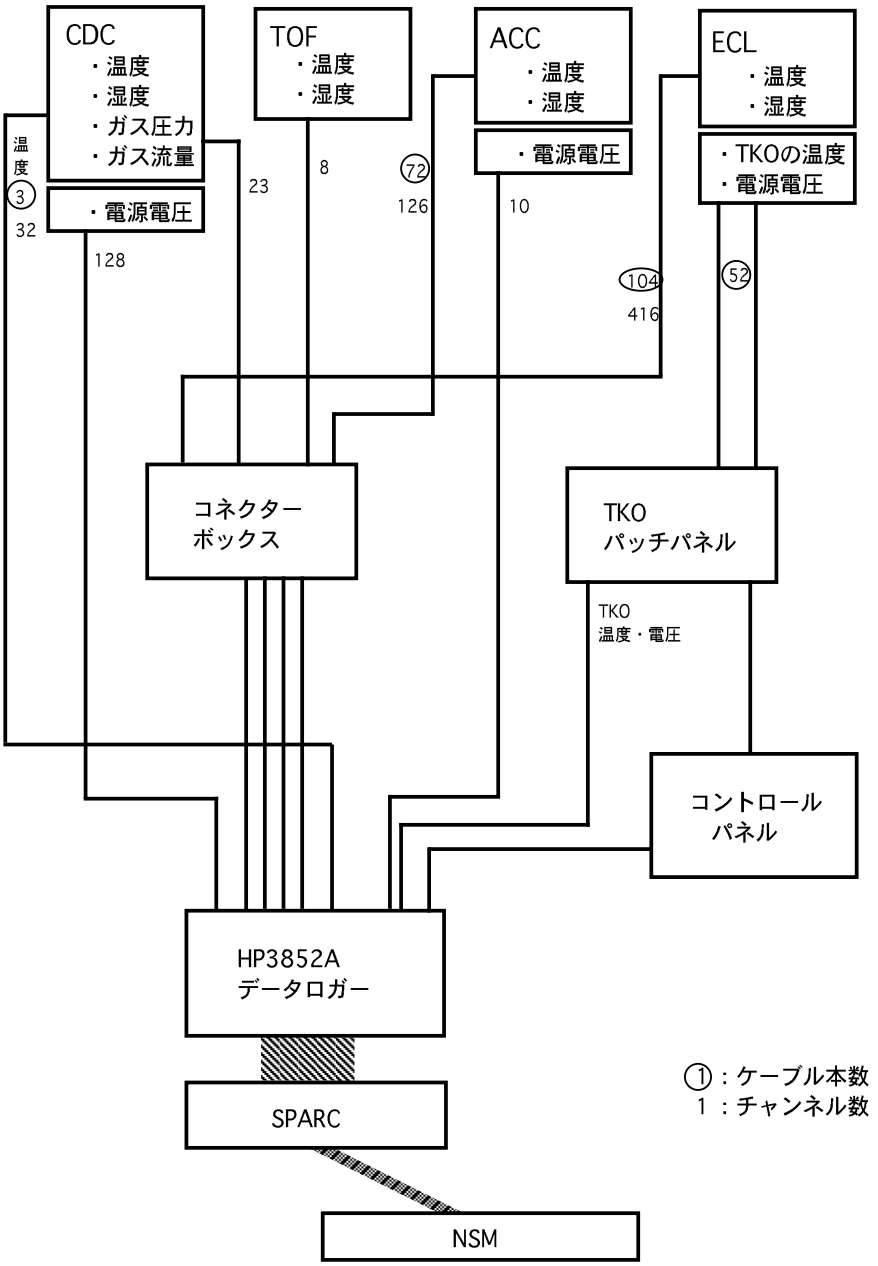


図 3.1: 測定器モニターシステム全体図

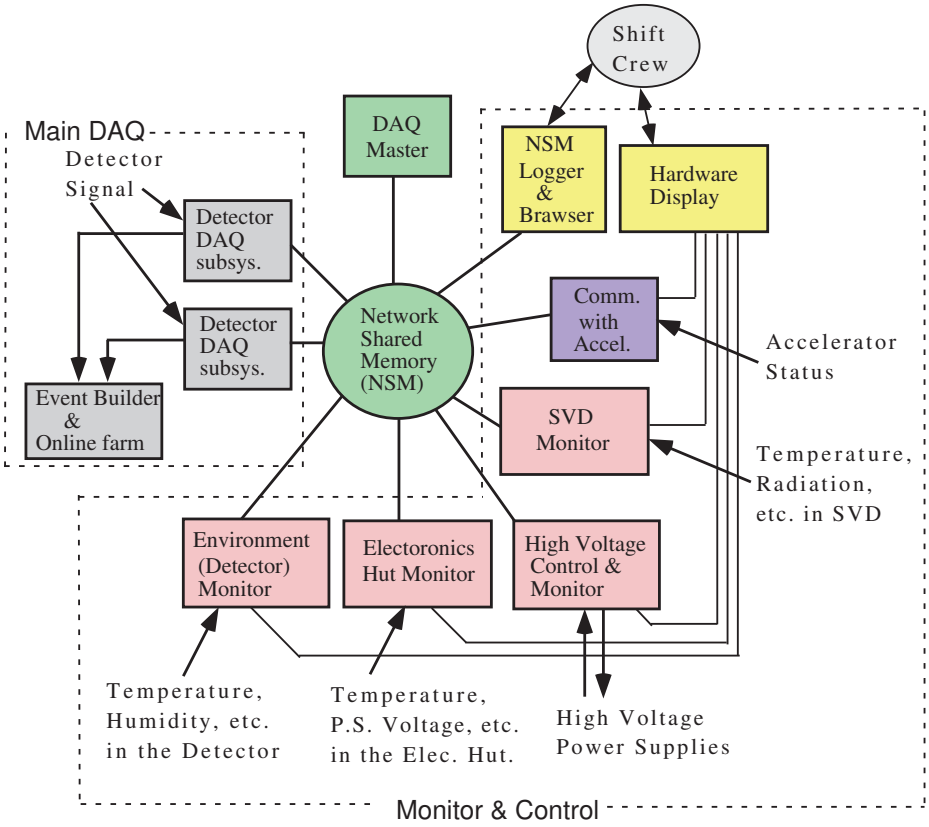


図 3.2: BELLE-DAQ システム全体図

3.2 測定器モニターシステムの構造

モニターの対象となる測定器環境データの一覧を表 3.1に示す。SVD のみ独自のモニターシステムを持っている為、当モニターシステムの対象とはならない。

これらの測定対象は、モニターしたデータの記録を行う温度・湿度・ガス圧力等の「測定器環境」と、データの記録に加えて制御（電源投入・切断の遠隔操作、非常停止）も執り行う「電源電圧」とに大別できる。次節ではこれらそれぞれについて順に述べていく。

3.2.1 測定器環境

以下、測定器環境のデータをモニターするそれぞれのセンサーについて順に述べていく。センサーで感知された信号の出力形態は、温度センサーは抵抗値、湿度センサーは直流電圧である。それ以外にも設置されたセンサーやコントローラは原則として直流電圧の形で信号を出力する。これらは HP3852A データロギングシステムで読み出して数値化される。

温度

温度測定にはサーミスタを使用した。サーミスタは金属酸化物を主原料とした高温で焼結して得られるセラミック半導体であり、その抵抗値が温度によって変化することを利用した温度センサーである。変換公式は

$$R_1 = R_2 \exp\left[B(\frac{1}{T_1} - \frac{1}{T_2})\right] \tag{3.1}$$

T_1,T_2: 絶対温度 (K)

R_1,R_2:T_1,T_2 におけるゼロ負荷抵抗値 (Ω)

B: B 定数 (K)

となっている。ゼロ負荷抵抗値とは、サーミスタを一定の温度で抵抗を測定した時の自己加熱による抵抗値変化を無視できるような十分低い消費電力で測定した場合の抵抗値を示す。また B 定数は抵抗 - 温度特性で任意の 2 点の温度から求めた抵抗値の変化の大きさを表す定数で次の式で表されたものである。

$$B = \frac{\ln R_1 - \ln R_2}{\frac{1}{T_1} - \frac{1}{T_2}}$$

この特性を $\ln R$ と 1/T でグラフ化すると直線で表すことができる。今回使用したセンサーは、石塚電子工業製 103AF2 型で、$25°C$ 時 10kΩ の抵抗値が、B 定数が

サブシステム	測定点	チャンネル数
TOF	パイプ温度	6
	湿度	2
CDC	チェンバー温度	32
	チェンバー周辺湿度	2
	チェンバー内ガス圧力	4
	ボンベ内ガス圧力	4
	バッファータンク内のガス圧力	3
	ガス流量	10
	酸素センサー	1
	水蒸気センサー	1
	エタンセンサー	1
	電源電圧	128
ACC	バレル部の温度	180
	バレル部の湿度	10
	エンドキャップ部の温度	24
	エンドキャップ部の湿度	12
	電源電圧	10
ECL	バレル部の温度	216
	バレル部の湿度	72
	エンドキャップ部の温度	96
	エンドキャップ部の湿度	32
	電源電圧	260
	TKO クレートの 温度	52
エレキハット	冷媒循環器の温度	4
	乾燥空気供給部の湿度	1
	合計	1163

表 3.1: HP3852A で測定される環境データ

3435 のものであり図 3.3 のような形状をしている。

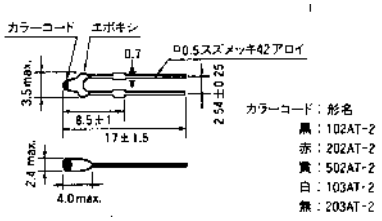


図 3.3: サーミスタ概観図

既に CsI 電磁カロリメータに使用する 372 個のサーミスタについて温度絶対値の較正が行なわれており、その結果 7 個のサーミスタを除いて $\pm$ 0.5°C 以内の精度で測定することができることがわかっている [10]。

湿度

我々が採用した湿度センサーはヴァイサラ社製のヒュミター -50U である。このセンサーは湿度によって多孔質ポリマーでできたセンサー素子の静電容量が変化することを利用しその静電容量を直流電圧に変換して出力する。長さ 51mm のハウジング内にはこれを行なう ASIC （特定用途集積回路）エレクトロニクスが入っている。出力電圧は 0 から 1V でこれが湿度の 0 から 100% に対応し、測定可能領域は 10% から 100% である。図 3.4 に湿度センサーの概観図を示す。既に、塩化リチウム飽和水溶液を入れた密閉された瓶内の湿度が気温の変化にかかわらずほとんど 11.3% を保つことを利用して CsI 電磁カロリメータの 107 個の湿度センサーの湿度の絶対値の較正が行なわれており、その結果全てのセンサーが 11.3% からその差が 1.2% 以内の値を示しており CsI 電磁カロリメータ内の湿度の測定には問題なく使用できることがわかっている [10]。

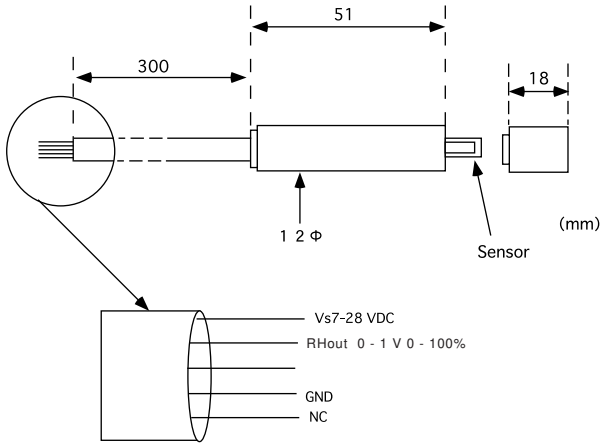


図 3.4: ヒュミター概観図

50m ケーブル取り付け時の湿度センサー動作テスト

BELLE 測定器内部に設置される湿度センサーの信号は、50m もの長いケーブルを介してエレキハット内のデータ集積システムへと運ばれる。よってこのケーブルによる抵抗が測定結果にどのような影響を及ぼすかを事前に調べておく必要がある。方法は塩化リチウム飽和水溶液を入れた密封容器の湿度が $15°C$ から $35°C$ の間 11.3% に保たれることを利用して 50m ケーブルを介した時と介さない時とで信号電圧の変化を調べた。

＜測定方法＞

センサーを図 3.5の様に塩化リチウム飽和水溶液の入った容器に差し込みセンサーの値が安定するまで 1 時間半程度放置し、その後 50m ケーブルを介した場合、介さない場合とでそれぞれ 1 分おきに 5 回電圧を測定し、これを 5 回繰り返して行なった。

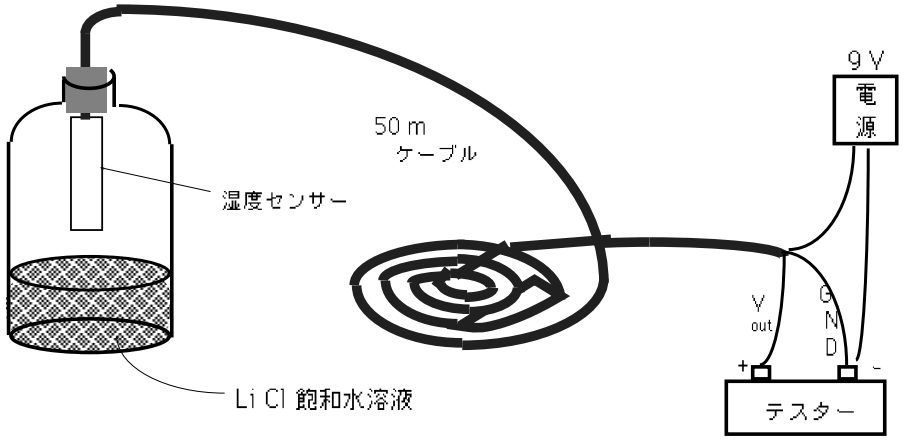


図 3.5: 50m ケーブル取り付け時の湿度センサーのテスト・セットアップ

＜結果＞
表 3.2はそれぞれの結果を示している。
（1V=100% として、% で表記）

これより、50m ケーブルを介した時と介さない時との差は、13.41-12.74=0.67(%) である。一方、ケーブル単独での抵抗実測値は 6.4Ω、ヒュミターの GND 線を流れる電流実測値は〜 1mA であった。これによる電圧降下が温度測定のずれとして現れるがその大きさは $1 \times 10^{-3} \times 6.4 \times 100 = 0.64(\%)$ である。したがってセンサーの実測値で見たずれはケーブルを流れる電流による電圧降下で説明でき、その大きさは高々 0.7 %程度である。よって、50m ケーブル抵抗による湿度測定値のずれは実用上問題ないということが確認された。

回数	ケーブル無し (%)	ケーブル有り (%)
1	13.14	13.51
	13.15	13.43
	13.15	13.32
	13.13	13.25
	13.07	13.31
2	12.77	13.72
	12.85	13.74
	12.91	13.76
	12.96	13.76
	12.99	13.77
3	12.57	13.42
	12.61	13.42
	12.64	13.41
	12.66	13.40
	12.68	13.40
4	12.51	13.38
	12.41	13.19
	12.38	13.18
	12.33	13.19
	12.34	13.21
5	12.61	13.34
	12.65	13.33
	12.68	13.32
	12.70	13.33
	12.70	13.34
平均値	12.74	13.41

表 3.2: 結果

センサーの設置と配線

温度湿度センサーの取り付け位置や採取した信号の流れなどを我々の担当したECLを例に上げて解説する。ACC、CDC、TOFにおいても信号の流れの仕組みは同じである。

CsI電磁カロリメータの外観図を図3.6に示す[9]。カロリメータはバレル部と前方・後方エンドキャップ部から成る。カロリメータが覆うのは電子ビーム軸からの偏角で、17.0° から150.0° の範囲である。総数8736本（総重量約40トン）のCsIカウンターがその縦方向を衝突点に向けて並べられる（図3.7参照）。

バレル部分は円周方向に72区画に分割されており、各々の区画を以下セグメントと呼ぶ。センサー取り付けは、1セグメントにつき温度3点・湿度1点で、この計4点が1つのケーブルにまとめられる。取り付ける場所は、温度センサーを前方、後方、中央部に、湿度センサーを前方に取り付ける。

温度センサーは結晶の吊金具にエポキシで接着し（図3.8）、湿度センサーはケーブル支えに束線バンドで緊縛した。

バレル部分７２本、エンドキャップ部分３２本で計１０４本の長さ50mのケーブルはエレキハット内のコネクターボックスへと接続される。コネクターBOXの図を図3.9に示す。

