JAERI-Tech
99-082

原研軟X線ビームライン(BL23SU)用挿入光源の制御系の開発

1999年12月

平松洋一・島田太平*・宮原義一*

日本原子力研究所
Japan Atomic Energy Research Institute

本レポートは，日本原子力研究所が不定期に公刊している研究報告書です。
入手の問合わせは，日本原子力研究所研究情報部研究情報課（〒319－1195 茨城県那珂郡東海村）あて，お申し越しください。なお，このほかに財団法人原子力弘済会資料センター（〒319－1195 茨城県那珂郡東海村日本原子力研究所内）で複写による実費頒布をおこなっております。

This report is issued irregularly.
Inquiries about availability of the reports should be addressed to Research Information Division, Department of Intellectual Resources, Japan Atomic Energy Research Institute, Tokai-mura, Naka-gun, Ibaraki-ken 〒319－1195, Japan.

編集兼発行　日本原子力研究所

原研軟X線ビームライン(BL23SU)用挿入光源の制御系の開発

日本原子力研究所関西研究所放射光利用研究部
平松　洋一※・島田　太平*・宮原　義一*

(1999年 11月 1日受理)

大型放射光施設 SPring-8 の 23 番セルに設置された原研軟X線ビームライン用挿入光源の制御系を開発した。この挿入光源は平面型可変偏光アンジュレータ(APPLE型）であり、上下に設置された 2 対の磁石列を相対的に動かして、水平直線偏光、垂直直線偏光、楕円偏光、左右円偏光の放射光を発生する装置である。上下磁石列間のギャップを変える（ギャップを駆動する）ことにより、軟X線領域において異なるエネルギーの放射光を発生することができる。本装置の最大の特徴は位相を周期的に動かして（位相駆動して）左右円偏光を 0.5 Hz の高速で切り替えられることである。本制御系では、磁石列のギャップ駆動と位相駆動に伴って、蓄積リングの電子ビーム軌道が変動するのを抑制するために、10 台の補正電磁石で早い（励磁周期 24 msec (42 Hz)の）軌道補正をかけることができる。放射光実験では磁石列の位相駆動と同期したデータの収集が必要となることから、一定の周期（周期 2 sec で、ばらつき誤差 0.1 % 以下）の位相駆動を実現することにも成功した。開発したシステムは SPring-8 全体の制御系で採用されている『SVOCコマンド制御方式』に合致したものになっている。

関西研究所：〒679-5143 兵庫県佐用郡三日月町光都1-1-1
※　外来研究員：石川島播磨重工業株式会社
*　（財）高輝度光科学研究センター

Control Software of a Variably Polarizing Undulator (APPLE Type) for SX Beamline in the SPring-8

Yoichi HIRAMATSU※, Taihei SHIMADA* and Yoshikazu MIYAHARA*

Department of Synchrotron Radiation Research

Kansai Research Establishment

Japan Atomic Energy Research Institute

Mikazuki-cho, Sayo-gun, Hyogo-ken

(Received November 1, 1999)

This paper describes the control software of a variably polarizing undulator (APPLE Type) that was installed at the SX beamline (cell number 23) in the SPring-8 storage ring in February, 1998. This undulator produces a polarized radiation in the energy range of soft X-ray by changing the gap distance between two pairs of permanent magnet arrays (gap movement). The main characteristic of the undulator is a capability to generate right and left circular polarization alternately at a period of 2 sec (0.5 Hz) by high speed phase-shifting (periodic phase movement). The developed software makes a fast correction of the closed orbit distortion (COD) of an electron beam by exciting steering magnets at a rate of time interval of 24 msec (42 Hz) during the movement of magnet arrays. Also, the software is capable to put these magnet arrays into a constant periodic phase movement with an error less than 0.1 % for the period of 2 sec. The software was developed in accordance with the directions of SPring-8 standard for software development.

Keywords: Control System, Software, SVOC, EM, EMA, Computer, VME, Variably Polarizing Undulator, SPring-8, Soft X-ray, Gap-distance, Phase-shift, Steering Magnet, Circular Polarization, Linear Polarization

---

※ On leave from Ishikawajima-Harima Heavy Industries CO., Ltd.

* Japan Synchrotron Radiation Research Institute

目　次

Contents

1. 概　要

SPring-8 蓄積リングの２３番セルの直線部に設置された原研軟Ｘ線ビームライン用挿入光源の制御系を開発した。本挿入光源は、軟Ｘ線領域で種々の偏光状態の放射光を発生することのできる APPLE (Advanced Planar Polarized Light Emitter) 型の可変偏光アンジュレータであり、原研専用の軟Ｘ線ビームライン BL23SU に放射光を供給する。本アンジュレータでは、左右円偏光を最大 0.5Hz で切り換えることができる。

本制御系では、上下２対の磁石列のギャップ駆動と位相駆動に伴って蓄積リングの電子ビーム軌道が変動するのを抑制するために、１０台の補正電磁石で速い（励磁周期 = 24 msec (42 Hz) の）軌道補正をかけることができる。また、磁石列の位相を一定の周期（周期 2sec で、ばらつき誤差 0.1% 以下）で駆動することも可能である。開発したシステムは、SPring-8 全体の制御系で採用されているＳＶＯＣコマンド制御方式に合致したものになっている。

ここでは、本アンジュレータの制御系の開発について、平成１１年９月現在の状態を報告する。

2. 制御系の仕様

以下に制御系の開発で目標とした仕様を示す。

Closed orbit distortion (COD)	Δx= 5 micron, Δy= 1 micron
Device type	APPLE-2 (Halback type)
Total length	1920 mm
Permanent magnet arrays	row1, row2, row3, row4
Radiation energy	0.15 keV to 4.5 keV (first harmonics)
Polarization	Vertical, Horizontal, Circular and Elliptical
Minimum gap-distance	36.0 mm
Maximum gap-distance	300.0 mm (full-open)
Minimum phase position	z = -60.0 mm
Maximum phase position	z = 60.0 mm
Phase of horizontal polarization	D = -120.0 mm, 0.0 mm, 120.0 mm
Phase of vertical polarization	D = -60.0 mm , 60.0 mm
Number of steering magnets	5 units for x-direction 5 units for y-direction

座標系 (Fig.1 を参照) :
x 軸：水平方向。 (蓄積リングの中心から発散する方向を正とする)
y 軸：ギャップ駆動方向。 (ギャップの開く方向が正)
z 軸：電子ビーム進行方向、或いは位相駆動方向。 (電子ビーム下流方向が正)

ここで D は、水平方向に隣接する磁石列の z 方向のずれ量であり、位相差(phase_shift)と呼ぶ。z 方向に４つの磁石列の端がそろった状態が基本状態であり、D = 0.0 mm である。
D = (upper_phase_position) - (lower_phase_position) と定義する。よって、位相差のとる範囲は、D = -120.0 mm から 120.0 mm である。

位相駆動における磁石列折り返し点のオーバーシュート： 0.1 mm 以下
位相駆動周期のバラツキ誤差： 0.1% 以下 (周期 = 0.5 sec (2 Hz) で動かした場合)

3. 制御系ハードウェアの構成

Fig.2に示すように、制御系は、ＶＭＥバス規格のオンボード計算機(HP743rt/100MHz, 64Mbyte RAM, 20Mbyte Flash-ROM) と各種のＩ／Ｏボードで構成される。ＯＳは、リアルタイム制を持ったHP-RT version 2.21であり、これはFlash-ROMに格納されてある。コンピュータ２０００年問題用のパッチはすでに当てており、対応済みである。計算機は、Flash-ROM を使ってブート（自動起動）できる。ブートは、ネットワーク経由で中央制御室のサーバーからも可能であるが、Flash-ROM経由で立ち上げることになっている。

計算機のＩ／Ｏボード（Table 1を参照）は、制御の対象であり、一方のソフトウェアは計算機ＣＰＵとＩ／Ｏボードを組織的に結ぶ手段である。この組織的構造としては、一般的に種々のものが考えられるが、SPring-8では、「計算機にSVOCコマンド（後述する）を入力して、コマンドを解釈させて、Ｉ／Ｏボードをアクセスさせる」ための手段を、ソフトウェアＥＭ(Equipment Manager、後述）、あるいはＥＭＡ（Equipment Manager Agent、後述）という名称で定義してある。ソフトウェアＥＭとＥＭＡの考え方では、制御対象のＩ／Ｏボードは「デバイス(device)」と呼ばれる。そして、Ｉ／Ｏボードに（アナログあるいはディジタルの）信号を入出力するスイッチ類、ＰＬＣ（シーケンサ），モータのドライブ・アンプ、電磁石電源等は、Ｉ／Ｏボードの延長という考え方になり、これらは「機器（equipment）」と呼ばれる。ＥＭ（Equipment Manager）という名称は、ここに由来する。したがって、計算機ＣＰＵで実行するＥＭとＥＭＡは、Ｉ／Ｏボード（device）をアクセス管理しながら、機器(equipment)を間接的に制御するソフトウェアである。

3-1 計算機Ｉ／Ｏ

制御計算機は、Fig.3に示すように、蓄積リングの保守通路に設置された一連の制御盤の１つに組み込まれてある。計算機のＩ／Ｏボードとして、次の７種類を採用した。（Fig.4を参照。右端からＣＰＵ、ＡＤＣボード、ＰＴＧボードの順）

(a) ＡＤＣボード：
電子ビーム位置検出系のアナログ信号（８チャネル、±10.0Ｖフルスケール）を取り込み、１６ビットのディジタル信号へ変換する。ここで、10.0 Ｖは0xFFFF(65535)に、0.0 Ｖは0x8000(32768)に、そして-10.0 Ｖは0x0000(0)に対応する。

(b) ＰＴＧボード：
ギャップ駆動用パルスモータと位相駆動用ＡＣサーボモータのために２枚を使用する。モータのドライブ・アンプを制御するためのパルス列を発生する。

(c) ＤＩボード：
リミットスイッチ類やロータリエンコーダ（ギャップの読み値）、イオンゲージコントローラ、イオンポンプコントローラ等の信号を取り込む。

(d) ＴＴＬＤＩボード：

リニアスケール（位相差の読み値）の信号を取り込む。

(e) ＤＯボード：

ロータリエンコーダとリニアスケールへプリセット信号を送る。

(f) ＴＴＬＤＩＯボード：

ロータリエンコーダとリニアスケールへラッチ信号を送る。

(g) ＲＩＯボード：

Remote I/O(Type-A)の略であり、計算機側をマスタ局、補正電磁石電源側をスレーブ局と呼ぶ。マスター局で設定した励磁電流値データを１０台のスレーブ局へ光ファイバー経由で伝える。マスター局のデータ更新周期は、2.1 msec(476 Hz)である。スレーブ局のアナログ出力部（ＤＡＣ）の電圧設定値とアナログ入力部（ＡＤＣ）のデータを１６ビットの解像度で読む。

3-2 センサー

計算機Ｉ／Ｏボードの入力となる信号源は、簡単に「センサー類」とした。次の４種類がある。

(a) リミットスイッチ：

磁石列の原点位置を知らせるスイッチやオーバートラベル用、減速スイッチ、ギャップ・フルオープン状態を知らせるスイッチ等をいう。

(b) ＰＬＣ(Programmable Logic Controller、シーケンサ)：

Fig.5とFig.6に示すように、２台のＰＬＣを使用する。１台目は、リミットスイッチの作動電圧(5 V,12 V,24 V)とＩ／Ｏボードの作動電圧(5 V,24 V)のレベル合わせのために使用する。緊急停止の回路もこの１台目のＰＬＣで組まれている。２台目は、ロータリエンコーダで読み取ったギャップと位相位置のビット列情報（グレイ・バイナリ・コード）をＢＣＤへ変換するために使用する。

(c) ロータリエンコーダ（絶対値型）：

ギャップと位相の位置を読み取るために使用する。位置を示す１０ビット幅の情報は、「グレイ・バイナリ・コード」で出力される。グレイ・バイナリ・コードは、エンコーダのディスクが回転中にビット・エラーを低減する目的で、ロータリエンコーダではよく使われる方法である。

(d) リニアスケール：

ギャップと位相の位置を読み取るために使用する。ロータリエンコーダに比べてリニアスケールの応答性は良く、読み取り精度も高い。よって、ソフトウェアでは、リニアスケールの読み値を採用している。ロータリエンコーダは、モータ付近の回転数を測定するが、一方のリニアスケールは、磁石列付近でギャップと位相の位置を直接読み取る。２種類の方法を使うことの狙いは測定の信頼性を高めることである。

3-3 ギャップ駆動と位相駆動用モータ

ギャップ駆動には、１パルスで 0.72 度を回転するパルスモータを使っている。ソフトウェア側で必要となるモータ駆動のパラメータは、最高パルス発生レート(pulse/sec)、最低パルス発生レート(pulse/sec)、パルス発生レートの変化率(microsec)である。これら３つのパラメータは、デバイス構成表で設定される。アンジュレータの運転中、デバイス構成表を編集してパラメータを変更することは可能である。

一方、位相駆動には、上側と下側の磁石列用に２台の電磁ブレーキ付きのＡＣサーボモータを使用する。１パルスで 0.18 度を回転する。ギャップ駆動と同じように、ソフトウェア側で必要となるモータ駆動のパラメータは、最高パルス発生レート(pulse/sec)、最低パルス発生レート(pulse/sec)、パルス発生レートの変化率(microsec)である。パラメータはデバイス構成表で設定されており、アンジュレータ運転中にパラメータ変更は可能である。電磁ブレーキは、位相駆動の停止時に効かす。

3-4 補正電磁石用電源

本アンジュレータでは、電子ビームのＣＯＤ（Closed Orbit Distortion）を補正するために、１０台の補正電磁石（ｘ方向とｙ方向で、５対）を使っている。電源はバイポーラ型であり、最大の出力電流値は 15.0 A(-15.0 A)、最大電圧値は 10.0 V(-10.0 V)である。電源の立ち上がり時間（0.0 A の通電状態から 15.0 A を通電するまでの時間）は、5 msec である。RIO マスター局のレジスター（１６ビット）に設定された電流値は、データ更新周期 2.1 msec（476 Hz）で電源側のスレーブ局へ転送される。データ更新周期 2.1 msec は、１０台の補正電磁石用電源を使った場合の値であり、電源の数が減ればこの更新周期は短くなる。電流値は１６ビットの精度で設定できるから、15.0 A は 0xFFFF(65535)、0.0 A は 0x8000(32768)、そして-15.0 A は 0x0000(0)に対応する。電流値を１６ビットのレジスターに設定するときの正確な定義では、0.0 A は 0x8000 と 0x7FFF の中間に当たる。しかし、RIO(Type-A)の性能誤差は、フルスケールに対して±0.005%であるために、カウント数 65535 の範囲で±３カウント分のズレは、実際ほとんど問題にならない。よって、ここでは設定電流値 0.0 A は 0x8000(32768)に対応するものと決める。