コネクターボックスはエレキハット内に2つ設置され、1つは主にECL用、もう1つはACC用である。

1セグメント毎だった信号は、コネクターBOXにて温度ごと、湿度ごとにまとめ直される。出力部は60芯コネクタになっており、そこから5mケーブルを介してエレキハット一階のHP3852Aデータロガーに接続される。

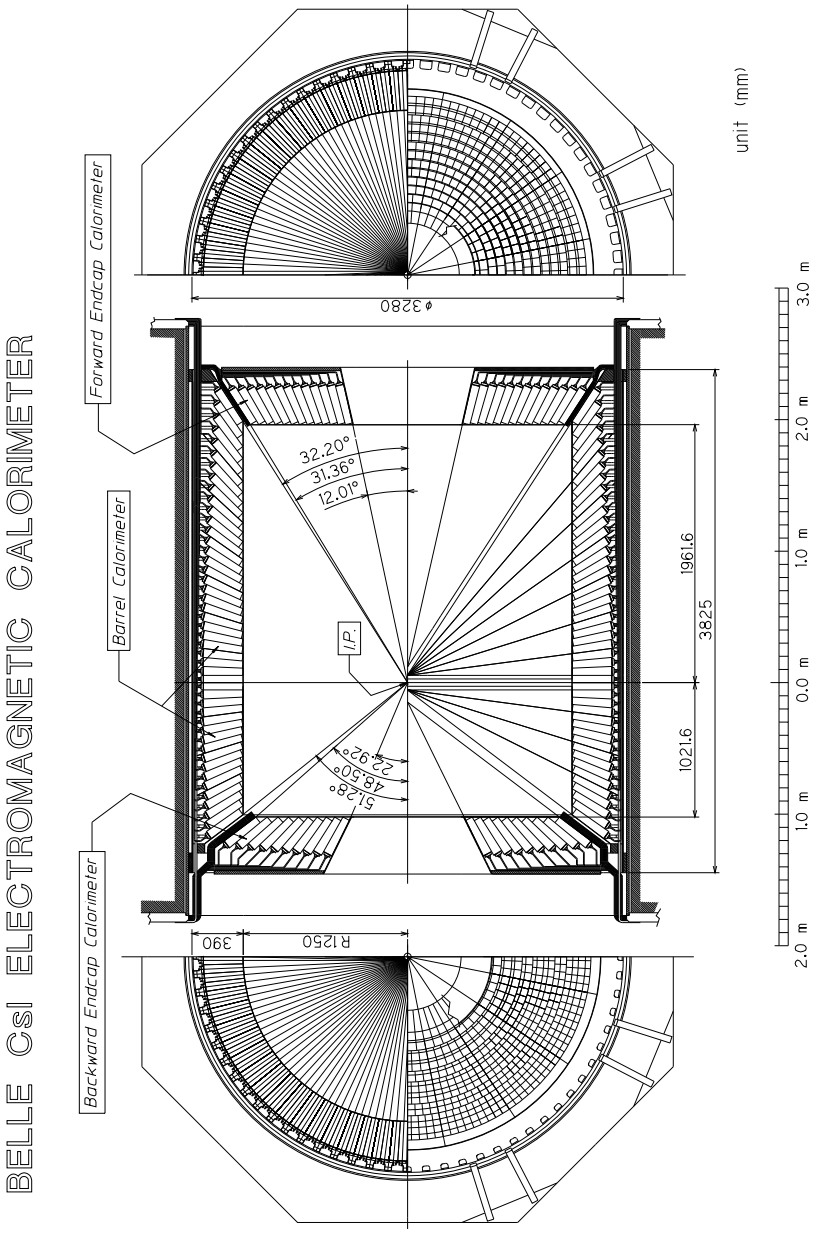


図 3.6: CsI 電磁カロリメータの全体像

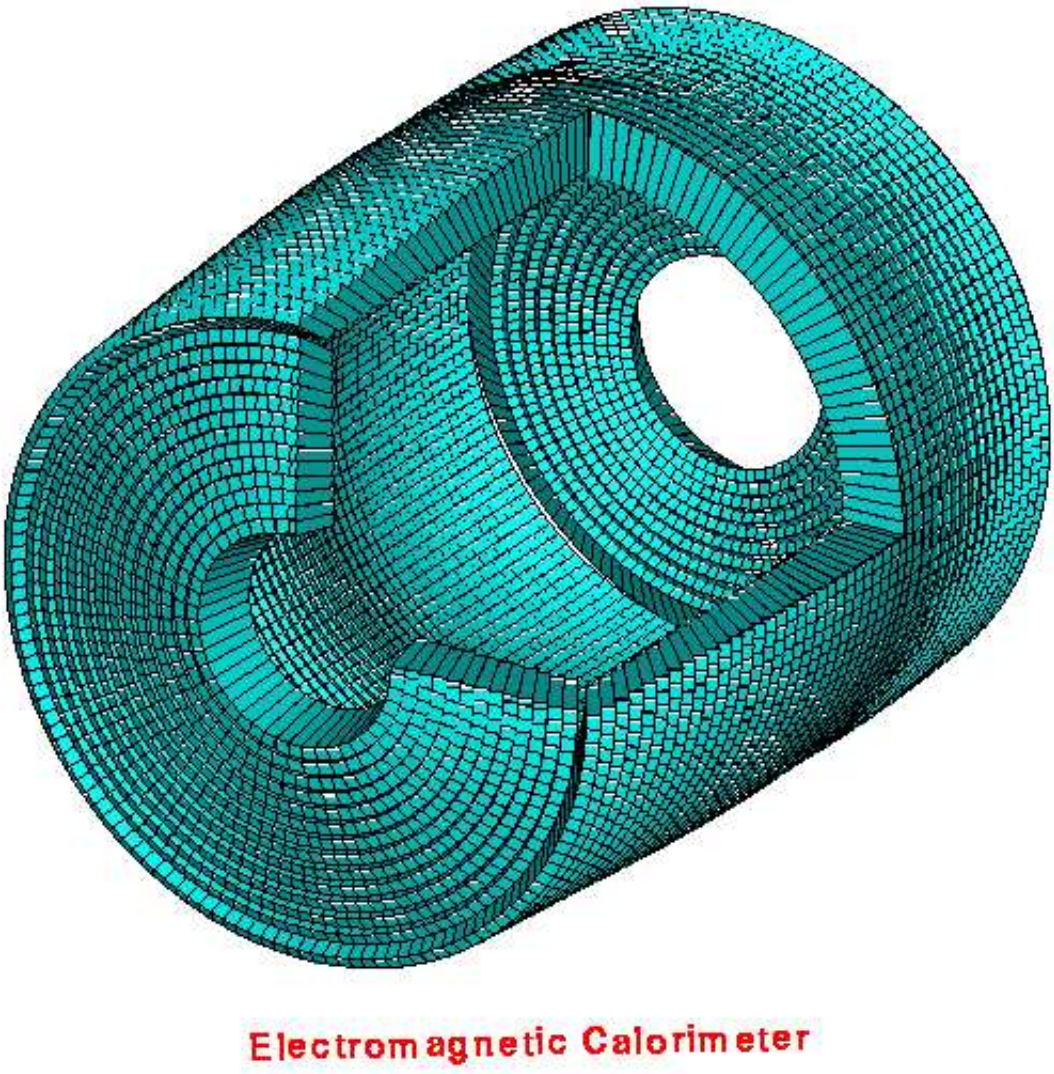


図 3.7: CsI 結晶が並べられた様子

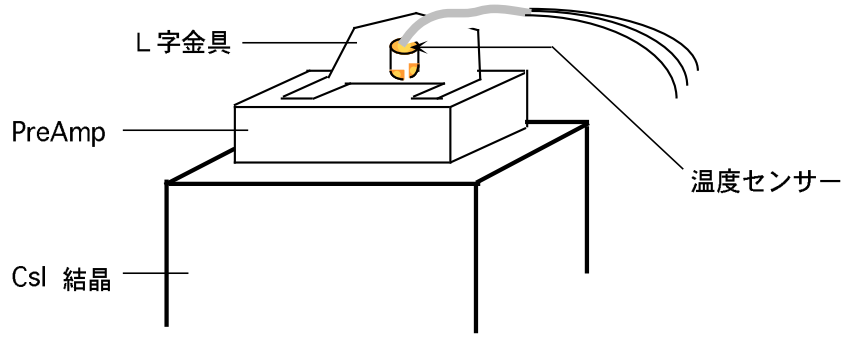


図 3.8: センサー取り付け位置（結晶）

温度・湿度センサー総合動作テスト

各センサーで読みとった信号は、図 3.10のように実に多くの媒体を介してデータロガーまで運ばれる。これら実験実施時の総合的な温度・湿度計測の一連の流れが正常に作動するかどうか、１セット（温度センサー：３、湿度センサー：１）を用いて確認のテストを行なった。

方法

図 3.11のように装置を設置し、各接続点で信号が来ていることを確認した後、HP3852A データロガーにてデータを採取する。

結果 1

湿度センサーにおいて、湿度を上げる為に人の手でセンサー部を握って計測を行なったところ、電圧の値が乱れるという現象が起きた。そこで、息を吹きかける等して直接センサーに触らずに湿度を上げ、計測を試みたところ、電圧の値は正常な上昇を見せた。測定結果を表 3.3に示す。

方法	50m ケーブルの有無	結果
手で握る	有	$20mV \sim 30mV$ （乱れる）
	無	$600mV \sim 700mV$
息を吹きかける	有	$600mV \sim 700mV$
	無	$600mV \sim 700mV$

表 3.3: 結果 1

Connector box

Rearrenging sensor cables ot be suitable the Multiplexer module in Data Logger system.

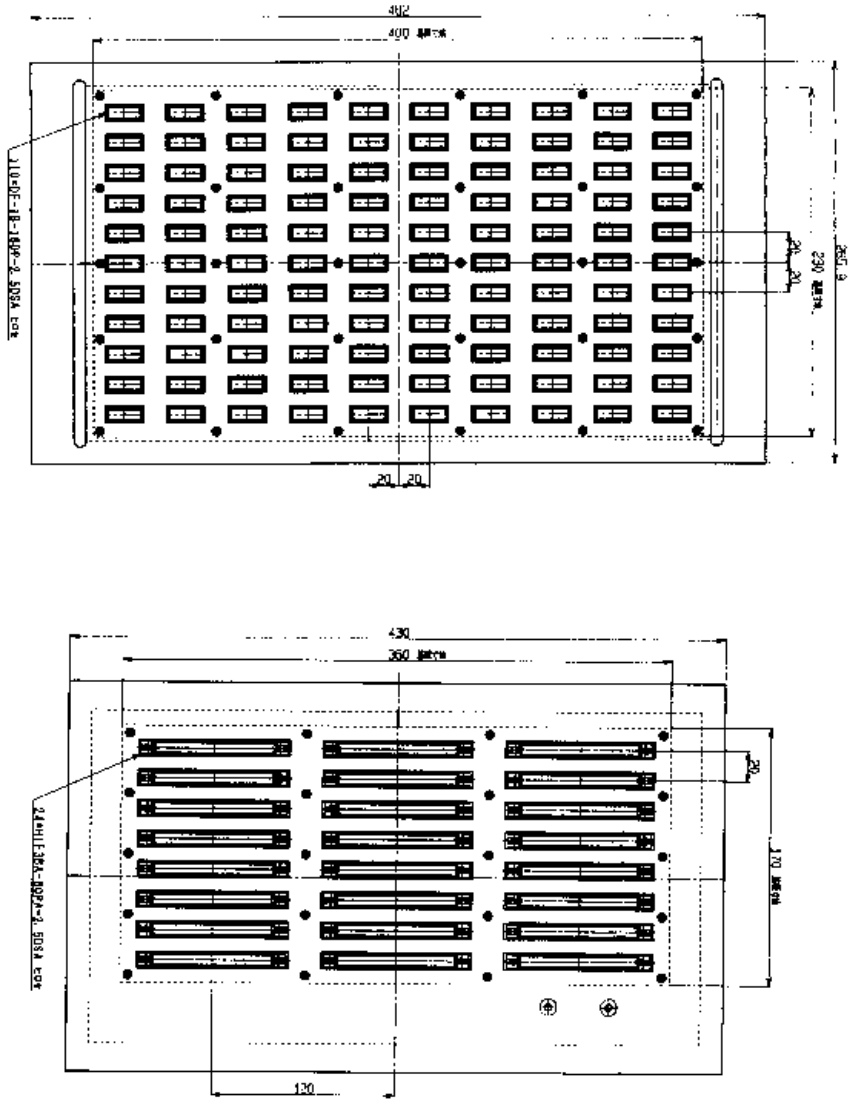


図 3.9: コネクター BOX

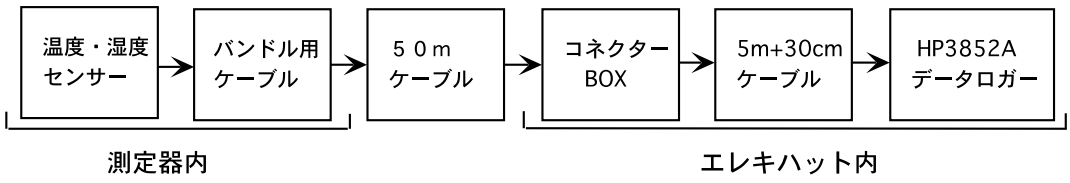


図 3.10: 温度・湿度モニター信号の流れ

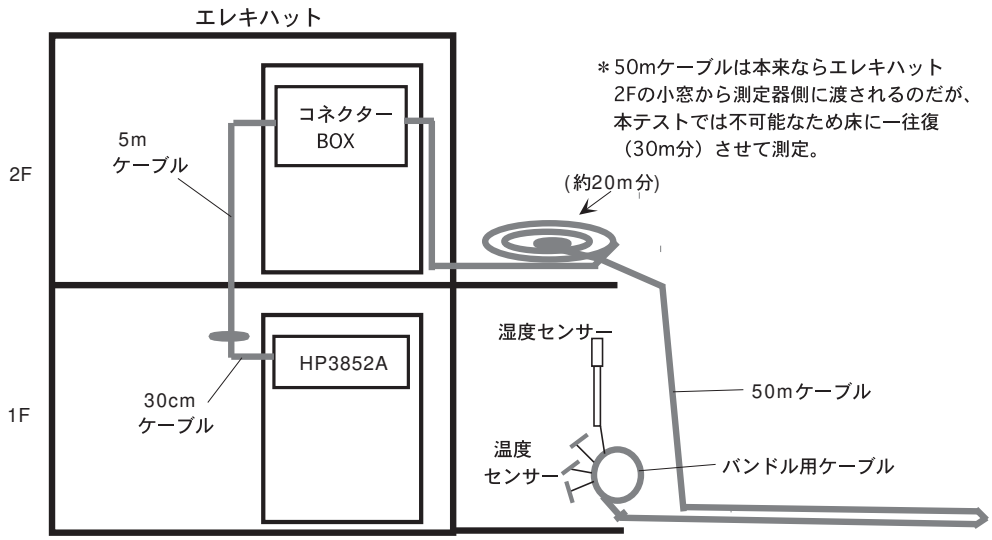


図 3.11: セットアップ

つまり、50m ケーブルを有し、かつセンサー部に接触した場合、異常が発生している。これは人の手でなくとも、測定器の構造体の金属部分等にセンサーが触れたときにも起こり得ることである。

＜原因＞

図 3.12に示すように、異なる二ヶ所で接地している場合、グランドは必ずしも一致しないので、電位差が生じる可能性がある。よってセンサーの外部容器（シールド）に人などが接触することで、50m という長いケーブルを介してグランドループが出来、正確な測定の妨げとなってしまう。

＜解決法＞

図 3.12の $\star$ の部分を切断し、シールド部を浮かせた状態にする。実際には、図 3.13の

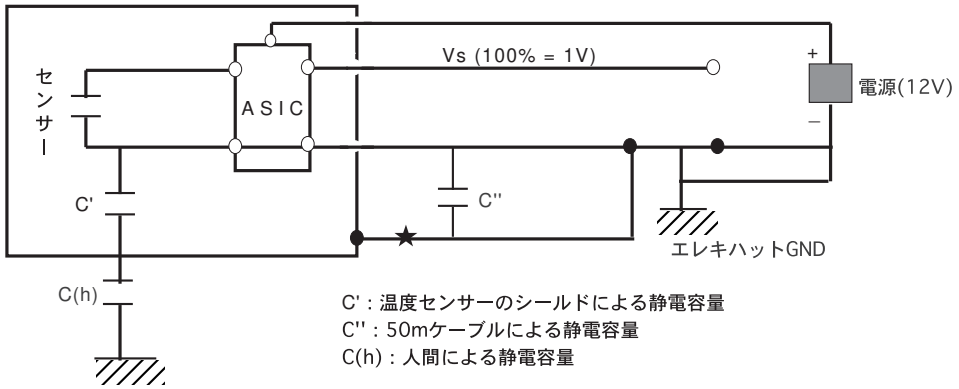


図 3.12: 回路図

ように湿度センサーコネクタ部からシールド線のみを引き抜いた。

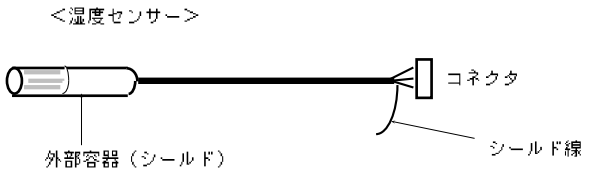


図 3.13:

この結果、再度測定を行なったところ、いずれの場合においても正常な測定値が得られた。

結果 2

HP3852A データロガーにおいて、HP3852A に内蔵されているデジタルボルトメータ (HP44701A) の積分時間（ゲートの長さ）を $200ms$、$20ms$、$10ms$ にしてそれぞれ 1000 回ずつ測定を行ない採取された結果を表 3.4〜3.7に示す。ケーブルがノイズをピックアップする効果のみの大きさを調べる目的で、温度センサー 3 のみ、センサーの代わりに $10K\Omega$ の抵抗を接続した。

温度センサー 1、2 共に約 $10.5K\Omega(= 23.8°C)$、湿度センサーも約 $532mV(= 53.2\%)$ と正常な数値を示している。ゲートの長さの違いでデジタルボルトメータの測定時間、測定精度、ノイズ除去が変化する。一般に、ゲートの長さが短いと測定速度は早いが測定精度が悪くなり、逆にゲートの長さが長いと時間はかかるが精度が良くなるとされているものだが、この結果 2 の温度センサーにおいては一番ゲートの長さが長い $200ms$ のときに標準偏差の値が大きくなっている。しかしいずれにして

ゲートの長さ	$200ms$	$20ms$	$10ms$
最小値 $(K\Omega)$	10.529	10.515	10.517
最大値 $(K\Omega)$	10.562	10.551	10.540
平均値 $(K\Omega)$	10.545	10.542	10.530
標準偏差 (Ω)	10.529	5.1150	5.1631

表 3.4: 温度センサー -1

ゲートの長さ	$200ms$	$20ms$	$10ms$
最小値 $(K\Omega)$	10.453	10.441	10.482
最大値 $(K\Omega)$	10.535	10.478	10.519
平均値 $(K\Omega)$	10.491	10.457	10.497
標準偏差 (Ω)	24.258	8.2122	8.6552

表 3.5: 温度センサー -2

ゲートの長さ	$200ms$	$20ms$	$10ms$
最小値 $(K\Omega)$	10.001	9.9918	9.9944
最大値 $(K\Omega)$	10.008	10.015	10.015
平均値 $(K\Omega)$	10.008	10.008	10.008
標準偏差 (Ω)	22.362	5.0486	5.0529

表 3.6: 温度センサー -3

もそのふらつきは測定値の 0.1% 未満と非常に小さい値であり、測定精度は充分に良いものであるといえる。

結論

50m ケーブルがピックアップするノイズの影響は充分小さく、温度の測定には実用上全く問題がない。湿度の測定に関しては、センサーのシールド線がエレキハットのグランドに接続されているとグランドループを作って正常な測定値を示さなかったので、これを切断したところ解決した。

ゲートの長さ	$200ms$	$20ms$	$10ms$
最小値 (mV)	531.04	530.76	529.50
最大値 (mV)	532.47	534.06	533.05
平均値 (mV)	531.83	532.52	531.53
標準偏差 (V)	2.7294×10^{-4}	4.1754×10^{-4}	4.2457×10^{-4}

表 3.7: 湿度センサー

3.2.2 電源電圧

ECL、ACC、CDC においては読みだしエレクトロニクスの電源の電圧をモニターする。また ECL の TKO 電源については、手動あるいは SPARC ボードあるいは HP データロガーからの遠隔操作による、電源投入・切断といったコントロールも行なう。

ここでは主に、ECL の TKO エレクトロニクスのモニターとコントロールについて述べる。TKO クレートの配置図を図 3.14に示す。図のように 5 2 個の TKO クレートは測定器を囲むように配置され、8 つのステーションに分けられる。 52 個のクレート内には 1 つずつ温度センサーを設置した。取り付け位置については後述のようにテストをおこなって決定した。

また TKO クレートには、+5.2V、-5.4V、+8.0V、-8.0V、+15V、-15V と 6 種類の電圧が供給されるので、これらもモニターする。TKO 電源装置の各々は遠隔制御用の端子を背面に装備しているので、これをもちいて電源投入・切断あるいは非常停止を行なう。これらモニターとコントロールに必要な信号は TKO クレートに一本ずつ、全体の合計 52 本の多芯ケーブルによってエレキハットまで配線され、そこで TKO パッチパネルへ接続される。TKO パッチパネルの設計図を図 3.15に示す。 パネル上半分の D サブコネクタ部が入力部で、TKO クレートからのケーブル 5 2 本はここにつながれる。各ピンごとの情報はパネルの裏側（写真 参照）で、温度・電圧・リモート制御用といった同種の信号ごとにまとめられる。温度・電圧はパネル下半分の 60 芯コネクタに接続し、表から出力され HP データロガーへと運ばれる。アラームの信号のみパネル中間部の 13 芯コネクタへとつながれ、パネル表からの出力は同エレキハット内のコントロールパネル（スイッチパネル）へと接続される。コントロールパネルの設計図を図 3.16に示す。このコントロールパネルの役割は、エレキハットにて TKO クレート各々の電源を手動で投入・切断するこ

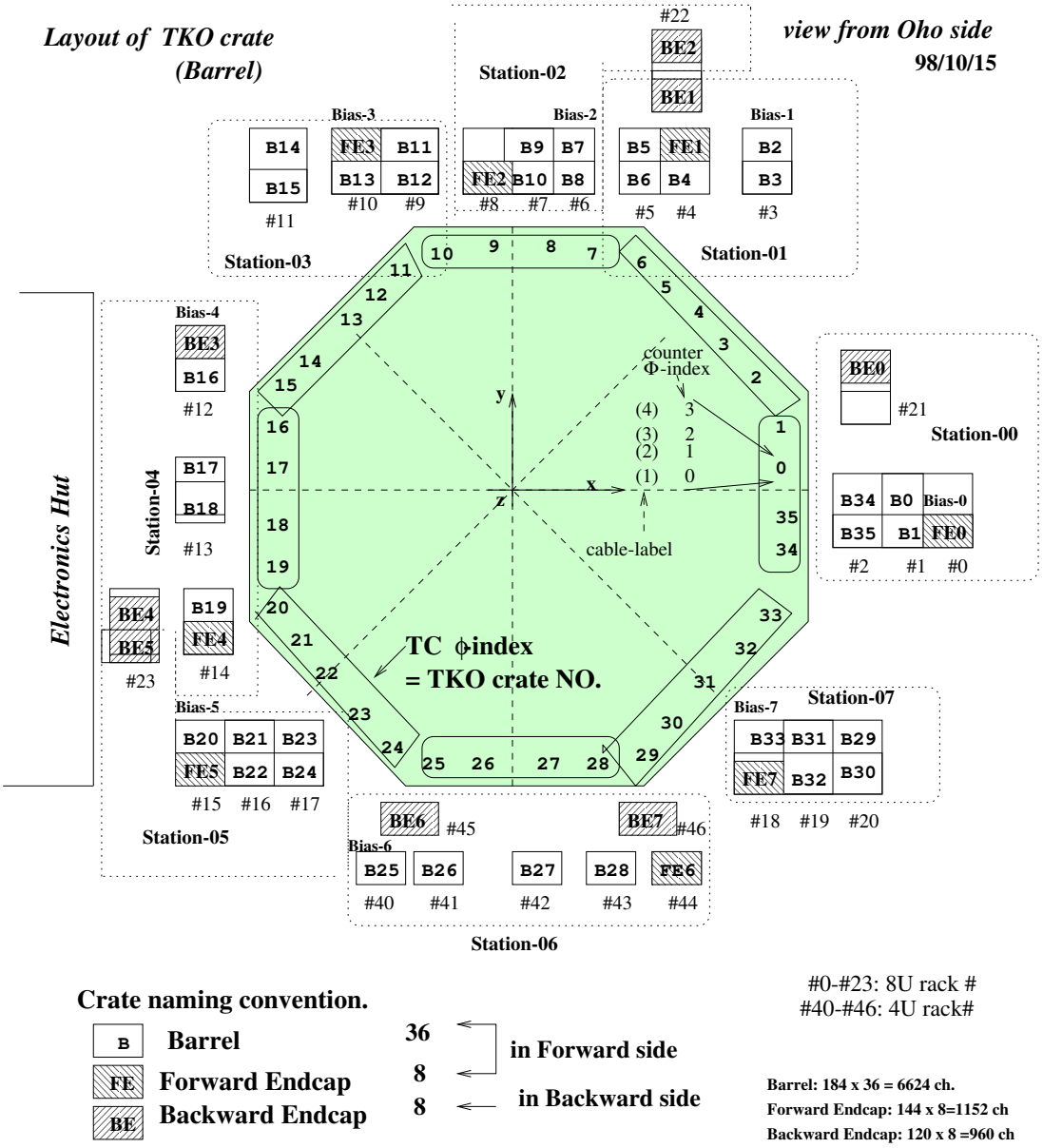


図 3.14: TKO クレート配置図

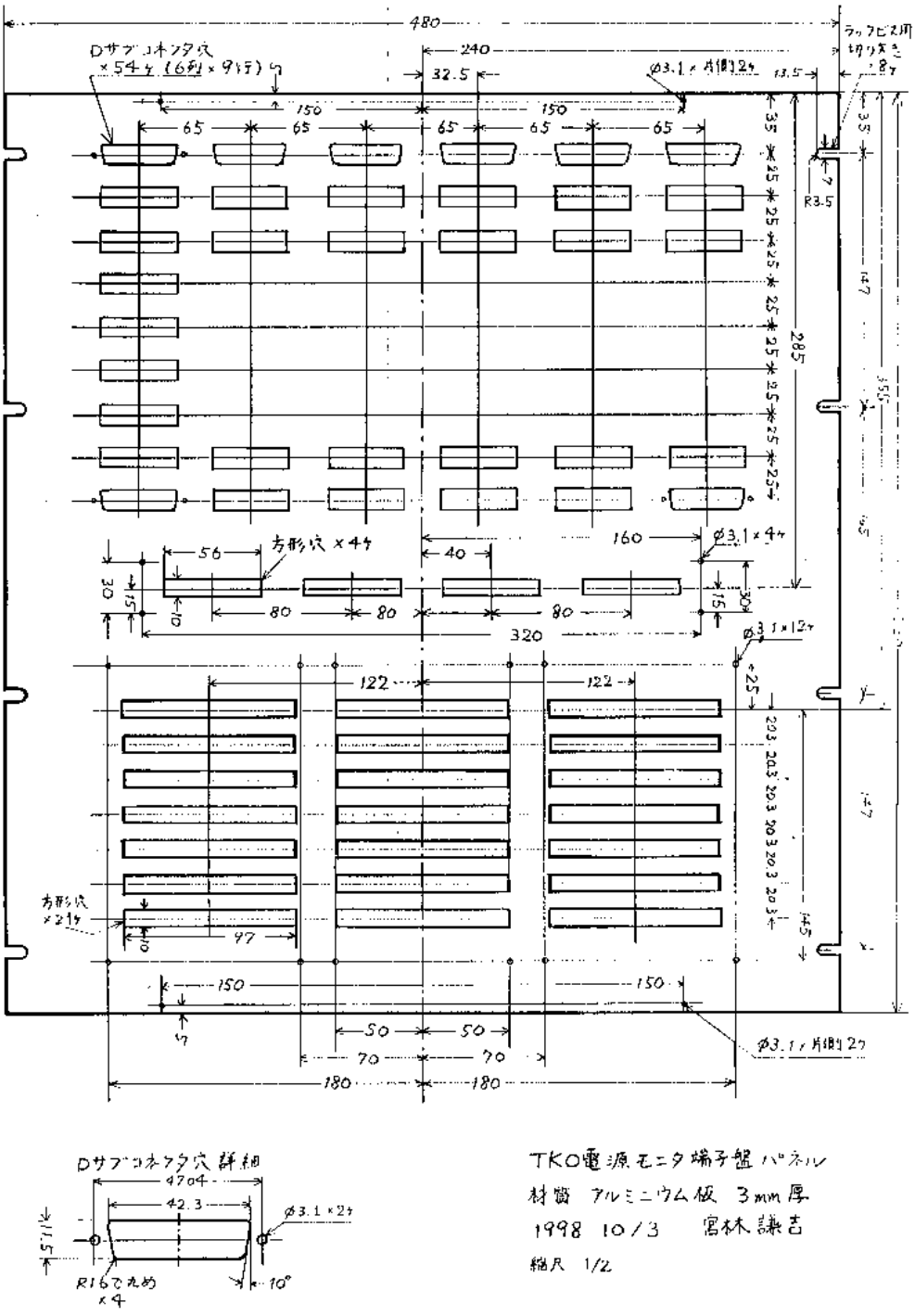


図 3.15: TKO パッチパネル（モニターパネル）

とを可能にするとともに、非常停止のような遠隔制御のための信号を各 TKO クレートに分配することである。 TKO クレート内の温度が著しく高くなった場合は、該当クレートの電源を非常停止するために、SPARC ボードが発するコマンドによって HP3852A が接点を切り替えて電源を切断する（インターロック）。この目的のためにリレーを用いて、後述する HP44725A GP スイッチから供給される 12 系統の接点信号を 52 個の TKO クレートに分配している。

TKO クレート内の温度測定テスト

TKO クレートには多数のモジュールが設置されるため、ファンによる空冷がなければモジュールから発せられる熱により内部は高温になり、電子回路やケーブルの破損につながる。このような状況を検知する目的で、TKO クレート内の温度を測定する。TKO クレート内部に温度センサーを取り付けるにあたり最適な測定点を調べるため、3ヶ所にセンサーを設置して温度の変化を観測した。センサーの設置場所は図 3.17に示したように、モジュール間の空気、シェーパー QT モジュールフロントパネルと、TKO クレート上部である。測定を行なったのは測定器上部にある B09、B10 （Station-2,rack # 7）のクレートである。

測定手順

1. 外気温を測定する
2. ファンをつけたまま、上段（B09）のクレートの電源を入れ、温度上昇が止まって一定の温度になるまで温度を測定する
3. 下段（B10）のクレートの電源を入れ、同様に一定の温度になるまで測定を続ける
4. ファンを消し、最高温度が 50 度になるまで測定したら、クレートの電源を落し、再びファンをつける。
5. 充分に熱が冷めたら、ファンを消し、再度クレートの電源を入れて測定を続ける

結果

図 3.18〜図 3.20は一度クレートの電源を落すまでのそれぞれの温度変化の様子を示している。 35 分後に下段（B10）のクレートの電源を入れ、 75 分後にファンを