4. ソフトウェアの開発

4-1 SVOCコマンドとEM、EMA

SPring-8 の加速器とビームラインの制御系では、「SVOC コマンドとEMによる制御方式」が採用されている。ＳＶＯＣとは、英文構成要素の Subject, Verb, Object, Complement の頭文字を組み合わせた SPring-8 独自の名称であり、ビーム・ユーザーがアンジュレータの制御対象と「会話したい」ときに発令するコマンドを指す。例えば、磁石列を(a)現在のギャップを知りたい、或いは(b)円偏光の位相差に変えたいときには、(a)s/get/bl_id23_gap/position 、(b)s/put/bl_id23_phase/32.0mm というコマンドをビーム・ユーザー側のＧＵＩ端末から発令する。発された SVOC コマンドは、中央制御室内で動作中のＭＳ(Message Server)とＡＳ(Access Server)というコマンド管理サーバープロセスを介して、ＬＡＮネットワーク経由で現場のＶＭＥ制御計算機へ伝えられる。一方のＥＭ(Equipment Manager)とは、ソフトウェア（実行コード）の名称であるとともに、計算機ＣＰＵに存在する制御サーバー・プロセスの名称でもある。このＥＭは、SVOC コマンドの到着点である。（Fig.7 を参照）。

ＧＵＩから文字列というデータ型で送られてきた SVOC コマンドは、ＥＭプロセスによって具体的な制御処理に変えられる。すなわち、デバイス・ドライバーを呼び出すための関数ＡＰＩ(Application Program Interface)を使って、計算機Ｉ／Ｏボードをアクセスする仕組みになっている。上記の例でみると、put のときＥＭは、デバイス・パラメータをＡＰＩへ渡し、逆に get のときは、ＡＰＩ経由でＩ／Ｏボードのビット列情報を取り込む。ＡＰＩの呼び出しが正しく行われるとＥＭは単語「ok」をコマンドの発信元へ返す。ＡＰＩの呼び出しが失敗ならば、「fail」を返す。また、例えば「123.0mm」というギャップ読み値のビット情報の場合には、ＥＭはこの戻り値を文字列でコマンドの発信元へ返す。

SPring-8 のような加速器複合設備において、制御系に対する要求は、多種多様である。つまり、制御の対象となる機器は多種類あり、ビーム・ユーザーは多様な使い方を求めてくる。しかも、機器を制御するためのアプリケーション・プログラムの追加やデバイス・パラメータの変更は、迅速に作業できなければならない。こうした必要性が根拠となって設計されたのものが「SVOC コマンドとＥＭによる制御方式」である。蓄積リングの建設中、この方式に則って様々なアプリケーション・プログラムが開発されていく過程で、機器制御の１部の機能（繰り返し制御の機能）をＥＭから切り離したいというユーザーからの要求がきっかけとなって、ＥＭＡ(Equipment Manager Agent)と呼ばれる制御サーバー・プロセスの概念が生まれたようである。ＥＭに主導権を持たせるために、ＥＭＡは、ＥＭの子プロセス(child process)としてUNIX システムコールの fork & exec で生成される仕組みになっている（Fig.8 を参照）。

SVOCコマンドとＥＭ（ＥＭＡ）による制御方式には、以下の特徴がある。

(a) 新たな制御機能は、簡単に追加でき、不要な機能は削除できる。

(b) 計算機及び関連のハードウェアは、簡単に取り替えできる。

(c) ＥＭは、上位計算機から送られてくるSVOCコマンドを受け付けて処理するだけで、ＥＭ内部で発行されたSVOCコマンドをも受け付けて処理することができる。この方法は、SVOCコマンドの再帰呼び出しと呼ばれる。

(d) 繰り返し性を要する制御機能は、ＥＭＡというプロセス名（複数可）で独立させることができる。

(e) ＥＭプロセス内のタイム・スイッチングは、ＯＳに任せることができる。

本制御系のソフトウェアは、主プロセスであるＥＭと周期的位相駆動を受け持つ専用プロセスＥＭＡで構成される。

4-2　ソフトウェアの開発手順

ＥＭとＥＭＡは、装置の技術グループの手によって作るものである。以下にＥＭとＥＭＡの実行コードが出来上がるまでの過程を示す。

(a) 制御系ハードウェアとビーム・ユーザーの要求を分析して、制御要求を一連の「単体動作」に整理する。

(b) 単体動作をＳＶＯＣコマンドの形式で表現する。本制御系では、６７個のＳＶＯＣコマンドがある。

(c) 各ＳＶＯＣコマンドに対し、コマンドをＥＭで実行するために必要となるアプリケーション・プログラム名とデバイス名（複数可）、デバイス・パラメータ（複数可）を定義する。１つのデバイス構成表(configuration table)にまとめる。

(d) アプリケーション・プログラムとヘッダー・ファイルを作る。（Ｃ言語で開発する）

(e) 補正電磁石の励磁テーブルを作る。（最初は、ダミー電流値でよい）

(f) メークファイルを作る。ここで、tellem*（単体テスト用ＥＭの実行コード）とcma_rt*（単体テスト用ＥＭＡの実行コード）を作る。

(g) 現場の制御計算機（ホスト名：id23rt）にログインして、tellem*を用い、全てのSVOCコマンドを発し、アンジュレータ単体動作の確認試験を実施する。予想外の動作を回避するために、get/bl_id23_...コマンドの戻り値を確認してから、put/bl_id23_...を実行するという順番で試験を進める。

(h) データベース用のアラーム値を定義する。

(i) rpc_em*（ＥＭの実行コード）と ema_rt*（ＥＭＡの実行コード）、pc_po_main*（poller-collectorの実行コード）を作る。これらは、ＲＰＣ経由である。

(j) 最後に、SVOCコマンドをＧＵＩに結び付けて、ＥＭとＥＭＡの開発は終了する。

4-3 開発環境

デバイス構成表が出来上がったら次は、アプリケーション・プログラムのコーディングである。全てのSVOCコマンドに対応したプログラムが必要になる。SPring-8は、プログラム開発専用のUNIXファイル・サーバーを持っており、これには現場のVME計算機用の実行コードを作るための開発環境が備わっている（Fig.9を参照）。他技術グループのソフトウェアも「SVOCコマンドとEM」という枠組みで開発されるために、技術グループ間で共有することのできる標準ライブラリが多数準備されている。本制御系のＥＭとＥＭＡの開発では可能な限り、SPring-8の標準ライブラリを取り入れた。

4-4 ＳＶＯＣコマンドとデバイス構成表

SVOC コマンドは、１つの機器から１つの動作（単体動作）を取り出すための記述である。例えば、「１０台の補正電磁石に3.0Aを、同時に流したい」のSVOCコマンドは、次のようになる。先ずは、コマンド「set/bl_id23_st_h_1/3.0A」を発行する。コマンドは主語(subject)で始まるはずであるが、主語はＭＳ(Message Server)プロセスで自動的に追加される決まりになっている。目的語"bl_id23_st_h_1"は、「ビームライン・グループ、セル２３番の挿入光源、電子ビームを水平方向へキックするための１番目の補正電磁石」を意味する。次に、２番目と３番目、４番目、５番目の水平方向を設定し、同じように垂直方向も設定する。その次に、「put/bl_id23_st/exec」を発行して１０台の補正電磁石を同時に励磁する。このように、SVOCコマンドは１つの単体動作を示すものであり、多種類のコマンドを再帰で呼び出しすれば、複雑な制御を実現することが可能である。全体システムをどのように制御したいかでコマンドの種類と数は異なるものだが、本制御系では６７個のコマンドを準備した。

一方、デバイス構成表(configuration table)は、全てのSVOCコマンドを一定の秩序で並べた文字列型のファイル、つまりＥＭとＥＭＡがコマンドを解釈するための参照ファイルである。ＥＭは、ある１つのSVOCコマンドを受け取ったとき、デバイス構成表を参照して、コマンドを実行するために必要となるアプリケーション・プログラム名とアクセスしなければならないデバイス名、デバイスの作動条件（パラメータ）等の情報を得る。

また、１つのSVOCコマンドでは複数のデバイスを呼び出すこともできる。例えば、ギャップ駆動中、ある瞬間に、get/bl_id23_gap/positionを発行してギャップ値を読みたいとする。ＥＭは先ず、リニアスケールにラッチをかける。次に、ギャップを読み取って、ラッチを外す。このように、ラッチをかけるためとギャップを読むための２種類のデバイスがアクセスされる。ギャップ用と上側位相用、下側位相用の３つのリニアスケールを使っているために、ギャップ用は、Ｉ／Ｏボードの何チャネル目に相当するかという情報が必要になってくる。このチャネル番号は、ＥＭへ渡されるべきデバイスのパラメータとして、デバイス構成表に記述される。

4-5　EMプロセス

EMは、一連のアプリケーション・プログラムを「EMフレーム」と呼ばれるソフトウェア・テンプレートに組み込んだときに出来上がる実行コードである。この実行コードをデバイス構成表と励磁テーブルと共に、制御計算機（ホスト名：id23rt）で実行した結果、生成されるＣＰＵプロセスをＥＭプロセスという。

Fig.10に示すように、ＥＭ実行コードには、(a) tellem* と呼ばれるテスト用のＥＭと、(b) 実際のSVOCコマンドをＲＰＣ(Remote Procedure Call)経由で受け付けることのできる rpc_em* の２種類がある。ＥＭが初めて出来上がるか、或いは改造されると、先ずは tellem* を用いてSVOCコマンドをテストしてみるという習慣になっている。全てのSVOCコマンドが予定どおりの戻り値を返せば、次のステップの rpc_em* は間違いなく動いてくれるはずである。UNIX ファイルサーバーで開発された tellem* は id23rt のディレクトリ /prj/bin/id23rt/test/ へコピーして、実行することができる。同ディレクトリには、tellem* の他に、デバイス構成表と励磁テーブルもコピーしておかなければならない。ＥＭＡについては後述するが、ＥＭＡ(ema_rt*)もこの tellem* と同じディレクトリにコピーしておいて実行する。id23rt コンソール画面のプロンプト行で tellem* とキーインすれば、EMは自動的にデバイス構成表と励磁テーブルを読み込み、自身の初期化処理を開始する。初期化を無事に通過したらＥＭプロセスは、コンソール画面にSVOCコマンドの入力行を表示し、SVOCコマンドを受け付けられる段階に入る。終了は、コマンド入力行で exit とキーインすれば、ＥＭは自動的に終了処理を始める。

一方、実際のＥＭである rpc_em* に関しては、これは制御グループによって常時ランニング状態に置かれている。id23rt をリブートしたとき、或いは停電（ミリ秒オーダーの瞬時停電を含む）が復旧されたときにでも、ＥＭは電子ビームの軌道に悪影響を与えることなく、停電する前のランニング状態で初期化処理を実行する。すなわち、ＥＭはホット・スタートが可能である。これは、ＲＩＯのマスターカードがリブート時或いは停電時の以前の励磁電流値を覚えているためである。

4-6 EMAプロセス

EMAは、既に述べたEMフレームとは別に、ソフトウェア・テンプレート「EMAフレーム」の構成要素である関数 em_ema_loop_sequence.c をカスタマイズして作る実行コードである。EMA実行コードは、EM実行コードと同じディレクトリに格納して実行するものである。Fig.11に示すように、EMAプロセスはEMプロセスの「子プロセス」であるから、EMAプロセスの初期化の振る舞いは、EMの初期化と一致する。両者の違いは、SVOC コマンドを受け付けるための通信ポートの違いにある。親プロセスのEMは、既に述べたように、RPC経由でSVOC コマンドを受け付けるが、子プロセスのEMAは、メッセージ・キューを介して親プロセスと通信する。

1つのCPUには、EMとEMA、そして Poller-Collector (SPring-8 のデータ収集用プロセス) の3つのプロセスが共存するために、1プロセスがランニング状態のときに、他プロセスは「待機状態」となる。しかし、OSは各々のプロセスに優先権を与えることができて、SPring-8の設定では、EMの優先度を最も高くしており、10 秒の間隔で SVOC コマンドを実行する Poller-Collector を最も低くしてある。よって、EMのランニング時、他プロセスはCPUに介入する余地は無いことになるが、EMがCPU資源を終始100%独占することは無い。10 秒間隔で起動する Poller-Collector の実行時間は (get/.../... だけの実行であるから) 極めて短いため、Poller-Collector はEMプロセスに悪影響を与えることは皆無である。

では、EMAはどうだろうか。EMAは元々、あらかじめ決まった周期的な動作を受け持つプロセスであるために、EMAが実行中のとき、親プロセスのEMは待機状態にあることが多い。もしEMAのランニング状態でEMが起動された場合、EMAプロセスはEMの負荷影響を受けると考えられるが、EMとEMAフレームの設計は、こうした悪影響を回避するように設計されてある。すなわち、EMAがCPU資源を開放している瞬間 (例えば、システム・コール sleep を呼び出したとき) に、EMと Poller-Collector は起動される仕組みになっている (Fig.12 を参照)。実際、EMAプロセスによる周期的位相駆動において、駆動周期の乱れは観測されなかった。(後述するが、0.1%以下という実験結果になる)

EMAプロセスは、EMプロセスがランニング状態 (tellem* で実行) のとき、次に示す一連の SVOC コマンドで生成とパラメータ設定、起動、停止、消滅等ができる。

(a) put/bl_id23_pattern/create

この SVOC コマンドは、EMに対しEMA (子プロセス) の生成を要求する。EMは、デバイス構成表で指定された名前のEMA (例 : "ema_rt") を生成する。

(b) put/bl_id23_pattern_phase/32.0mm_-32.0mm

これは、位相駆動の駆動範囲を設定するためのコマンドである。この例では、位相差 D = 32.0 mm からもう一方の位相差 D = -32.0 mm の範囲で磁石列を動かすという意味になる。"32.0mm_-32.0mm" (数字と単位の間は、空白なし。2つのパラメータは、アンダーバーで区切る) という設定では、磁石列は位相差 D = 32.0 mm の状態で動き始める。

逆に、"-32.0mm_32.0mm"という設定では、動きは位相差D = -32.0 mmの状態で始まる。位相差 D は、-120.0 mm から 120.0 mm の範囲であれば、任意の実数値を設定できる。"32.0mm_-43.0mm"という異なる位相差を設定することも可能である。

(c) put/bl_id23_pattern_hold/200msec_200msec

これは、位相駆動中に設けるフラット部（hold-time によって磁石列が定期的に停止する部分）を設定するためのコマンドである。数字(200)と単位(msec)の間に、空白は入れない。２つのパラメータの間は、アンダーバーで区切る。１番目の 200msec は、上記の(b)を例にとるとD = 32.0 mmの磁石列位置における保持時間、２番目の200msecは、D = -32.0 mmの位置での保持時間である。hold-time は、200 msec 以上ならば、任意の整数値を設定できる。"200msec_1300msec"という異なる保持時間を設定することも可能である。

(d) put/bl_id23_pattern/start

このコマンドは、位相駆動を開始する。現在のＥＭＡでは、連続繰り返し回数は 20000（約１１時間の駆動に相当する）に設定してあるが、ソースコード（em_ema_loop_sequence.c）を編集して任意の繰り返し数を設定できる。

.....