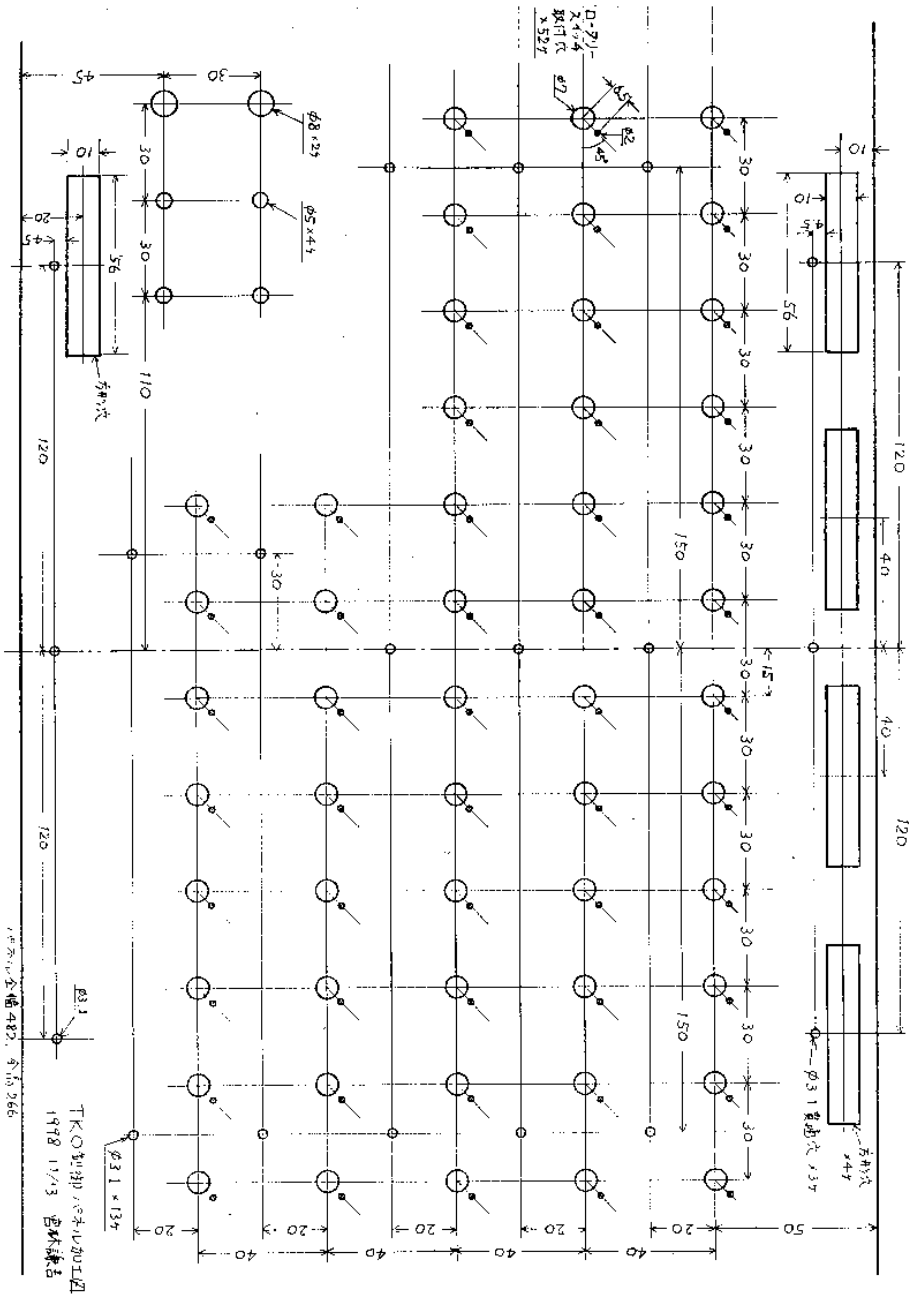


図 3.16: コントロールパネル（スイッチパネル）

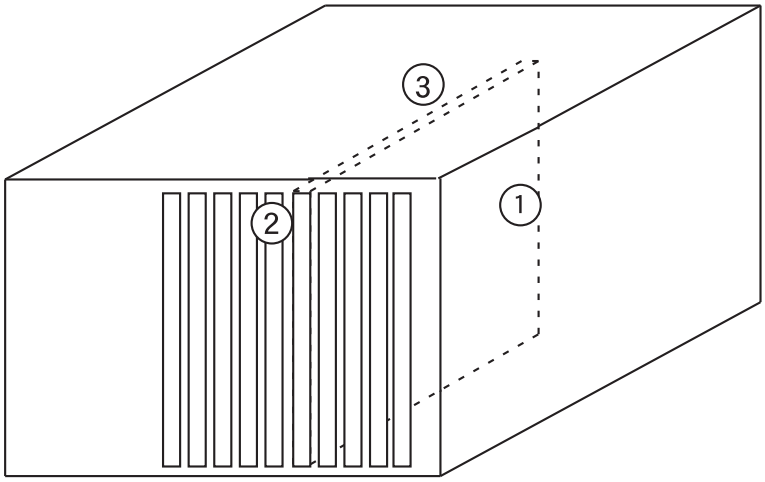


図 3.17: 温度センサー取り付け位置

消した。上段クレートのみ電源を入れている時のサッチレート後の温度は、それぞれモジュール間が $29°C$、フロントパネルが $38°C$、TKO クレート上部内側が $34°C$ である。また、下段クレートの電源を入れることによる影響は、$+\ 2°C$ 以内である。

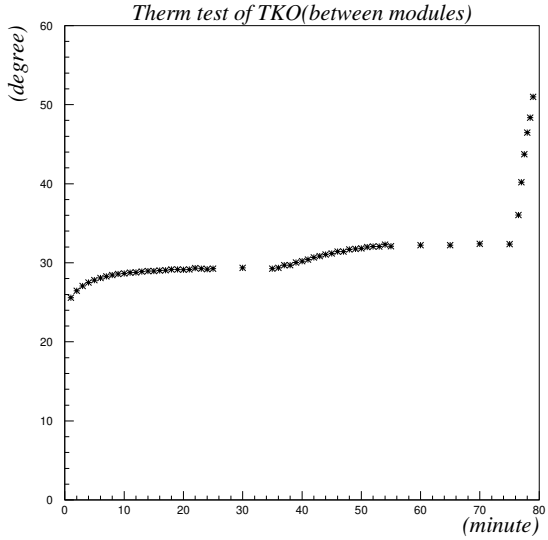


図 3.18: TKO クレート内部温度変化の様子（モジュール間）

図 3.18〜図 3.20からも分かるように、ファンを消したときに急激に温度が上がるのはモジュール間のセンサーのみとなった。その他 2 点にて温度が下がっているのは、ファンを止めたことにより熱源からの温風が届かなくなったためと思われる。また、表 3.8に示すように、ファンを消したままクレートの電源を入れた場合にも、モジュール間のみわずか 5 分間で $25°C$ から $40°C$ と急激な温度上昇が起こった。

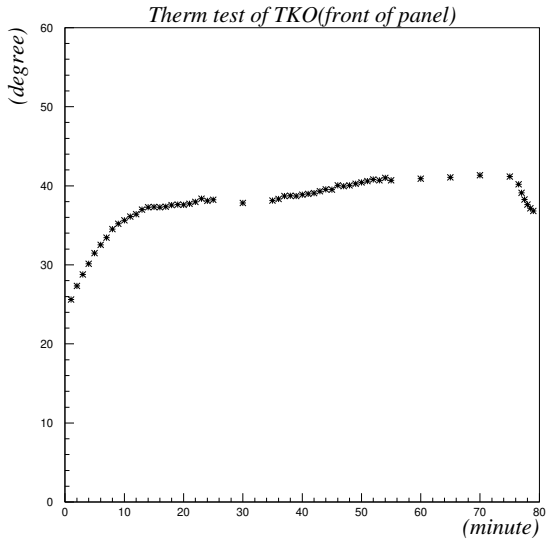


図 3.19: TKO クレート内部温度変化の様子（フロントパネル）

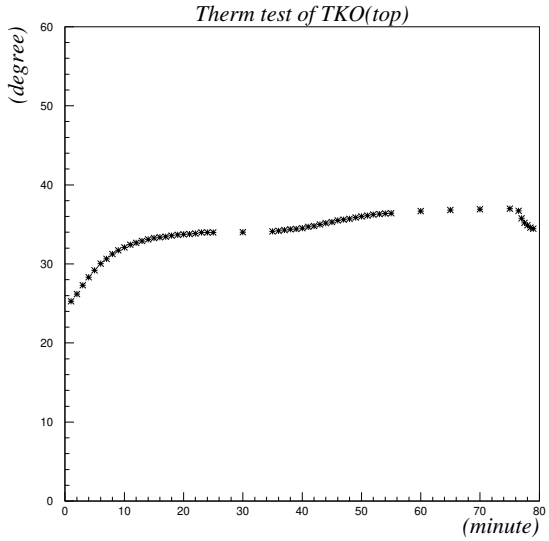


図 3.20: TKO クレート内部温度変化の様子（クレート上部）

測定点	外気温 ($°C$)	飽和後の平均温度 ($°C$)	下段クレート POWER ON 後の変化	FAN OFF 後の温度上昇	温度上昇 (FANOFF→ CRATE ON)
モジュール間	23.96	29.00	+ 2.3	14.96	16.01
フロントパネル	23.96	38.00	+2.4	-3.39	3.41
クレート上部内側	24.69	34.00	+2.8	-2.26	2.52

表 3.8: TKO クレート内部温度変化の様子

結論

ファンが正常に動作している間はフロントパネル部の温度が最も高くなっているがこれは許容範囲である。それよりもファンが作動していない場合に顕著に影響が現れるのがモジュール間であり、また温度が上昇していくにつれ最も被害を受け易いのがモジュール内部の IC 等で、これの限界温度の上限は約 50 度であるのでファンが切れた場合は大変危険である。よって、この状態をすばやく察知するためにも、温度センサーはモジュール間に設置することに決定した。また、モジュール間においても 5 番目のスロットと１２番目のスロットとにセンサーを設置して比較した結果を図 3.21に示す。これにより、より危険を察知できると思われる 5 番目のスロットに決定した。

3.3 測定器環境データの読みだし

各センサーによって感知された信号は、50m ケーブルを介してエレクトロニクスハットまで運ばた後、コネクター BOX、モニターパネルにより測定器ごと・種類（温度、湿度などの）ごとの配線並びにまとめられ、最終的にヒューレットパッカード社の HP3852A データロギングシステムにおいて数値化される。HP3852A からのデータの読み出しには、HP3852A と GPIB バスを通じて接続されている SPARC ボードを使用する。

以下のセクションでは、それぞれの装置の機能と実装について説明する。

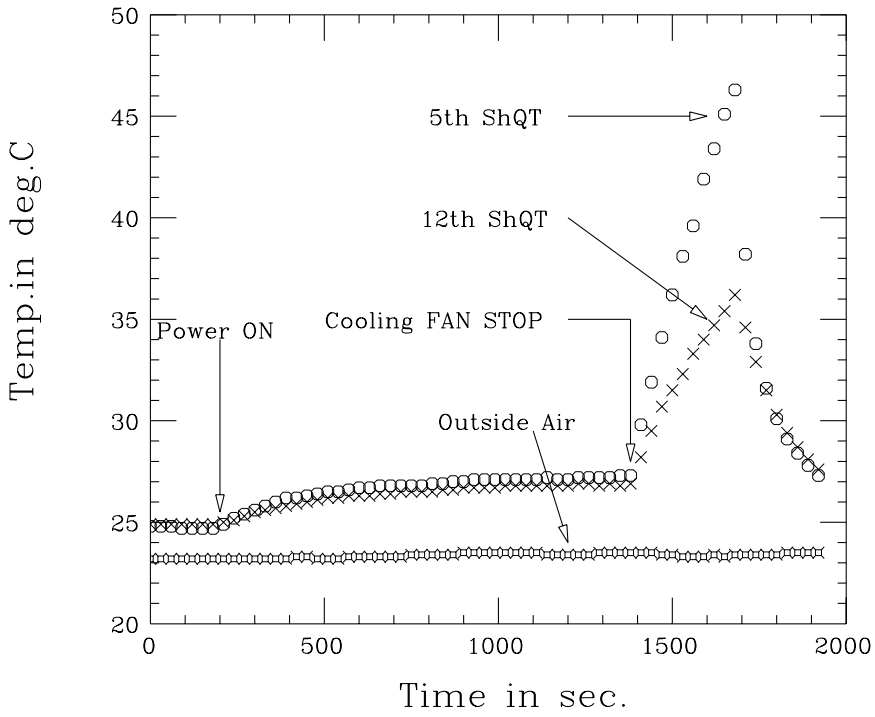


図 3.21: 5 番目のスロットと 1 2 番目のスロットとの温度比較

3.3.1 HP3852A·HP3853A データロギングシステム

HP3852A データロギングシステムはデータを集録・制御する装置である。HP3852A のみではデータ集録を行なうことは出来ず、HP3852A のリアパネル（図 3.23）[11] にプラグインアッセンブリを接続しなければならない。HP3852A は、7 つまでアッセンブリを設置するスロットを持っており、それ以上の設置には、1 台あたり 10 スロットを装備した HP3853A エクステンダを用いる。1 台の 3852A （メインフレーム）に 7 台のエクステンダを接続できる。抵抗、電圧などの測定はコマンドを入力することによって行なわれる。入力方法はフロントパネル（図 3.22）[11] のキーボードからと GPIB バスを経由してコンピュータからの入力が可能である。読み出されたデータはフロントパネルのディスプレイ上、または GPIB を通じてコンピュータ上に表示される。

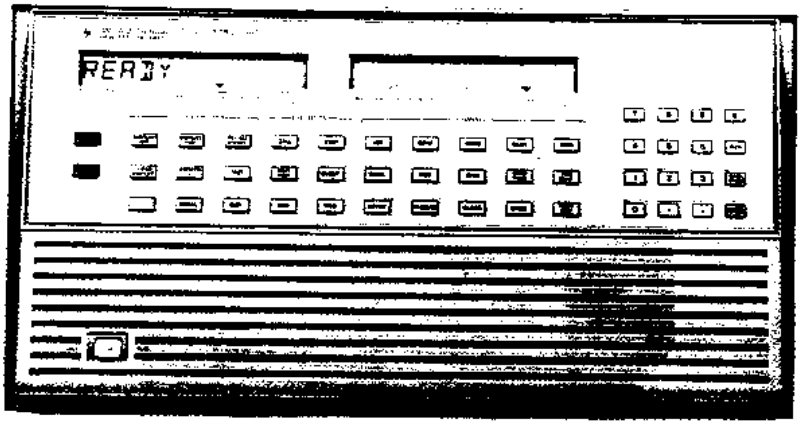


図 3.22: 3852A のフロントパネル

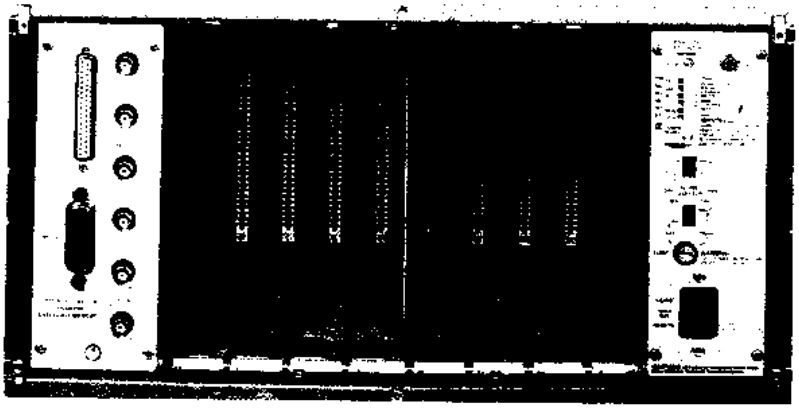


図2－3　3852Aのリア・パネル

図 3.23: 3852A のリアパネル

環境情報は、前に述べたように電圧、抵抗を測定することにより可能である。HP3852A 及び HP3853A エクステンダには 1000 点あまりの環境情報を測定するために、HP44701A インテグレイティング DVM（デジタルボルトメータ）と HP44705A リレーマルチプレクサアッセンブリを、設置する。

44701A インテグレイティング DVM 44701A はディジタル・マルチメータを内蔵したプラグイン・アッセンブリである。内部の電荷積分型 A/D コンバータにより電圧 (10nV ～ 300V)・抵抗 (10$\mu\Omega$ ～ 3MΩ)・電流のようなアナログ量を数値化し測定をおこなう。数値化のために電荷積分する際に、積分時間は電源周期 (関東地区は 50Hz のため 0.02 秒) と同じかまたはその 1/10、1/200 などに設定できる。信号がデジタル値に変換されるとそのデータはディスプレイに表示されるか GPIB を介してコンピュータ上に送られる。