.....

（位相駆動中であり、放射光実験はこの間に実施される。駆動は、任意の時間に停止できる）

.....

.....

(e) put/bl_id23_pattern/stop

位相駆動を停止する。磁石列の動きは、１駆動ループ（１周期）の終わりで停止する。put/bl_id23_pattern/start を再度発行すれば、駆動を再開始できる。また、この停止状態では、位相差と保持時間を再設定することが可能である。

(f) put/bl_id23_pattern/destroy

ＥＭＡを消滅する。put/bl_id23_pattern/create でＥＭＡを再度、生成することができる。

(g) get/bl_id23_pattern/status

ＥＭＡプロセスの状態を知るために発行する。Appendix Ｊで示すように、ＥＭＡは４つの状態（inactive, not-ready, ready, active）を持つ。"inactive"は、ＥＭＡがまだ生成されてないときの状態を示し、"not-ready"は、ＥＭＡの生成後、位相差(phase-shift, D)と保持時間(hold-time)がまだ設定されてないときの状態を示す。一方、"ready"は位相差と保持時間は設定済みで、位相駆動を開始(start)できる状態を示す。最後に、"active"は、位相駆動中の状態を示す。この状態で、任意の時間に位相駆動を停止(stop)することは可能である。

(h) get/bl_id23_pattern_phase/position
位相差の設定値を知るために発行する。戻り値は、32.0mm_-32.0mm（例）。位相差がまだ設定されてないと、単語"fail"が返ってくる。

(i) get/bl_id23_pattern_hold/time
保持時間(hold-time)の設定値を知るために発行する。戻り値は、200msec_200msec（例）。保持時間がまだ設定されてないと、"fail"が返ってくる。

位相駆動中、上下の磁石列の磁場によって発生する引力と反発力は、ギャップの小さいときに最大となる。このような最大の力が発生する“ギャップ最小”の条件のもとで、短い周期の位相駆動（例えば、2 sec (0.5 Hz)の繰り返し運動）を実施すると、個々の磁石要素の固定ボルトは短期間で金属疲労を起こし、最悪の場合、位相駆動中にボルトが折れることになる。ボルトが折れると、磁石要素の落下で真空チャンバーが大きなダメージを受けることは避けられない。そこで、磁石固定ボルトの寿命を伸ばすという目的で、放射光実験側が許す限り、長い保持時間(hold-time)を設定し、長い周期の位相駆動を実施するが望ましい。

ＥＭＡプロセスの本質は、関数 em_ema_loop_sequence.c (Appendix Ｉのソースコードを参照)に書き込まれてある SVOC コマンドを順番に実行し、関数の最後に到達したらまた最初へ戻り、このループをエンドレスに繰り返すことにある。これらの SVOC コマンドは、ＥＭＡフレーム内で再帰的に呼び出される。位相駆動中、アンジュレータ放射光は右偏光から左偏光へ、そして左から右へと周期的に切り替えられることから、em_ema_loop_sequence.c では、磁石列の位置を示す「位相トリガー信号」が SVOC コマンドで発令されている。位相トリガー信号は、Fig.13 に示す仕組みで、保守通路側（制御盤側）から光ケーブル経由で 120 m 先きのビームライン側へ渡され、放射光実験で利用される。

5. 動作確認

5-1 ギャップ駆動

ギャップ駆動を確認するために、位相差を 0.0mm に固定し、ギャップ 36.0mm（最小値）から300.0mm（最大値）へ動かした。駆動時間は、約７分である。駆動中の速度は終始一定であるが、駆動開始時には一定の速度に到達するまで加速され、停止時には加速時と同じレートで減速される。パルスモータの回転速度と加速レートは、デバイス構成表で設定できる。Fig.14 に示すように、１０台の補正電磁石の励磁周期は、24msec(42Hz)である。この周期は、ギャップと位相差を読むためのリニアスケールのラッチ・オンからオフまでの時間間隔で決まる。ギャップと位相差は、励磁テーブルから励磁電流を読み取るために用いられる。ギャップと位相差がＥＭに取り込まれた瞬間から１０台の補正電磁石を同時に励磁するまでの時間は 16msec である。

5-2 位相駆動

位相駆動を確認するために、ギャップを 300.0mm に固定し、位相差を-100.0mm から 100.0mm へ動かした。駆動の開始時には、磁石列は一定の速度に到達するまで加速される。停止時には、加速と同じレートで減速される。ＡＣサーボモータの回転速度と加速レートは、デバイス構成表で設定できる。Fig.15 と Fig.16 に示すように、位相駆動中、１０台の補正電磁石の励磁周期は、24msec(42Hz)である。この周期は、ギャップと位相差を読むためのリニアスケールのラッチ・オンからオフまでの時間間隔で決まる。ギャップと位相差は、励磁テーブルから励磁電流を読み取るために用いられる。ギャップと位相差がＥＭに取り込まれた瞬間から１０台の補正電磁石を同時に励磁するまでの時間は 16msec である。

一方では、周期的位相駆動中の磁石列折り返し点オーバーシュートを低減するために、ＡＣサーボモータのパラメータを変える試験も実施した。対象としたパラメータは、最高パルス発生レート（pulse/sec）、最低パルス発生レート（pulse/sec）とパルス発生レートの変化率（microsec）である。磁石列の動きを捕えるには、リニア・ポテンショメータ（抵抗５ＫΩ）を磁石列に取り付けて、ディジタル・マルチメータで抵抗値を測定した。結果は Fig.17 で示すように、オーバーシュートは、最適化する前の 0.8 mm（ケースＡ）から最適後は 0.1 mm（ケースＢ）以下となった。この値は仕様を満足する。

5-3 ギャップと位相の動的読み取り試験

本アンジュレータでは、既に述べたように、ギャップと位相の位置を読み取るためにリニアスケールとロータリエンコーダを使っている。

リニアスケールは信号をBCD(単位:mm)で出力し、この読み値を直接ＶＭＥデバイス(ttldio_0)へ送信している。一方、ロータリエンコーダは、信号をグレイ・バイナリ・コードで出力するために、先ず表示器系ＰＬＣでグレイ・バイナリ・コードをBCD（単位：mm)に変換して、BCDをＶ

ＭＥデバイス(di_0)へ送信している。リニアスケールの読み取り速度は、ラッチ信号の繰り返し周期（カタログ値は、25 microsec）で決まる。一方のロータリエンコーダの読み取り速度は、ＰＬＣのスキャン・タイム（カタログ値は、20 msec）で決まり、応答性はリニアスケールの約１／１０００である。こうした理由から、ＥＭとＥＭＡでは、ギャップと位相の位置を高速でサンプリングしなければならないために、リニアスケールの読み値を使っている。ロータリエンコーダの読み値は、静止状態におけるリニアスケール読み値との比較に使っている。

しかし、ギャップと位相の駆動中におけるリニアスケール読み値の検証が必要である。リニアスケールの高速サンプリングで得られる読み値を確かなデータにしておかないと、位相駆動と励磁電流の相関を求めることができないためである。そこで、新しいＰＬＣ（横河電機製のFAM3、スキャン・タイム 1msec）を追加導入して、ロータリエンコーダの出力信号（グレイ・コード）をギャップと位相の実際位置に変換するという測定方法を作った。この方法では、一度に 4960 点のギャップと位相の読み値をパソコンへ取り込むことができる。さらに一方では、補正電磁石の励磁電流を周期 100 Hz で測定するには、時間軸のジッターを事前に調べておく必要がある。ディジタル・マルチメータ Keithley-2001 で電圧を測定した結果、Fig. 18 で示すように、連続データ数 1000 点の範囲において、ジッターは± １％以内であることも検証した。