44705A リレーマルチプレクサ 44705A は、44701A のようなボルトメータアッセンブリを用いて多数の測定をおこなうときに 20 チャンネルの信号を切り変えるためのものである。コモン・モードの雑音を最大限に除去するために、20 チャンネルのすべてにおいて HIGH,LOW,GUARD の切換えをおこなっている。図 3.24はカバーを外したマルチプレクサのターミナルモジュールを示し、チャンネル 0 に抵抗を接続した場合を示している [11]。20 チャンネルをこのように配線しコマンドを入力すると、デジタルボルトメータは、0 チャンネルから順に測定を開始する。

44725A 16 チャンネル GP スイッチ 44725A は、16 チャンネルのリレー C 接点を持つ汎用スイッチモジュールである。リレー C 接点とは、電源投入直後あるいはリセット時にコモン (C) とノーマルクローズ (NC) が閉じてノーマルオープン (NO) は開いている状態にあり、必要に応じてこれと逆の C と NO が閉じた状態との間を自在に遷移できるスイッチである（図 3.25参照）。これを用いて異常検知時にアラーム（警報）としてブザーを鳴らしたり、TKO 電源を非常停止したりすることが可能である。

3.3.2 GPIB バス

GPIB とはコンピュータと測定器とを接続するためのインターフェイスを作るために標準化されたバスラインで General Purpose Interfase Bus の略称である。前節で述べたように HP3852A は、フロントパネルまたは GPIB バスを通じてコンピュー

タから受けとったコマンドによって測定、データの処理をおこなう。環境モニターシステムでは、SPARC ボードと HP3852A は GPIB バスを用いてコマンドや測定値の授受をしている。GPIB には IEEE-488、IEEE-IB、GPIB、HP-IB、IFC など様々な呼び方があるが規格はほぼ同じであり、ここでは呼称を GPIB バスに統一する。

3.3.3 SPARC ボード

SPARC ボードは VME バス規格のボード上に実装された計算機であり、フォース・コンピューター社の CPU-5V 型を使用している。CPU としてサン・マイクロシステムズ社製 microSPARC-II 型マイクロプロセッサを搭載しており、同社製 SPARCstation5 と完全に互換性がある。このボードには、入出力のバスとして VME バス SCSI バス (ハードディスクなどの周辺機器接続用)、シリアルポート (VT100 端末など RS-232C 規格の機器接続用)、そして後述の SBUS などが用意されている。OS は Solaris2.5 であり、BELLE データ収集システムのサーバー計算機 (bwsdaq) に接続されたハードディスク上に存在する OS をネットワーク経由でロードして立ち上がるネットワークブートを用いている。

SBUS-GPIB

SPARC ボードから GPIB バスをコントロールするために、SBUS-GPIB インターフェースを用いる。SBUS は、サン・マイクロシステムズ社が開発した I/O 拡張バスで、SPARC ボードにもサポートされているバスである。SPARC ボードからは直接 GPIB へデータの入出力ができないため、このインターフェースを SPARC ボードの SBUS ソケットにとりつける。このインターフェースを SPARC ボードからコントロールするためには、デバイスドライバーが必要であり、インターフェイス製造元のナショナルインストゥルメント社が開発した NI488.2 ソフトウェアを用いている。

3.3.4 ネットワークシェアードメモリー (NSM)

BELLE 実験のデータ収集系は、大きく分けて、リアルタイムに動作するデータ収集サブシステムとモニターシステムのようなリアルタイム性を要しないサブシステムに分けられる。実験運転中は、全てのサブシステムはマスターコントロールによって制御されている。これらのサブシステムは TCP/IP プロトコルを基盤とした

ネットワークアプリケーションであるネットワークシェアードメモリー（NSM）を用いて、相互の情報交換・伝達をおこなう。

NSM の構造は次のようになっている [12]。

1. それぞれのホストコンピュータ（ノード）にデーモンプロセス（nsmd）として存在する。

2. ユーザーは NSM 関係のライブラリー（NSMLIB）をリンクしたプログラムを使ってノード名で示されたメモリー領域にデータの読み書きができる。

3. サブシステムの 1 対 1 のやりとりには”リクエスト”と”メッセージ”として情報を伝達し合う。（TCP）

4. データの共有（1 対多数）には”ブロードキャスト”を用いる。（ITP）

NSM によって交信している各サブシステム上では nsmd なるデーモンプロセスが常に走っている。nsmd は自らの属するサブシステムの上にシェアードメモリー（共有メモリー領域）を確保し、このシェアードメモリーにアクセスするプロセスの存在場所と中に書かれたデータの内容を他のサブシステムとの交信によって保持・更新している。nsmd は 1 サブシステムに唯一つしか存在できない。一方、client と呼ばれるユーザーのプロセスは 1 サブシステムにいくつ存在しても構わない。client は自らの存在するサブシステムの nsmd と TCP プロトコルや SIGUSR1 シグナルで対話し、必要に応じてシェアードメモリーの許可された領域のデータを読み書きする。データの書き込みや、新たな client の生成、既存 client の消滅といった事象が発生すると、その情報は各サブシステムの nsmd の間の交信によって伝達され、サブシステム毎に確保されたシェアードメモリーの内容が更新されるのである。このような通信手順を摸式的に示したのが図 3.27である。

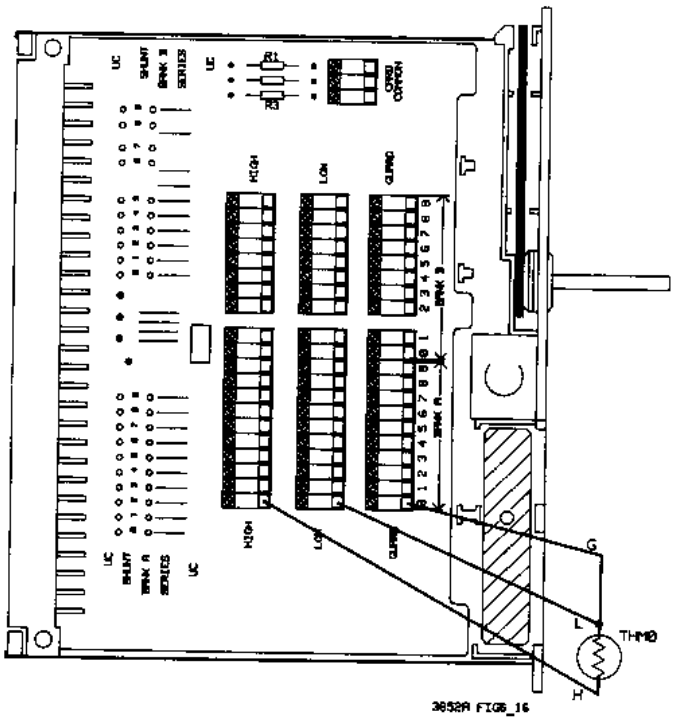
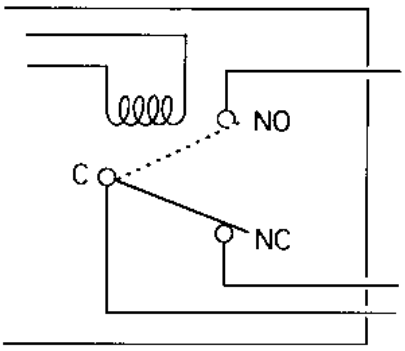


図 3.24: リレーマルチプレクサのターミナルモジュール



Normal Close = NC, Commom = C, Normal Open = NO

図 3.25: GP スイッチ切りかえの様子

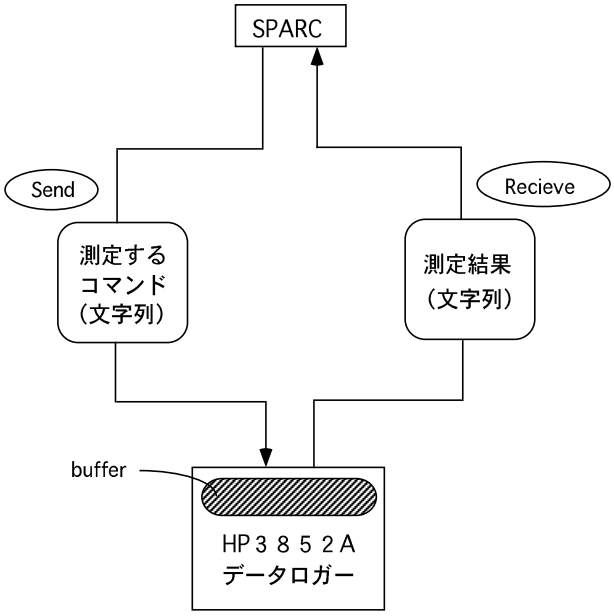


図 3.26: GPIB 構造図

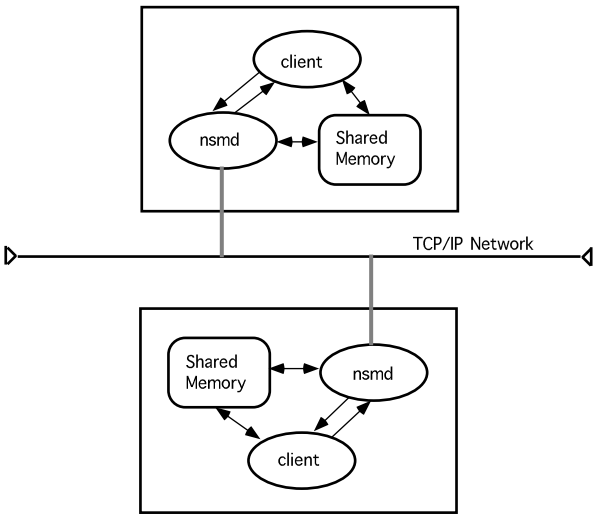


図 3.27: NSM 構造図

第 4 章

測定器モニターシステムの動作及び結果

この章では、測定器モニターシステムのソフトウェアの実装に関して、また測定結果と動作実績に関して述べる。

4.1 測定器モニターシステムのソフトウェアのアルゴリズム

前章で述べたように、測定器モニターシステムの制御計算機である SPARC ボードが GPIB バスを通じて HP3852A データロギングシステムと交信し、測定を行なうコマンドの送信と測定値の受信を行なう。また測定の結果必要ならアラームやインターロックを発する。これを実現するソフトウェアについて以下に述べる。

SBUS-GPIB インターフェイスのドライバーソフトウェア NI488.2[13] は、C 言語の関数ライブラリの形で提供されているので、測定器モニターシステムのプログラムが GPIB バスを経由して HP3852A にアクセスするには、NI488.2 の中に用意されている関数をコールしてやれば良い。GPIB バス上の通信を行なう上で欠かせない基本的命令は以下の 4 つである。

(1) インターフェイスクリア（IFC）…バスの IFC ラインを 100μs「L」レベルに保ち、バスに接続されている全てのインターフェイスをリセットする

(2) リモート・イネーブル（REN）…バスの REN ラインを「L」レベルにして、インターフェイスをリモートモードにする

(3) リード（READ）…コンピュータ自身をリスナーにし、トーカーからのデータを読みとる

(4) ライト（WRITE）…コンピュータをトーカとし、リスナーに対してデータを送信する

NI488.2M では、(1)、(3)、(4) についてはそれぞれ SendIFC、Send、Receive という関数が用意されている。(2) は DevClear という関数をコールすると、その中で実行される。

実際に稼働中の測定器モニタープログラムが、SPARC ボード上で起動された後に実行される内容と順番は以下の通りであり、その流れ図を図 4.1と図 4.2に示した。

1. SendIFC をコールし、インターフェイスクリアを行なう。

2. HP3852A の GPIB アドレスは 9 なので、行先を 9 に指定して DevClear をコールする。これで HP3852A の GPIB バッファがクリアされ、リモートイネーブルされる。

3. 以上で測定の準備が整ったので、scantko をコールし TKO クレートの温度と電源電圧の値を測定する。ctrltko にフラッグ＝ 1 を与えてコールすると、結果に異常がなければ TKO 電源の遠隔制御ラインをイネーブルし、異常がみとめられた場合（温度が $45°C$ を越えていた場合）はこれをとりやめる。

4. システムのクロックから計測開始日時を得る。それでいよいよ測定・記録のステップに入る。以下は測定器モニターのプロセスが死ぬまで、無限ループで永久に続けられる。

5. scanecl をコールし、ECL の温度・湿度を測定する。続いて wrtecl をコールし、測定結果をファイルに書き込む。前者は Send をコールして測定コマンドを HP3852A に送り、かつ Receive をコールして測定値を受けとっている。

6. scantof と wrttof をコールし、TOF の温度・湿度の測定とファイルへの書き込みを行なう。

7. scanacc と wrtacc をコールし、ACC の温度・湿度の測定とファイルへの書き込みを行なう。

8. scancdc と wrtcdc をコールし、CDC の温度・湿度の測定とファイルへの書き込みを行なう。

9. scantko と wrttko をコールし、TKO クレートの温度と電圧を測定、ファイルに書き込む。直後に ctrltko にフラッグ＝ 0 を与えてコールしており、温度が $45°C$ 以上の TKO クレートがあれば該当する遠隔制御ラインをディスエーブルし、TKO 電源の非常停止をする。

10. 次回の測定・記録のステップを始める時刻になるまで、スリープしつつ2秒に一回システムのクロックをチェックしつつ待つ。

11. 新たな測定・記録ステップを始める際に、前回の測定と日付が変わっていたら、測定結果を記録するファイル（ログファイル）を更新する。

プログラムの開発を通して注意したことを以下に列挙する。

(1) 測定器サブシステム毎に、測定の実行、結果のファイルへの書き込み、インターロック等の制御、と機能別に関数を用意して関数名にも規則性を持たせ、プログラムの保守・拡張を容易にした。

(2) (1) で述べた各々の関数の間のデータの受け渡しは、測定器サブシステム毎に、測定データを格納するのに適した型の構造体を定義し、これら構造体のアドレスによるデータの受け渡し（構造体ポインタの参照）を用いた。この構造体群はenvmon.hなるヘッダーファイルにまとめられており、これもプログラムの保守性と拡張性に貢献している。

(3) 現段階では、あえて1163点におよぶ全チャンネルの測定を一巡するまでの速度を求めず、安定性を重視してHP44705Aリレーマルチプレクサ1台ごと（20チャンネル）に測定コマンドの発行と測定値の受信を実行している。

次節で測定結果と動作実績について述べる。

4.2 測定結果と動作実績

BELLE測定器システム全体で1163ヶ所におよぶ測定点が存在するが、その全ての測定記録を示すのは不可能であるので、バレルカロリメーター内の温度・湿度の測定記録を図4.3に示す。これは、72ヶあるバレルカロリメーターのセグメントのうち、一つを選んで1999年2月7日午前0時から2月9日午前0字までの48時間の温度・湿度の履歴を表示したものである。$20°C$の水を循環させることによる冷却と乾燥空気の導入を開始してから既に一カ月以上が経過しており、かつ読みだしエレクトロニクスの電源も投入した状態を一週間以上継続しているので、温度・湿度とも平衡に達して安定している様子が分かる。また、カロリメーター内の場所による温度の違いも明瞭に測定されている。

現状では1163点の測定に70秒程度を要し、2分に一回の測定・記録を行なっている。1998年10月下旬から測定器の温度・湿度のモニターを始め、ハードウェア

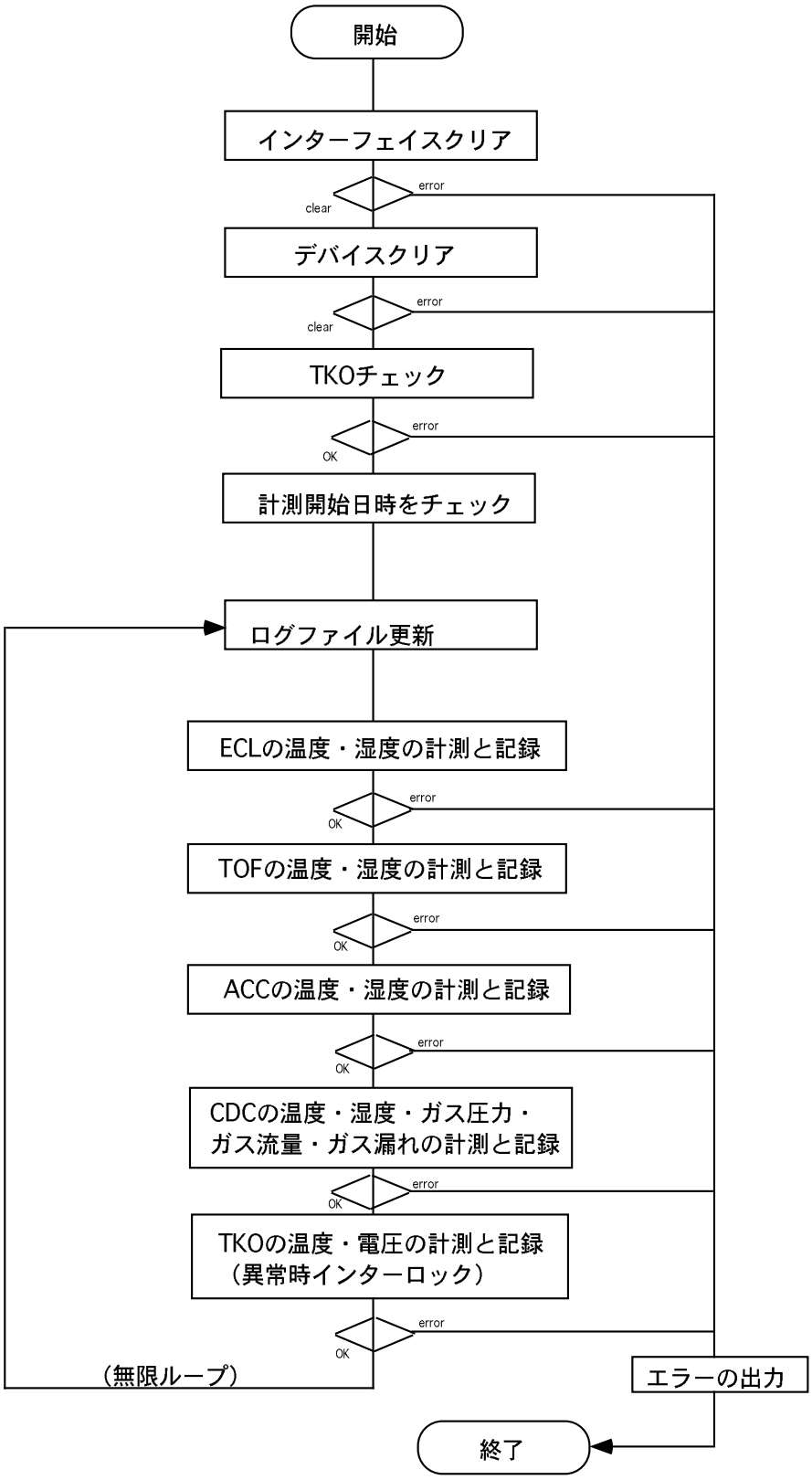


図 4.1: データ収集プログラムの流れ（メインプログラム）

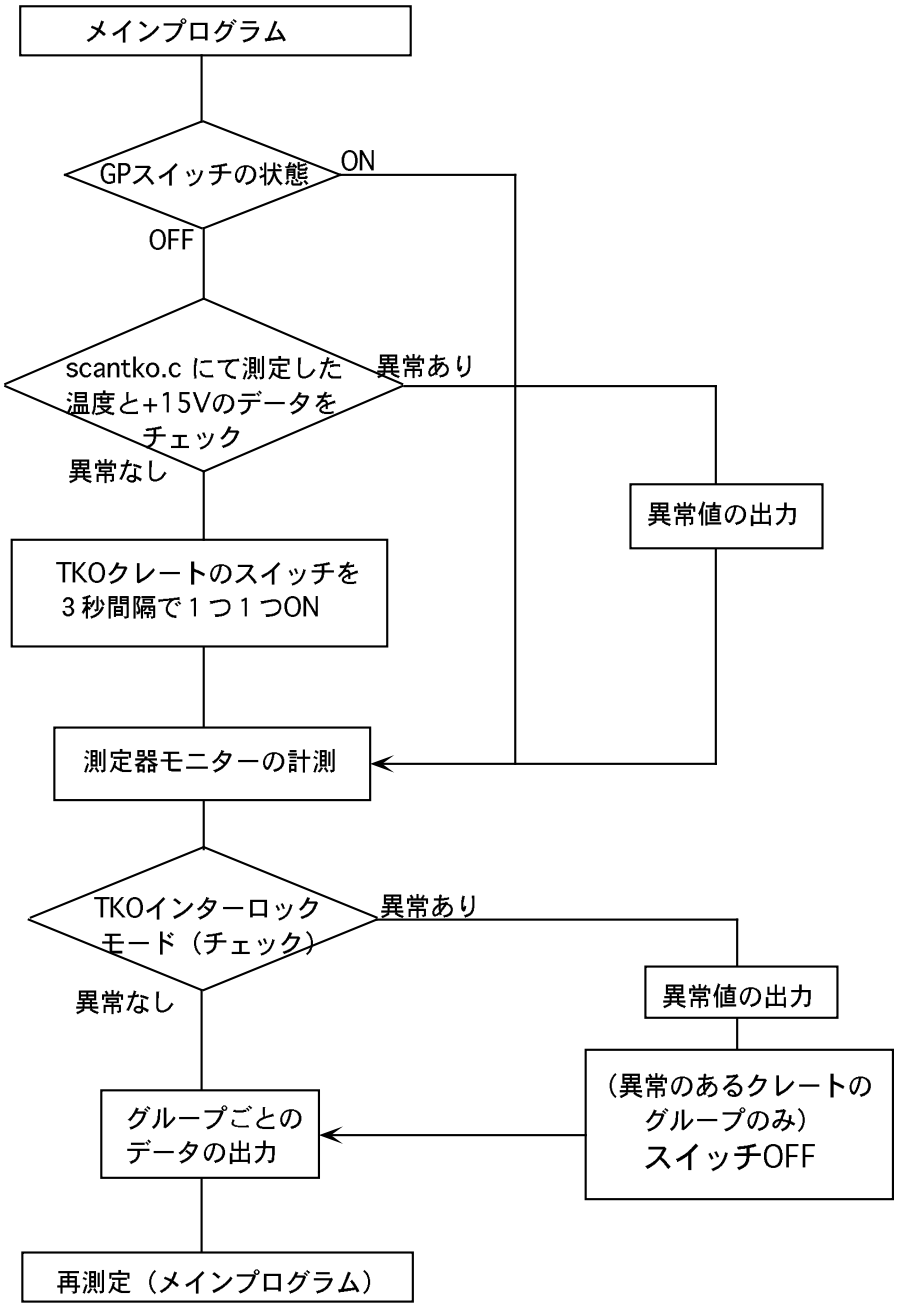


図 4.2: プログラムの流れ（インターロック）

の配線とソフトウェアのバージョンアップを繰り返してきたが、三カ月以上に渡って予期せざるトラブルによる停止は一度も発生していない。

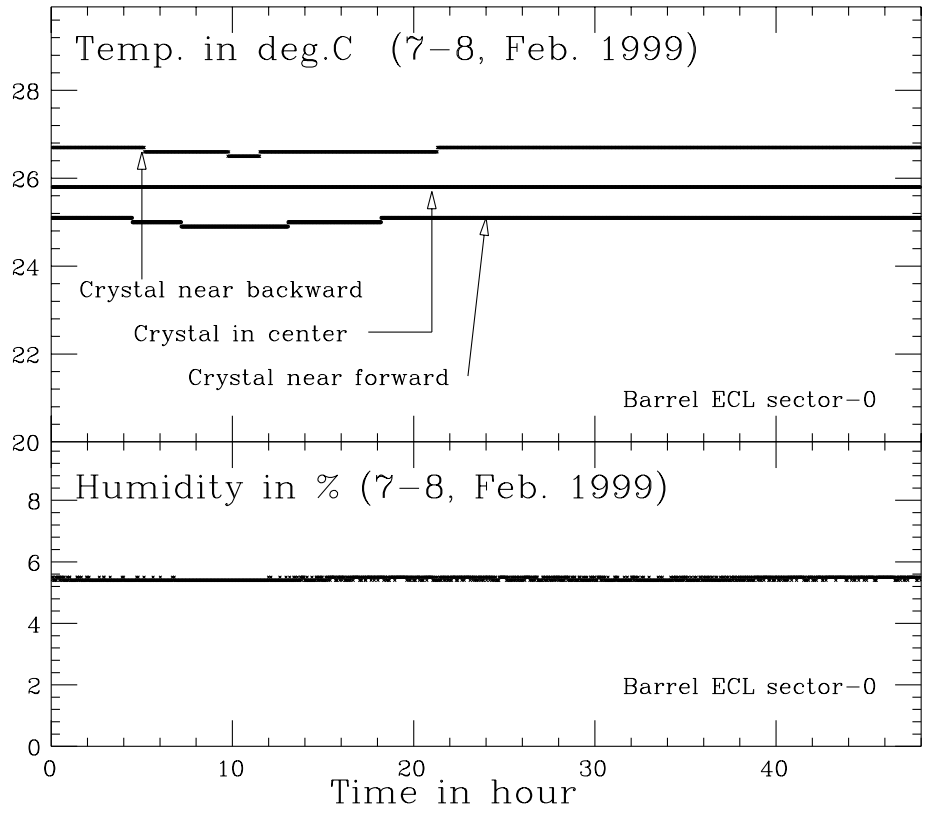

Temp. in deg.C (7−8, Feb. 1999)
28
26
24
22
20
Crystal near backward
Crystal in center
Crystal near forward
Barrel ECL sector−0
Humidity in % (7−8, Feb. 1999)
8
6
4
2
0
Barrel ECL sector−0
0
10
20
30
40
Time in hour

図 4.3: 測定結果

第 5 章

まとめ

BELLE 測定器モニターシステムを製作をし、これを稼働させた。測定器モニターシステム実機の本格的な構築に先立ち、以下のテストを行なった。

1. 湿度センサーの信号読みだしに対する 50m ケーブルの抵抗による影響の測定
2. エレキハット内各種機器接続後の温度・湿度センサー総合動作テスト
3. TKO クレート内部温度測定テスト

1 の結果、ケーブルを介した場合と介さなかった場合での湿度の測定値の差は 0.7 % 以下であり、50m ケーブル抵抗による湿度測定値のずれは実用上問題ないということが確認された。

2 の結果では、センサーの外部容器（シールド）に人や物が接触することでグランドループが生じ測定値が乱れるという問題が起こったが、シールド線を切断することで解決された。またエレキハット内での最終的なセットアップの形で、測定値のふらつきが 0.1 %未満という充分な精度を持って測定できることが分かった。

3 の結果、同じクレート内でも場所によって大きく温度の振る舞いが異なることが発見され、冷却用ファンが切れた時などの非常事態を検知できる場所として、熱源であるモジュールの間に温度センサーを設置することを決定した。

以上のテストの結果を踏まえて、各測定点へのセンサーの設置、各中継機器の設計製作と設置、接続を行なって、ハードウェア面の実装を完了した。

また測定器モニターシステムによるデータ収集と処理にあたり、以下の機能を果たすソフトウェアを作成した。

1. 一定の時間間隔でデータロガーから測定データを収集し、ファイルに保存すると共に、異常を検知した場合はアラームとインターロックを発する。

2. ファイルに保存されたデータを読みだし、測定値の履歴を表示する

これにより、BELLE コラボレーションに属する誰からでも測定器環境の過去・現在のデータを見ることが出来る。また、測定器環境の異常をリアルタイムで察知し、測定器自体や周辺機器の損傷を最小限にくいとめることが出来る。そしてこれらの情報はイベントデータ解析時に、実験データの較正に参照される。

現在、この BELLE 測定器モニターシステムは連続して稼働中であり、安定に動作している。

1999 年春から始動する B ファクトリー実験において当システムが果たす役割は重要であり、実験の成功へと貢献することを確信している。

付録 A

データ収集プログラム

SPARC 上で起動される測定器モニターのプログラムをここでは ECL に関するソースコードのみを以下に記載する。

1. メイン関数プログラム <env04r2.c>

```
/*****************************************************************
 * env04r2.c                    created 1999/02/04  Kenkichi Miya.
 * Environment monitor Version0.4 Release2 main function.
 * The implemented features are;
 * 1) Monitoring temperature and humidity in CDC, ACC, TOF, ECL.
 *    Original version of each function was written by S.Yokoi.
 * 2) Monitoring temperature and voltages of ECL's TKO crates.
 *    Original version of this function was written by S.Yokoi.
 * 3) Controling ECL's TKO.  Power ON at the begining, automatic
 *    shutdown in emergency case.
 * 4) Temerature alarm of ACC and TKO were implemented.
 *****************************************************************/
#include"ugpib.h"   /* GPIB instruction header */
#include<stdio.h>   /* standard input/output library header */
#include<stdlib.h>
#include<string.h>
#include"envmon.h"
#include <time.h>    /* to access system clock. */
#include <sys/time.h>
```

```
#include <sys/types.h>

int scanecl( struct envecl *ptrecl ); /* Prototype of func. to scan
                                         temp.& humid. of ECL      */
int scantof( struct envtof *ptrtof ); /* Prototype of func. to scan
                                         temp.& humid. of TOF      */
int scanacc( struct envacc *ptracc ); /* Prototype of func. to scan
                                         temp.& humid. of ACC      */
int scancdc( struct envcdc *ptrcdc ); /* Prototype of func. to scan
                                         temp.& humid. of CDC      */
int scantko( struct envtko *ptrtko ); /* Prototype of func. to scan
                                         temp.& volt. of TKO elec. */
int ctrltko( struct envtko *ptrtko, int mode ); /* Prototype of func.
       to contrl TKO elec. */
int almacc( struct envacc *ptracc ); /* Prototype of func. of ACC alarm */
int almtko( struct envtko *ptrtko ); /* Prototype of func. of TKO alarm */

time_t tloc;
struct tm *lt;
int idate, month, mday, hour, min, sec;
int isecstart, isec, isecnext, filedate;

/*=======
 * Main routine.
 *=======*/
int main()
{
  int icond, mode;
  int iline, ieven, iodd, itemp, iseq;
  struct envecl ptrecl;
  struct envtof ptrtof;
  struct envacc ptracc;
  struct envcdc ptrcdc;
```

```
struct envtko ptrtko;

/*================
 * Interface clear
 *================*/
SendIFC(0);  /* The installed board is 0th one. */
if( ibsta & ERR ){gpiberr("SendIFC error"); exit(1);}

/*==============================
 * Device Clear (Single device)
 *==============================*/
DevClear( 0, 9);  /* HP3852A's GPIB address is 9. */
if( ibsta & ERR ){gpiberr("DevClear error"); exit(1);}

/*============================
 * Scan TKO electronics, then pass the result to enable if it's OK.
 *============================*/
icond = scantko( &(ptrtko) );
icond = ctrltko( &(ptrtko), 1 );
if( icond!=0 )
{
  printf("Temp. or Volt. problem in TKO. Not all ones are enabled.\n");
}

/*-----------------
 * Check start date & time.
 *-----------------*/
tloc = time( NULL ); lt = localtime( &tloc );
mday = lt->tm_mday; sec = lt->tm_sec;
filedate = mday; isecstart = sec; isecnext = sec;
if( isecnext < 2 ){ isecnext = isecnext + 60; }

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

```
 * Periodic measurement by sleep call.
 *==================================*/
while(1)
{
  /* Refresh log file */
  if( filedate != mday )
  {
    system("mv today_ecl.dat yesterday_ecl.dat");
    system("mv today_fbecl.dat yesterday_fbecl.dat");
    system("mv today_tof.dat yesterday_tof.dat");
    system("mv today_acc.dat yesterday_acc.dat");
    system("mv today_cdc.dat yesterday_cdc.dat");
    system("mv today_tko.dat yesterday_tko.dat");
    filedate = mday;
  }
  /* Periodic measuremsnt in every min. */
  if( sec == isecstart )
  {
    /*----------------------------------
     * ECL measure and write temp.& humid.
     *----------------------------------*/
    icond = scanecl( &(ptrecl) );
    icond = wrtecl( &(ptrecl) );
    /*----------------------------------
     * TOF.
     *----------------------------------*/
    icond = scantof( &(ptrtof) );
    icond = wrttof( &(ptrtof) );
    /*----------------------------------
     * ACC.
     *----------------------------------*/
    icond = scanacc( &(ptracc) );
    icond = wrtacc( &(ptracc) );
```