位相駆動中、ロータリエンコーダの出力信号（グレイ・コード）を使って磁石列の位相の位置を動的に読み取る試験を実施した。 Fig. 19 は、位相差 = 32.0 mm （上側位相 = 16.0 mm、下側位相 = -16.0 mm）と繰り返し周期 = 3.7 sec の条件で実施した位相駆動をスキャン・タイム 1 msec のＰＬＣで読み取った試験結果である。スキャン・タイム 1msec の高速ＰＬＣを使っているとはいえ、実際のギャップと位相駆動では、パルスモータとＡＣサーボモータともに回転速度が高いために、ギャップと位相の位置を 100 Hz でサンプリングすることは困難である。そこで現在、次の試験準備として、ロータリエンコーダの出力信号に直接 100 Hz の周期でラッチをかけられるよう、「ラッチ回路」の導入を検討している。ラッチ回路を使ったギャップと位相の動的読み取り試験については、別途報告する予定である。

6. 結　論

制御系ソフトウェアがSPring-8のSVOCコマンド制御方式に合致したものとなったのは、平成１０年２月上旬であった。同月の下旬には、ビームラインＢＬ２３ＳＵのコミッショニングで、ギャップを60.0 mmに固定し、位相を0.0 mm（水平直線偏光），32.0mm（円偏光）と変えて、フロントエンド側で放射光の初観測に成功した。制御ソフトウェアＥＭを開発する過程では、加速器制御グループとビームライン制御グループの指導を受けながら、ＥＭとPoller-Collectorとの間で発生するデバイスの取り合いの問題、ＲＩＯの配線の問題、SVOCコマンドの再帰呼び出しといった難問を乗り越えてきた。その後、制御グループの技術的要請に応えながら、パラメータを変えてＥＭを最適化してきた。位相駆動と補正電磁石の励磁を同期させて相関をとるための計測系の開発、位相駆動中に発生する磁石列のオーバーシュートの低減、メーカー納入の完成図書（制御系に関係する部分）のレビュー、ＰＬＣ（No.１とNo.2）シーケンス制御系のレビュー、各センサーの追従性の検討といった課題を通して、本制御系から取り出すことのできる性能の限界を見極めた。のち、１つのＥＭ関数でデバイスを２個以上呼ぶための改造、またＥＭシミュレータを使ってソースコードを効率よくデバッグするための改造、ＳＲＡＭ(1 MByte) と ＤＲＡＭ(128 MByte) によるログ機能の導入等を実施した。そして平成１１年５月には、拡張ソフトウェアＥＭＡを開発し、アンジュレータの最大の特徴である周期的位相駆動の実現に成功した。

平成１１年９月現在、本制御系は以下の特徴を持つ状態に至っている。

(a) ギャップ36.0 mmから300.0 mmまでの駆動時間は、約７分。
(b) 位相駆動時の磁石列の折り返し点オーバーシュトは、0.1 mm以下。
(c) ギャップ駆動と位相駆動中の補正電磁石の励磁周期は、24 msec(42 Hz)。
(d) 周期的位相駆動の周期のバラツキ誤差は、0.1% 以下。
(e) すべての補正電磁石（１０台）は、テーブル励磁できる。
(f) 停電の復旧時、ＥＭプロセスはホット・スタートで立ち上がる。
(g) 補正電磁石の励磁電流は、位相駆動中にタイムスタンプ付きで測定できる。ディジタル・マルチメータKeithley-2001を用いた場合の測定点数は、最大１４２６点。
(h) ギャップと上位相位置、下位相位置は、ロータリエンコーダのグレイ・コード出力をＢＣＤへ変換して、動的に読み取ることができる。横河電機製ＰＬＣ（FAM3）を用いた場合の測定点数は、最大４９６０点。
(i) 原点出し（ギャップ軸、上側位相軸、下側位相軸）は、ローカルＥＭ（tellem*）を使って自動的に実施できる。

謝 辞

本制御系ソフトウェアの開発にあたって多大な協力をいただいた SPring-8 加速器制御グループとビームライン制御グループの皆様に謝意を表します。

参考文献

1) S. Sasaki, K. Miyata, and T. Takada: Jpn. J. Appl. Phys., Vol.31, L1794(1992)
2) 角野和義、佐々木茂美、島田太平、宮原義一：JAERI-M 93-156, “新型可変偏光アンジュレータの解析” (1993)
3) H. Kobayashi, S. Sasaki, T. Shimada, M. Takao, A. Yokoya and Y. Miyahara, “Design of Variable Polarizing Undulator (APPLE Type) for SX Beamline in the SPring-8”, Proceedings of the Fifth European Particle Accelerator Conference (EPAC-96, Barcelona), Vol. 3, June (1996).
4) 小林秀樹、佐々木茂美、島田太平、高雄勝、横谷明徳、宮原義一: JAERI-Tech 96-013 “SPring-8 原研軟X線ビームライン用挿入光源の磁場解析” (1996)
5) Y. Teraoka, SPring-8 Annual Report 1997 (1997).
6) A. Yokoya, T. Sekiguchi, Y. Saitoh, T. Okane, T. Nakatani, T. Shimada, H. Kobayashi, M. Takao, Y. Teraoka, Y. Hayashi, S. Sasaki, Y. Miyahara, T. Harami and T. A. Sasaki, “Soft X-ray Beamline Specialized for Actinides and Radioactive Materials Equipped with a Variably Polarizing Undulator”, Journal of Synchrotron Radiation (1998) Vol. 5, 10-16.
7) 「蓄積リング制御ソフトウェア設計方針 (V2.0) 」、1995 年 10 月、SPring-8 蓄積リング制御グループ発行。
8) 「蓄積リング制御ソフトウェア書式規定 (C言語版、V1.8) 」、1996 年 2 月、SPring-8 蓄積リング制御グループ発行。
9) 「Equipment Manager 開発マニュアル (第 3.2 版) 」、1997 年 8 月、 SPring-8 蓄積リング制御グループ発行。
10) Y. Hiramatsu, T. Shimada, T. Bizen, and Y. Miyahara, “Design Status of a Double Array Undulator Control System”, SPring-8 Annual Report 1996 (1996).
11) Y. Hiramatsu, T. Shimada, and Y. Miyahara, “Control System for an APPLE Type Undulator”, SPring-8 Annual Report 1998 (1998).
12) Y. Hiramatsu, T. Shimada, and Y. Miyahara, “Dynamic Measurement of Magnet Array Phase Position in an APPLE Type Undulator”, SPring-8 Annual Report 1998 (1998).
13) Y. Hiramatsu, T. Shimada, and Y. Miyahara, “Development of EM Software for the Dynamic Excitation of Steering Magnets in an APPLE Type Undulator”, 第 12 回加速器科学研究発表会 (理化学研究所、1999 年 10 月) 。
14) Y. Hiramatsu, T. Shimada, and Y. Miyahara, “Development of EMA Software for the Periodic Movement of Magnet Arrays in an APPLE Type Undulator”, 第 12 回加速器科学研究発表会 (理化学研究所、1999 年 10 月) 。

Table 1 VME I/O Boards and Installation Data

Slot	Destination	Model	AM Code	Base addr	Memory area	INT level	INT vector	Device name
1&2	CPU	HP9000-743rt	------	------	------	------	------	------
3	ADC	PVME-311	A32/supervisor	0x80000000	base+0x1FFFFF	IRQ5	24-28	adc311_0
(3)	------	------	(A24/supervisor)	(0x800000)	(base+0x7FFFF)	------	------	------
4	Gap movement	MP-0350	A16/supervisor	0x1000	base+0x3F	IRQ4	8-15	ptg0350_0
5	Phase movement	MP-0350	A16/supervisor	0x2000	base+0x3F	IRQ6	16-23	ptg0350_2
6	Rotary-encoder	AVME-9421	A16/supervisor	0x3000	base+0x3FF	(IRQ2)	(1)	di_0
7	Linear-scale	HIMV-610/TI-96	A16/supervisor	0x4000	base+0xF	(IRQ3)	(1)	ttldi_0
8	Sensors	AVME-9421	A16/supervisor	0x5000	base+0x3FF	(IRQ2)	(2)	di_1
9	Preset signal	AVME-9431	A16/supervisor	0x6000	base+0x3FF	None	None	do_0
10	Latch signal	HIMV-630/TIO-96	A16/supervisor	0x7000	base+0xF	None	None	ttldio_0
11	RIO master	MELTAC-C(DFBJ)	A24/supervisor	0x100000	base+0xFFF	None	None	rio_0
(11)	------	------	A16/supervisor	0x8000	base+0xFF	------	------	------
12	RIO branch1	MELTAC-C(DSPJ)	------	------	------	------	------	------
13	RIO branch2	MELTAC-C(DSPJ)	------	------	------	------	------	------
14	Ion pump controller	AVME-9421	A16/supervisor	0x9000	base+0x3FF	(IRQ2)	(3)	di_2
15	Interlock	HIMV-602A	A16/data access	0xa000	base+0x1000	None	None	dio602a_0
16	GAPA	HIMV-602A	A16/data access	0xa010	base+0x1000	None	None	dio602a_1
17	DRAM(128MB)	AVME-242	A32/data access	0x60000000	base+0x0800000	None	None	DRAM_0
18	Reserved slot	------	------	------	------	------	------	------
19	Reserved slot	------	------	------	------	------	------	------
20	SRAM(4MB)	HIMV-220	A32/data access	0x00200000	base+0x3FFFFF	None	None	sram_0(EM)
(20)	------	------	------	------	------	------	------	sram_1(polcol)
(20)	------	------	------	------	------	------	------	sram_2(reserve)
(20)	------	------	------	------	------	------	------	sram_3(reserve)