```
        icond = almacc( &(ptracc) );
        /*----------------------------------
         * CDC.
         *----------------------------------*/
        icond = scancdc( &(ptrcdc) );
        icond = wrtcdc( &(ptrcdc) );
        /*---------------------------------
         * TKO electronics for ECL.
         *---------------------------------*/
        icond = scantko( &(ptrtko) );
        icond = wrttko( &(ptrtko) );
        icond = ctrltko( &(ptrtko), 0 );
        icond = almtko( &(ptrtko) );
      }
      /*-------------------------------------------------------------
       * Check clock with 2sec. interval to resume next measurement.
       -------------------------------------------------------------*/
      tloc = time( NULL ); lt = localtime( &tloc );
      mday = lt->tm_mday; sec = lt->tm_sec;
      if( (isecnext-sec)>2 )
      {
        sleep(2);
      }
    }
  }
```

2. <scanecl.c> (muxecl をコール、ECL の温度・湿度の測定)

```
/*****************************************************************
 * scanecl.c created                    1998/10/26 Kenkichi Miya.
 * Scan all temp.& humid. sensors installed in Barrel CsI.
 * IEEE488 interface clear and HP3852A device clear should
 * be done before calling this function in main function.
```

```
 * Modify in 1998/11/03 by Kenkichi Miya.;
 *        #include"envmon.h" instead of each struct decreration.
 * Modify in 1998/12/17 by Kenkichi Miya.;
 *        Both endcaps are also scanned.
 ************************************************************/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include "envmon.h" /* Envmon. header */

float regist[20], volt[20];

int muxecl( int id_mux, int *nomeas, struct multiplexer *result  );
                                    /* func. to scan one multiplexer. */

/*=====
 * entry point
 ******/
int scanecl( struct envecl *ptrecl )
{
  int ireturn;
  int icond, id_mux, number_of_meas;
  int iseq, ich;
  struct multiplexer result;

  ireturn = 0; /* normal end.*/
  /*---
    Temperature measurement
    ---*/
  iseq = 0; /* reset the counter. */
  /* for( id_mux=0; id_mux<11; id_mux++ )*/ /* scan only barrel.(981027)*/
  for( id_mux=0; id_mux<16; id_mux++ ) /* scan whole ECL.(981217)*/
  {
```

```
    /* Scanning one HP4705A multiplexer */
    icond = muxecl( id_mux, &number_of_meas, &(result) );
    if( icond!=0 ){printf("error\n");}

    for( ich=0; ich<number_of_meas; ich++)
    {
      regist[ich] = result.value[ich];
      ptrecl->tempecl[iseq]=therm( regist[ich] );
      iseq++;
    }
  }
  /*  printf("after temp. iseq=%d\n", iseq ); */ /* for debug. */

  /*---
    Humidity measurement
    ---*/
  iseq = 0; /* reset the counter */
  /* for( id_mux=16; id_mux<20; id_mux++)*/ /* scan only barrel.(981027)*/
  for( id_mux=16; id_mux<22; id_mux++)  /* scan whole ECL.(981217)*/
  {
    icond = muxecl( id_mux, &number_of_meas, &(result) );
    if( icond!=0 ){printf("error\n");}

    for( ich=0; ich<number_of_meas; ich++)
    {
      volt[ich] = result.value[ich];
      ptrecl->humidecl[iseq]= volt[ich]*100.0;
      iseq++;
    }
  }
  /*  printf("after humid. iseq=%d\n", iseq ); */ /* for debug. */
  return ireturn=icond;
}
```

3. <muxecl.c> (HP3852A へコマンドを送り、データを受けとる)

```
/*****************************************************************
 * muxecl.c created                      1998/10/26 Kenkichi Miya.
 * Scan one HP44705A multiplexer in the proper slot specified by
 * the integer number(id_mux) given as the argument.
 * IEEE488 interface clear and HP3852A device clear should
 * be done before calling this function in somewhere.
 * nomeas is int pointer to return the number of measurement in
 * the multiplexer.
 * The measured values are returned by float pointer *result[20].
 * Return value ... 0=normal, -1=send error, -2=receive error.
 * Bug fix ... number of Receive exec. was corrected. 1998/10/31.
 *****************************************************************/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include "ugpib.h"   /* GPIB instruction header */
#include "envmon.h"  /* Envmon. header          */
#define MAVbit 0x45  /* Message AVailable bit */
char buffer[24];      /* buffer to receive measurement */

/*===
 * entry point
 *===*/
int muxecl( int id_mux, int *nomeas, struct multiplexer *result  )
     /* id_mux=seq.# of multiplexer, nomeas=# of measurements,
measurements are returned by result[20]. */
{
  int ireturn; /* 0=normal, -1=send error, -2=receive error. */
  int ich;
  ireturn = 0;
```

```
switch( id_mux )
{
/*---
 * Temperature measurement is done by OHM mode.
 *---*/
case 0:
  Send( 0, 9, "CONF MEAS OHM, 400-419, USE 0", 29L, DABend);
  *nomeas = 20; break;
case 1:
  Send( 0, 9, "CONF MEAS OHM, 500-519, USE 0", 29L, DABend);
  *nomeas = 20; break;
case 2:
  Send( 0, 9, "CONF MEAS OHM, 600-619, USE 0", 29L, DABend);
  *nomeas = 20; break;
case 3:
  Send( 0, 9, "CONF MEAS OHM, 700-719, USE 0", 29L, DABend);
  *nomeas = 20; break;
case 4:
  Send( 0, 9, "CONF MEAS OHM, 1000-1019, USE 0", 31L, DABend);
  *nomeas = 20; break;
case 5:
  Send( 0, 9, "CONF MEAS OHM, 1100-1119, USE 0", 31L, DABend);
  *nomeas = 20; break;
case 6:
  Send( 0, 9, "CONF MEAS OHM, 1200-1219, USE 0", 31L, DABend);
  *nomeas = 20; break;
case 7:
  Send( 0, 9, "CONF MEAS OHM, 1300-1319, USE 0", 31L, DABend);
  *nomeas = 20; break;
case 8:
  Send( 0, 9, "CONF MEAS OHM, 1400-1419, USE 0", 31L, DABend);
  *nomeas = 20; break;
case 9:
```

```
  Send( 0, 9, "CONF MEAS OHM, 1500-1519, USE 0", 31L, DABend);
  *nomeas = 20; break;
case 10:
  Send( 0, 9, "CONF MEAS OHM, 1600-1619, USE 0", 31L, DABend);
  *nomeas = 20; break;
case 11:
  Send( 0, 9, "CONF MEAS OHM, 1700-1719, USE 0", 31L, DABend);
  *nomeas = 20; break;
case 12:
  Send( 0, 9, "CONF MEAS OHM, 1800-1819, USE 0", 31L, DABend);
  *nomeas = 20; break;
case 13:
  Send( 0, 9, "CONF MEAS OHM, 1900-1919, USE 0", 31L, DABend);
  *nomeas = 20; break;
case 14:
  Send( 0, 9, "CONF MEAS OHM, 2000-2019, USE 0", 31L, DABend);
  *nomeas = 20; break;
case 15:
  Send( 0, 9, "CONF MEAS OHM, 2100-2111, USE 0", 31L, DABend);
  *nomeas = 12; break;
/*---
 * Humidity measurement is done by DCV mode.
 *---*/
case 16:
  Send( 0, 9, "CONF MEAS DCV, 2200-2219, USE 0", 31L, DABend);
  *nomeas = 20; break;
case 17:
  Send( 0, 9, "CONF MEAS DCV, 2300-2319, USE 0", 31L, DABend);
  *nomeas = 20; break;
case 18:
  Send( 0, 9, "CONF MEAS DCV, 2400-2419, USE 0", 31L, DABend);
  *nomeas = 20; break;
case 19:
```

```
    Send( 0, 9, "CONF MEAS DCV, 2500-2519, USE 0", 31L, DABend);
    *nomeas = 20; break;
  case 20:
    Send( 0, 9, "CONF MEAS DCV, 2600-2619, USE 0", 31L, DABend);
    *nomeas = 20; break;
  case 21:
    Send( 0, 9, "CONF MEAS DCV, 2700-2703, USE 0", 31L, DABend);
    *nomeas = 4; break;
  }
  if( ibsta & ERR ){gpiberr("Send error"); ireturn = -1;}

  /*----------------
   * Receive message
   *----------------*/
  for( ich=0; ich<*nomeas; ich++ )
  {
    Receive( 0, 9, buffer, 15L, STOPend);
    if( ibsta & ERR ){gpiberr("Receive error"); ireturn = -2;}
    result->value[ich] = atof( buffer );
    /*    printf("%d %f\n", ich, result->value[ich]); */ /* for debug. */
  }
  return ireturn;
}
```

4. <wrtecl.c> (測定データをファイルに書き込む)

```
/*****************************************************************
 * wrtecl.c                    created 1998/11/04  Kenkichi Miya.
 * Get the monitored temp.& humid. in BELLE ECL via struct
 * envecl and write the contents into file(recent_env.dat).
 * Modify : Both endcaps are written into other files.
 *                                    1998/12/17  Kenkichi Miya.
 *****************************************************************/
```

```
#include<stdio.h>   /* standard input/output library header */
#include<stdlib.h>
#include<string.h>
#include"envmon.h"
#include<time.h>
#include<sys/time.h>
#include<sys/types.h>

time_t tloc;
struct tm *lt;
int year, month, date, hour, min, sec;
FILE *outfile;
FILE *data_today;
FILE *outf_endcap;
FILE *data_endcap;

/*=======
 * Entry point
 *=======*/
int wrtecl( struct envecl *ptrecl)
{
  int icond, iline, ieven, iodd, itemp, iseq, iheven, ihodd;
  /*----------------
   * What time is it?
   *----------------*/
  tloc = time( NULL ); lt = localtime( &tloc );
  year=lt->tm_year; month=lt->tm_mon+1; date=lt->tm_mday;
  hour=lt->tm_hour; min=lt->tm_min; sec=lt->tm_sec;

  /*---------------------------
   * print out Barrel ECL measurement.
   *---------------------------*/
  outfile = fopen("recent_ecl.dat", "w");
```

```
data_today = fopen("today_ecl.dat", "a");

fprintf( outfile, "%2d %2d %2d %2d %2d %2d\n",
 year, month, date, hour, min, sec);
fprintf( data_today,
 "%2d %2d %2d %2d %2d %2d\n", year, month, date, hour, min, sec);

for( iline=0; iline<36; iline++ )
{
  ieven = 2*iline; iodd = 2*iline+1;
  itemp = 6*iline;
  /* Recent file */
  fprintf( outfile,
   "%2d %5.1f %5.1f %5.1f %5.1f   %2d %5.1f %5.1f %5.1f %5.1f\n",
   ieven, ptrecl->tempecl[itemp],ptrecl->tempecl[itemp+1],
   ptrecl->tempecl[itemp+2],ptrecl->humidecl[ieven],
   iodd, ptrecl->tempecl[itemp+3],ptrecl->tempecl[itemp+4],
   ptrecl->tempecl[itemp+5],ptrecl->humidecl[iodd]);
  /* Log today's record. */
  fprintf( data_today,
   "%5.1f %5.1f %5.1f %5.1f   %5.1f %5.1f %5.1f %5.1f\n",
   ptrecl->tempecl[itemp],ptrecl->tempecl[itemp+1],
   ptrecl->tempecl[itemp+2],ptrecl->humidecl[ieven],
   ptrecl->tempecl[itemp+3],ptrecl->tempecl[itemp+4],
   ptrecl->tempecl[itemp+5],ptrecl->humidecl[iodd]);
}
fclose(outfile);
fclose(data_today);

/*---------------------------
 * print out Endcap ECL measurement.
 *---------------------------*/
outf_endcap = fopen("recent_fbecl.dat", "w");
```

```
data_endcap = fopen("today_fbecl.dat", "a");

fprintf( outf_endcap, "%2d %2d %2d %2d %2d %2d\n",
 year, month, date, hour, min, sec);
fprintf( data_endcap,
 "%2d %2d %2d %2d %2d %2d\n", year, month, date, hour, min, sec);

for( iline=0; iline<16; iline++ )
{
  ieven = 2*(iline%8); iodd = 2*(iline%8)+1;
  iheven = 2*iline + 72; ihodd = iheven + 1;
  itemp = 6*iline + 216;
  /* Recent file */
  fprintf( outf_endcap,
   "%2d %5.1f %5.1f %5.1f %5.1f   %2d %5.1f %5.1f %5.1f %5.1f\n",
   ieven, ptrecl->tempecl[itemp],ptrecl->tempecl[itemp+1],
   ptrecl->tempecl[itemp+2],ptrecl->humidecl[iheven],
   iodd, ptrecl->tempecl[itemp+3],ptrecl->tempecl[itemp+4],
   ptrecl->tempecl[itemp+5],ptrecl->humidecl[ihodd]);
  /* Log today's record. */
  fprintf( data_endcap,
   "%5.1f %5.1f %5.1f %5.1f   %5.1f %5.1f %5.1f %5.1f\n",
   ptrecl->tempecl[itemp],ptrecl->tempecl[itemp+1],
   ptrecl->tempecl[itemp+2],ptrecl->humidecl[iheven],
   ptrecl->tempecl[itemp+3],ptrecl->tempecl[itemp+4],
   ptrecl->tempecl[itemp+5],ptrecl->humidecl[ihodd]);
}
fclose(outf_endcap);
fclose(data_endcap);

/** Scan temp. and beep if greater ten 40 deg. **/
for( iseq=0; iseq<312; iseq++ )
{
```

```
    if( ptrecl->tempecl[iseq]>40.0 )
    {

      printf("\007");    /* beep. */
      if( iseq<216 )
      {
itemp = iseq/3;
printf("High temp. %5.1f deg. at Barrel sector %d.\n",
     ptrecl->tempecl[iseq], itemp);
      }
      else if( iseq<264 )
      {
itemp = (iseq-216)/3;
printf("High temp. %5.1f deg. at FwdCsI sector %d.\n",
     ptrecl->tempecl[iseq], itemp);
      }
      else
      {
itemp = (iseq-264)/3;
printf("High temp. %5.1f deg. at BwdCsI sector %d.\n",
     ptrecl->tempecl[iseq], itemp);
      }
    }
  }
}
```

5. <ctrltko.c> (scantko をコールした後、制御を行なう)

```
/*********************************************************************
 * ctrltko.c                        created 1998/12/19 Kenkichi Miya.
 * function to enable/disable/interlock TKO Power Supply.
 * The temprature and voltages of TKO crate have to be given by
 * the pointer of struct envtko.
 * The input argument(int mode) is the mode selector;
```

```
 *   1: enable all TKO crates except for problematic ones,
 *   0: nominal interlock.
 *  -1: disable all TKO crates.
 * The return value is set to be;
 *   0: Normal return, -1: GPIB send error, -2: GPIB receive error,
 *   100+(ch.# of HP44725 GP switch) in emergency stop case.
 ****************************************************************/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include "ugpib.h"   /* GPIB instruction header */
#include "envmon.h" /* Envmon. header */
#define MAVbit 0x45  /* Message AVailable bit */
char buffer[24];      /* buffer to receive measurement */

/*====
 * entry point
 *=====*/
int ctrltko( struct envtko *ptrtko, int mode )
{
  int ialready, iftemp, ifp15v, ich, icrate;
  const int ctrlines[52] = { 0, 0, 1, 1, 1, 1, 1, 2, 2, 2, 2,
3, 3, 3, 3, 3, 4, 4, 4, 4,
5, 5, 5, 5, 5, 6, 6, 6, 6,
7, 7, 7, 7, 7, 0, 0,
8, 8, 8, 8, 9, 9, 9, 9,
10,10,10,10,11,11,11,11 };
  int idisable;
  int status[12]; /* Status of each channel. 0=off, 1=on. */
  int ireturn;    /* return value */
  ireturn = 0;    /* reset return value */

  switch( mode )
```

```
  {
  /*-----------------------
   * CASE 1: Enable TKO electronics
   *-----------------------*/
  case 1:
    /* Check whether TKO are already enabled. */
    Send( 0, 9, "CLOSE? 6000-6011", 16L, DABend);
    if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}

    ialready = 1;
    for( ich=0; ich<12; ich++ )
    {
      Receive( 0, 9, buffer, 8L, STOPend);
      if( ibsta & ERR ){gpiberr("Receive error"); ireturn=-2;}
      status[ich] = atoi( buffer );
      if( status[ich] == 0 ) /* Disabled TKO exists */
      {
ialready = 0;
      }
    }
    /* Just for debug. */
    printf("TKO status before enable= %d %d %d %d %d %d %d %d %d %d %d %d\n",
   status[0], status[1], status[2], status[3], status[4],
   status[5], status[6], status[7], status[8], status[9],
   status[10], status[11]);


    /* Check temperatures and +15V voltage. */
    iftemp = 0; ifp15v = 0;
    for( icrate=0; icrate<52; icrate++ )
    {
      if( ptrtko->temptko[icrate]>45.0 )   /* High temp. > 45deg. */
      /* if( ptrtko->temptko[icrate]>38.0 ) /* Test */
```

```
      {
iftemp = 1;
printf(
"High temp. %4.1f in TKO %d (0-35:barrel 36-43:Fwd 44-51:Bwd)\n",
       ptrtko->temptko[icrate], icrate );
ireturn = 100 + icrate; /* Set return value here. */
      }
      if( ptrtko->p15v[icrate]>16.0 ) /* Abnormal volt. in +15V */
      {
ifp15v = 1;
printf(
"Abnormal volt. %4.1f in TKO %d (0-35:barrel 36-43:Fwd 44-51:Bwd)\n",
       ptrtko->p15v[icrate], icrate );
ireturn = 100 + icrate; /* Set return value here. */
      }
    }

    /* If everything is OK, */
    if( iftemp==0 && ifp15v==0 )
    {
      /* if not yet enabled, enable TKO station one by one
       with 3 sec interval */
      if( ialready==0 )
      {
Send( 0, 9, "CLOSE 6000", 10L, DABend);
if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
else{printf("Barrel TKO station 0 enabled.\n"); sleep(3);}

Send( 0, 9, "CLOSE 6001", 10L, DABend);
if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
else{printf("Barrel TKO station 1 enabled.\n"); sleep(3);}

Send( 0, 9, "CLOSE 6002", 10L, DABend);
```

```
if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
else{printf("Barrel TKO station 2 enabled.\n"); sleep(3);}

Send( 0, 9, "CLOSE 6003", 10L, DABend);
if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
else{printf("Barrel TKO station 3 enabled.\n"); sleep(3);}

Send( 0, 9, "CLOSE 6004", 10L, DABend);
if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
else{printf("Barrel TKO station 4 enabled.\n"); sleep(3);}

Send( 0, 9, "CLOSE 6005", 10L, DABend);
if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
else{printf("Barrel TKO station 5 enabled.\n"); sleep(3);}

Send( 0, 9, "CLOSE 6006", 10L, DABend);
if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
else{printf("Barrel TKO station 6 enabled.\n"); sleep(3);}

Send( 0, 9, "CLOSE 6007", 10L, DABend);
if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
else{printf("Barrel TKO station 7 enabled.\n"); sleep(3);}

Send( 0, 9, "CLOSE 6008", 10L, DABend);
if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
else{printf("Fwd CsI TKO(FE0-FE3) enabled.\n"); sleep(3);}

Send( 0, 9, "CLOSE 6009", 10L, DABend);
if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
else{printf("Fwd CsI TKO(FE4-FE7) enabled.\n"); sleep(3);}

Send( 0, 9, "CLOSE 6010", 10L, DABend);
if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
```

```
else{printf("Bwd CsI TKO(BE0-BE3) enabled.\n"); sleep(3);}

Send( 0, 9, "CLOSE 6011", 10L, DABend);
if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
else{printf("Bwd CsI TKO(BE4-BE7) enabled.\n");}
      }
      /* if already enabled, keep as it is. */
    }
    /* Display status before leaving. */
    Send( 0, 9, "CLOSE? 6000-6011", 16L, DABend);
    if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
    for( ich=0; ich<12; ich++ )
    {
      Receive( 0, 9, buffer, 8L, STOPend);
      if( ibsta & ERR ){gpiberr("Receive error"); ireturn=-2;}
      status[ich] = atoi( buffer );
    }
    printf("TKO status after enable= %d %d %d %d %d %d %d %d %d %d %d %d\n",
   status[0], status[1], status[2], status[3], status[4],
   status[5], status[6], status[7], status[8], status[9],
   status[10], status[11]);

    break;

  /*-------------------------
   * CASE 0: Nominal interlock mode.
   *-------------------------*/
  case 0:
    /* Check temperatures and +15V voltage.                        *
     * If there is abnormal temperature or voltage, disable the    *
     * relavant TKO station.                                       *
     * The problematic TKO crate is returned by func value.       */
    for( icrate=0; icrate<52; icrate++ )
```

```
    {
      if( ptrtko->temptko[icrate]>45.0 )  /* High temp. > 45deg. */
      /* if( ptrtko->temptko[icrate]>38.0 )*/ /* For test. */
      {
printf(
"High temp. %4.1f in TKO %d (0-35:barrel 36-43:Fwd 44-51:Bwd)\n",
       ptrtko->temptko[icrate], icrate );
idisable = ctrlines[icrate];
ireturn = 100+icrate;
switch( idisable )
{
case 0:
  Send( 0, 9, "OPEN 6000", 9L, DABend);
  if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
  else{printf("Barrel TKO station 0 disabled.\n");}
  break;

case 1:
  Send( 0, 9, "OPEN 6001", 9L, DABend);
  if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
  else{printf("Barrel TKO station 1 disabled.\n");}
  break;

case 2:
  Send( 0, 9, "OPEN 6002", 9L, DABend);
  if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
  else{printf("Barrel TKO station 2 disabled.\n");}
  break;

case 3:
  Send( 0, 9, "OPEN 6003", 9L, DABend);
  if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
  else{printf("Barrel TKO station 3 disabled.\n");}
```

```
  break;

case 4:
  Send( 0, 9, "OPEN 6004", 9L, DABend);
  if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
  else{printf("Barrel TKO station 4 disabled.\n");}
  break;

case 5:
  Send( 0, 9, "OPEN 6005", 9L, DABend);
  if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
  else{printf("Barrel TKO station 5 disabled.\n");}
  break;

case 6:
  Send( 0, 9, "OPEN 6006", 9L, DABend);
  if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
  else{printf("Barrel TKO station 6 disabled.\n");}
  break;

case 7:
  Send( 0, 9, "OPEN 6007", 9L, DABend);
  if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
  else{printf("Barrel TKO station 7 disabled.\n");}
  break;

case 8:
  Send( 0, 9, "OPEN 6008", 9L, DABend);
  if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
  else{printf("Fwd TKO(FE0-FE3) disabled.\n");}
  break;

case 9:
```

```
  Send( 0, 9, "OPEN 6009", 9L, DABend);
  if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
  else{printf("Fwd TKO(FE4-FE7) disabled.\n");}
  break;

case 10:
  Send( 0, 9, "OPEN 6010", 9L, DABend);
  if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
  else{printf("Bwd TKO(FE0-FE3) disabled.\n");}
  break;

case 11:
  Send( 0, 9, "OPEN 6011", 9L, DABend);
  if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
  else{printf("Bwd TKO(FE4-FE7) disabled.\n");}
  break;
}
/* Display status when interlock asserted. */
Send( 0, 9, "CLOSE? 6000-6011", 16L, DABend);
if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
for( ich=0; ich<12; ich++ )
{
  Receive( 0, 9, buffer, 8L, STOPend);
  if( ibsta & ERR ){gpiberr("Receive error"); ireturn=-2;}
  status[ich] = atoi( buffer );
}
printf("status= %d %d %d %d %d %d %d %d %d %d %d %d\n",
       status[0], status[1], status[2], status[3], status[4],
       status[5], status[6], status[7], status[8], status[9],
       status[10], status[11]);

      }
      if( ptrtko->p15v[icrate]>16.0 ) /* Abnormal volt. in +15V */
```

```
      {
printf(
"Abnormal volt. %4.1f in TKO %d (0-35:barrel 36-43:Fwd 44-51:Bwd)\n",
       ptrtko->p15v[icrate], icrate );
ireturn = 100+icrate;
switch( idisable )
{
case 0:
  Send( 0, 9, "OPEN 6000", 9L, DABend);
  if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
  else{printf("Barrel TKO station 0 disabled.\n");}
  break;

case 1:
  Send( 0, 9, "OPEN 6001", 9L, DABend);
  if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
  else{printf("Barrel TKO station 1 disabled.\n");}
  break;

case 2:
  Send( 0, 9, "OPEN 6002", 9L, DABend);
  if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
  else{printf("Barrel TKO station 2 disabled.\n");}
  break;

case 3:
  Send( 0, 9, "OPEN 6003", 9L, DABend);
  if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
  else{printf("Barrel TKO station 3 disabled.\n");}
  break;

case 4:
  Send( 0, 9, "OPEN 6004", 9L, DABend);
```

```
  if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
  else{printf("Barrel TKO station 4 disabled.\n");}
  break;

case 5:
  Send( 0, 9, "OPEN 6005", 9L, DABend);
  if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
  else{printf("Barrel TKO station 5 disabled.\n");}
  break;

case 6:
  Send( 0, 9, "OPEN 6006", 9L, DABend);
  if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
  else{printf("Barrel TKO station 6 disabled.\n");}
  break;

case 7:
  Send( 0, 9, "OPEN 6007", 9L, DABend);
  if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
  else{printf("Barrel TKO station 7 disabled.\n");}
  break;

case 8:
  Send( 0, 9, "OPEN 6008", 9L, DABend);
  if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
  else{printf("Fwd TKO(FE0-FE3) disabled.\n");}
  break;

case 9:
  Send( 0, 9, "OPEN 6009", 9L, DABend);
  if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
  else{printf("Fwd TKO(FE4-FE7) disabled.\n");}
  break;
```

```
case 10:
  Send( 0, 9, "OPEN 6010", 9L, DABend);
  if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
  else{printf("Bwd TKO(FE0-FE3) disabled.\n");}
  break;

case 11:
  Send( 0, 9, "OPEN 6011", 9L, DABend);
  if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
  else{printf("Bwd TKO(FE4-FE7) disabled.\n");}
  break;
}

/* Display status when interlock asserted. */
Send( 0, 9, "CLOSE? 6000-6011", 16L, DABend);
if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
for( ich=0; ich<12; ich++ )
{
  Receive( 0, 9, buffer, 8L, STOPend);
  if( ibsta & ERR ){gpiberr("Receive error"); ireturn=-2;}
  status[ich] = atoi( buffer );
}
printf("status= %d %d %d %d %d %d %d %d %d %d %d %d\n",
       status[0], status[1], status[2], status[3], status[4],
       status[5], status[6], status[7], status[8], status[9],
       status[10], status[11]);

      }
    }
    break;

  /*-------------------------
```

```
 * CASE -1: Disable TKO electronics.
 *-------------------------*/
case -1:
  /* Disable all TKO station one by one with 3 sec interval */
  Send( 0, 9, "OPEN 6000", 9L, DABend);
  if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
  else{printf("Barrel TKO station 0 disabled.\n"); sleep(3);}

  Send( 0, 9, "OPEN 6001", 9L, DABend);
  if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
  else{printf("Barrel TKO station 1 disabled.\n"); sleep(3);}

  Send( 0, 9, "OPEN 6002", 9L, DABend);
  if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
  else{printf("Barrel TKO station 2 disabled.\n"); sleep(3);}

  Send( 0, 9, "OPEN 6003", 9L, DABend);
  if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
  else{printf("Barrel TKO station 3 disabled.\n"); sleep(3);}

  Send( 0, 9, "OPEN 6004", 9L, DABend);
  if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
  else{printf("Barrel TKO station 4 disabled.\n"); sleep(3);}

  Send( 0, 9, "OPEN 6005", 9L, DABend);
  if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
  else{printf("Barrel TKO station 5 disabled.\n"); sleep(3);}

  Send( 0, 9, "OPEN 6006", 9L, DABend);
  if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
  else{printf("Barrel TKO station 6 disabled.\n"); sleep(3);}

  Send( 0, 9, "OPEN 6007", 9L, DABend);
```

```
 if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
 else{printf("Barrel TKO station 7 disabled.\n"); sleep(3);}

 Send( 0, 9, "OPEN 6008", 9L, DABend);
 if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
 else{printf("Fwd CsI TKO(FE0-FE3) disabled.\n"); sleep(3);}

 Send( 0, 9, "OPEN 6009", 9L, DABend);
 if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
 else{printf("Fwd CsI TKO(FE4-FE7) disabled.\n"); sleep(3);}

 Send( 0, 9, "OPEN 6010", 9L, DABend);
 if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
 else{printf("Bwd CsI TKO(BE0-BE3) disabled.\n"); sleep(3);}

 Send( 0, 9, "OPEN 6011", 9L, DABend);
 if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
 else{printf("Bwd CsI TKO(BE4-BE7) disabled.\n"); }

 /* Display channel status after desable instruction complete. */
 Send( 0, 9, "CLOSE? 6000-6011", 16L, DABend);
 if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
 for( ich=0; ich<12; ich++ )
 {
   Receive( 0, 9, buffer, 8L, STOPend);
   if( ibsta & ERR ){gpiberr("Receive error"); ireturn=-2;}
   status[ich] = atoi( buffer );
 }
 printf("status= %d %d %d %d %d %d %d %d %d %d %d %d\n",
status[0], status[1], status[2], status[3], status[4],
status[5], status[6], status[7], status[8], status[9],
status[10], status[11]);
```

```
    break;
  }
  /* Check status of all the channels before leaving. */
  /* Just for debug */
  /*  Send( 0, 9, "CLOSE? 6000-6011", 16L, DABend);
  if( ibsta & ERR ){gpiberr("Send error"); ireturn=-1;}
  for( ich=0; ich<12; ich++ )
  {
    Receive( 0, 9, buffer, 8L, STOPend);
    if( ibsta & ERR ){gpiberr("Receive error"); ireturn=-2;}
    status[ich] = atoi( buffer );
  }
  printf("status= %d %d %d %d %d %d %d %d %d %d %d %d\n",
 status[0], status[1], status[2], status[3], status[4],
 status[5], status[6], status[7], status[8], status[9],
 status[10], status[11]);
  */
  return ireturn;
}
```

参考文献

[1] Phys.Rev.Lett.vol.13(1964)138-140pp

[2] M.Kobayashi and T.Maskawa ,Prog.Theor.Phys.**42**(1973)652.

[3] A.B.Carter and A.I.Sanda,Phys.Rev.Lett.45(1980)952,Phys. Rev.D23(1981)1567.I.I.Bigi and A.I.Sanda,Nucl.Phys.B193(1981)85.

[4] L.Wolfenstein , Phys,Rev.Lett.**51**(1983)1945.

[5] 真木晶弘 高エネルギー物理学実験

[6] 黒川真一 et al, 日本物理学会誌 Vol49,No9,(1994)718

[7] 横井恵美,BELLE 検出器用 CsI 電磁カロリメータにおけるトリガーシステムの設計・製作, 平成 9 年度修士学位論文

[8] 鈴木史郎 et al, 日本物理学会誌 Vol49,No9,(1994)718

[9] Progress Report

[10] 高田明子,BELLE 測定器用における環境モニターシステムの開発, 平成 9 年度修士学位論文

[11] （株）横河・ヒューレット・パッカード HP3852A 操作ガイド

[12] M.Nakao,NSM User's manual(for version 1.002),1998.6.17

[13] NI-488.2M Software Reference Manual,1996

謝辞

本研究を行うにあたり、お世話になりました皆様方に心からの感謝を申し上げます。

常に私を叱咤激励し正しい方向へと導いて下さった野口先生、研究全般において重要で的確なアドバイスを下さった林井先生には大変感謝致します。そして、実に最後まで根気強く、直接御指導いただいた宮林先生には実験に関わる様々な知識と共に研究に対する姿勢というものを教えて頂きました。先生方の御心労を思うと感に堪えません。本当にありがとうございました。

また、KEK においてCsI カロリメータグループの皆様には多大な御助言、御協力を頂き誠にありがとうございました。実験が始まろうとするこの時期に KEK-B ファクトリーという素晴らしきプロジェクトに参加することが出来た事を、心より誇らしく思います。

そして敬愛する先輩方、奈良においては足立先輩、KEK においては池田先輩、お二人には勉学・研究に関する沢山の事をいつも優しく、丁寧に教えて頂き、感謝のあまりお礼の申しようもありません。本当に、お世話になりました。また足立先輩をはじめ、研究室 4 回生の浅井さん、池田さん、岡崎さん、賀来さん、桑原さん、杉本さんには KEK にて多くの製作作業を手伝って戴きました。皆様のおかげで作業進行のスピードが上がったことは言うまでもありません。本当に、ありがとうございました。

最後に、二年間私を励まし支えて下さったすべての方々に深く感謝を申し上げます。