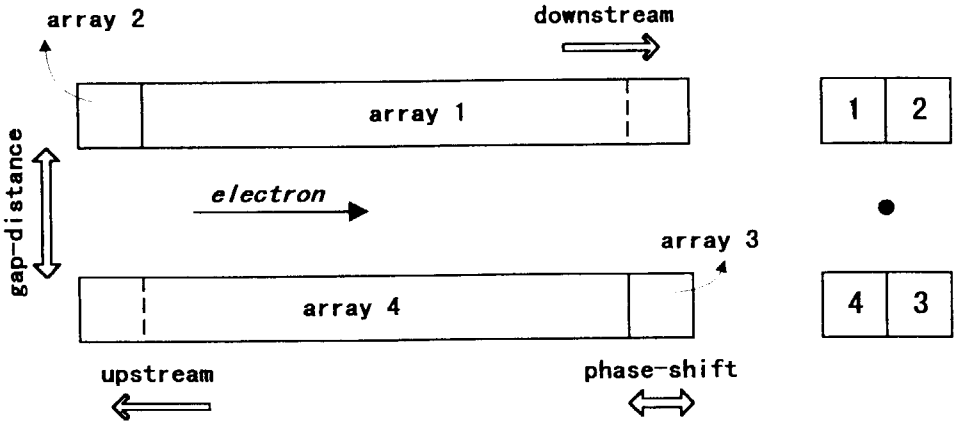


Fig. 1 Gap-distance and Phase-shift of the Magnet Arrays

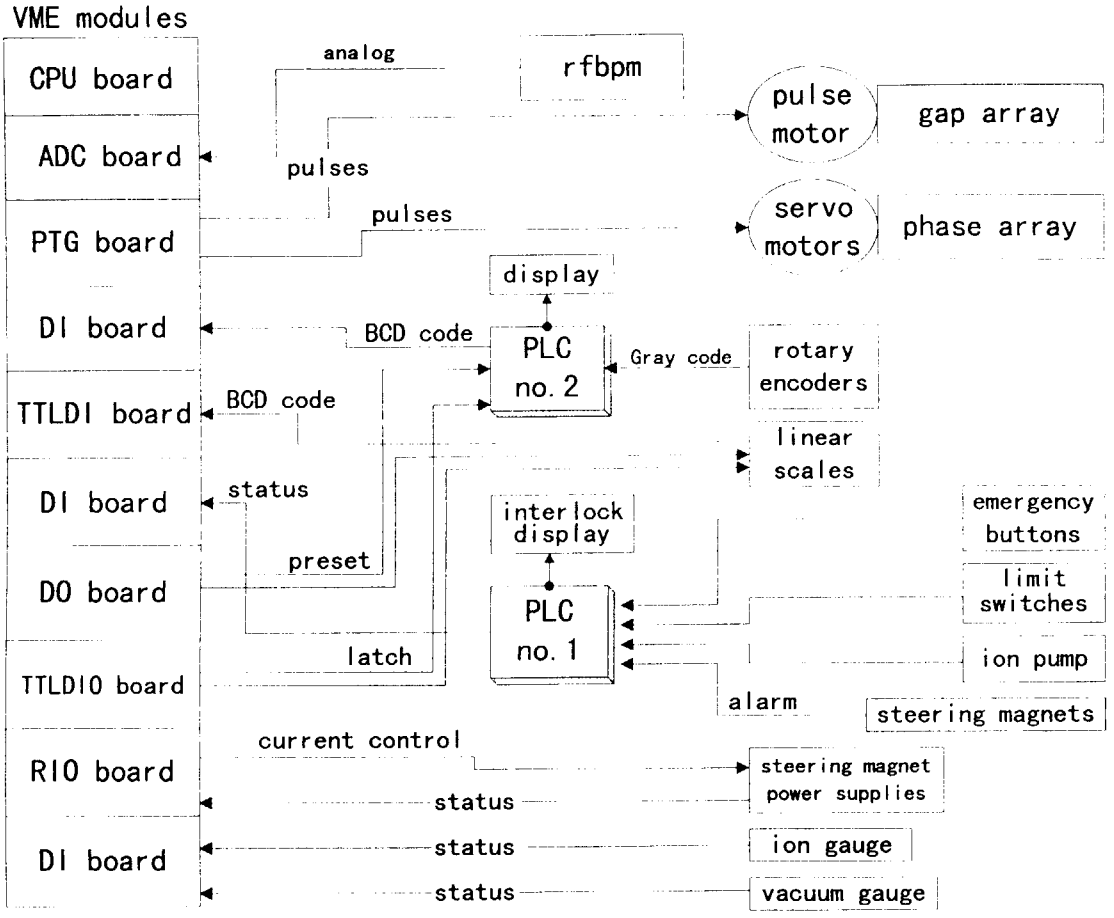


Fig. 2 Schematic of Control System

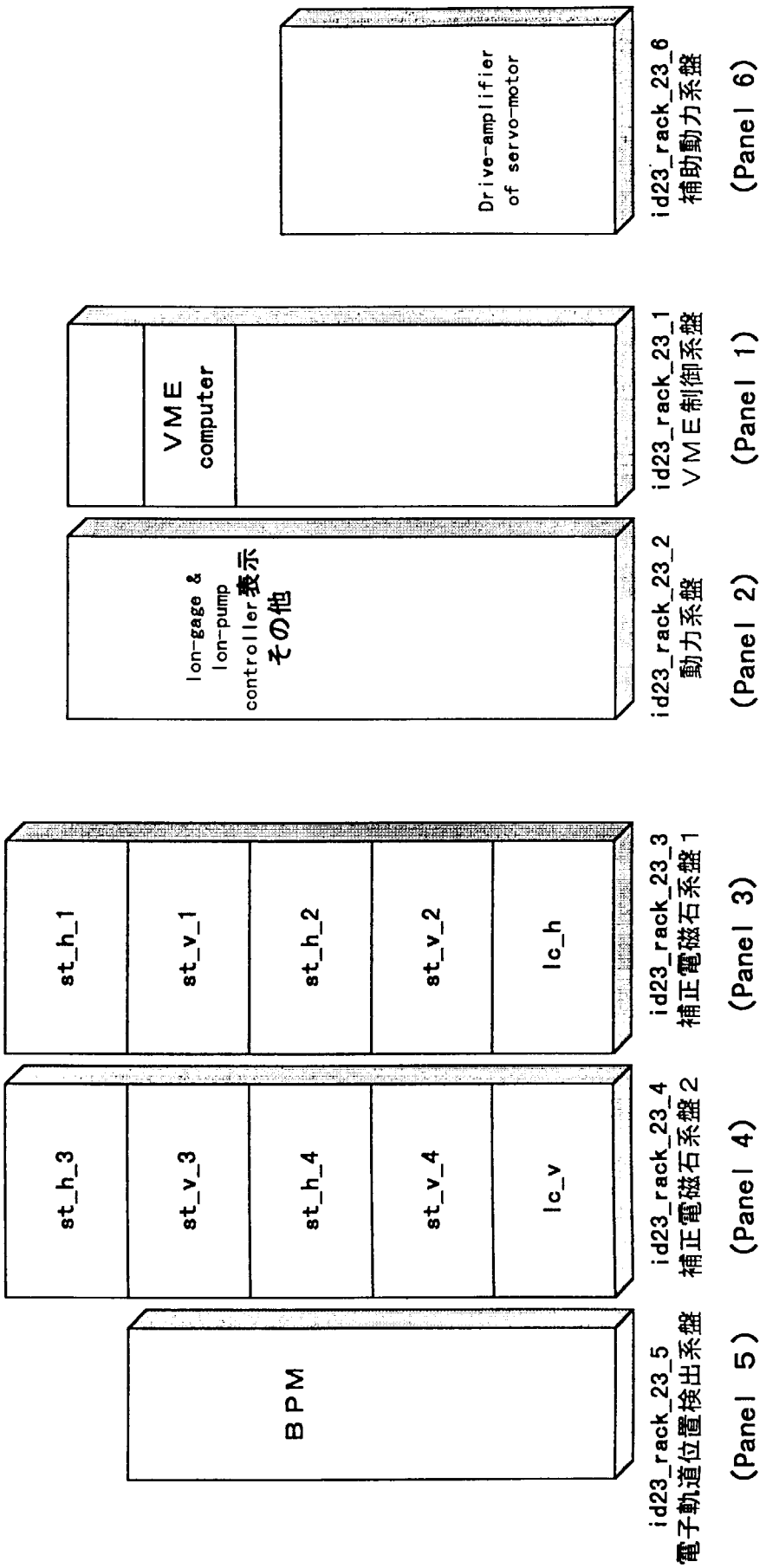


Fig. 3 Lay-out of Control Panels

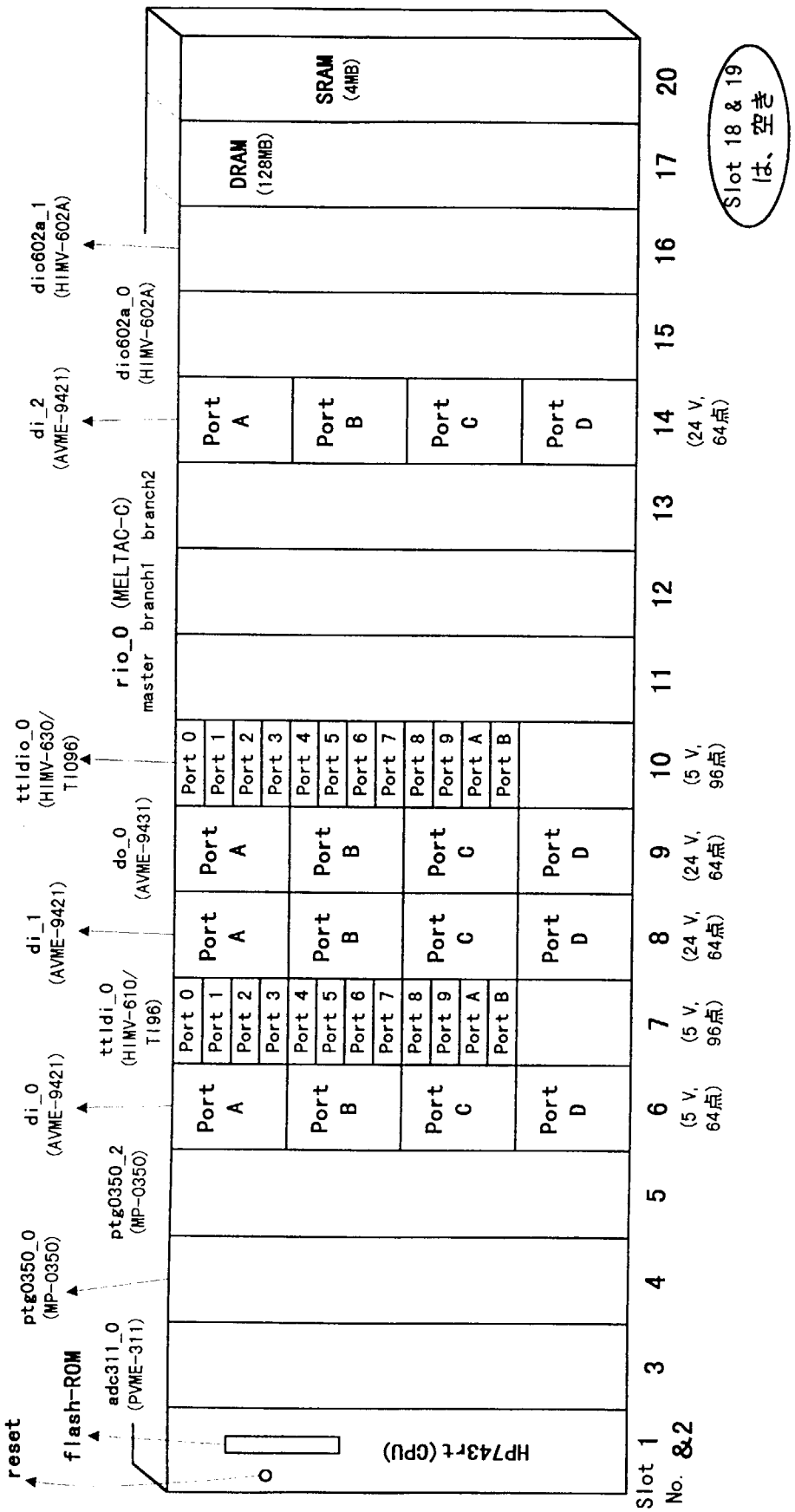


Fig. 4 Lay-out of VME I/O Boards

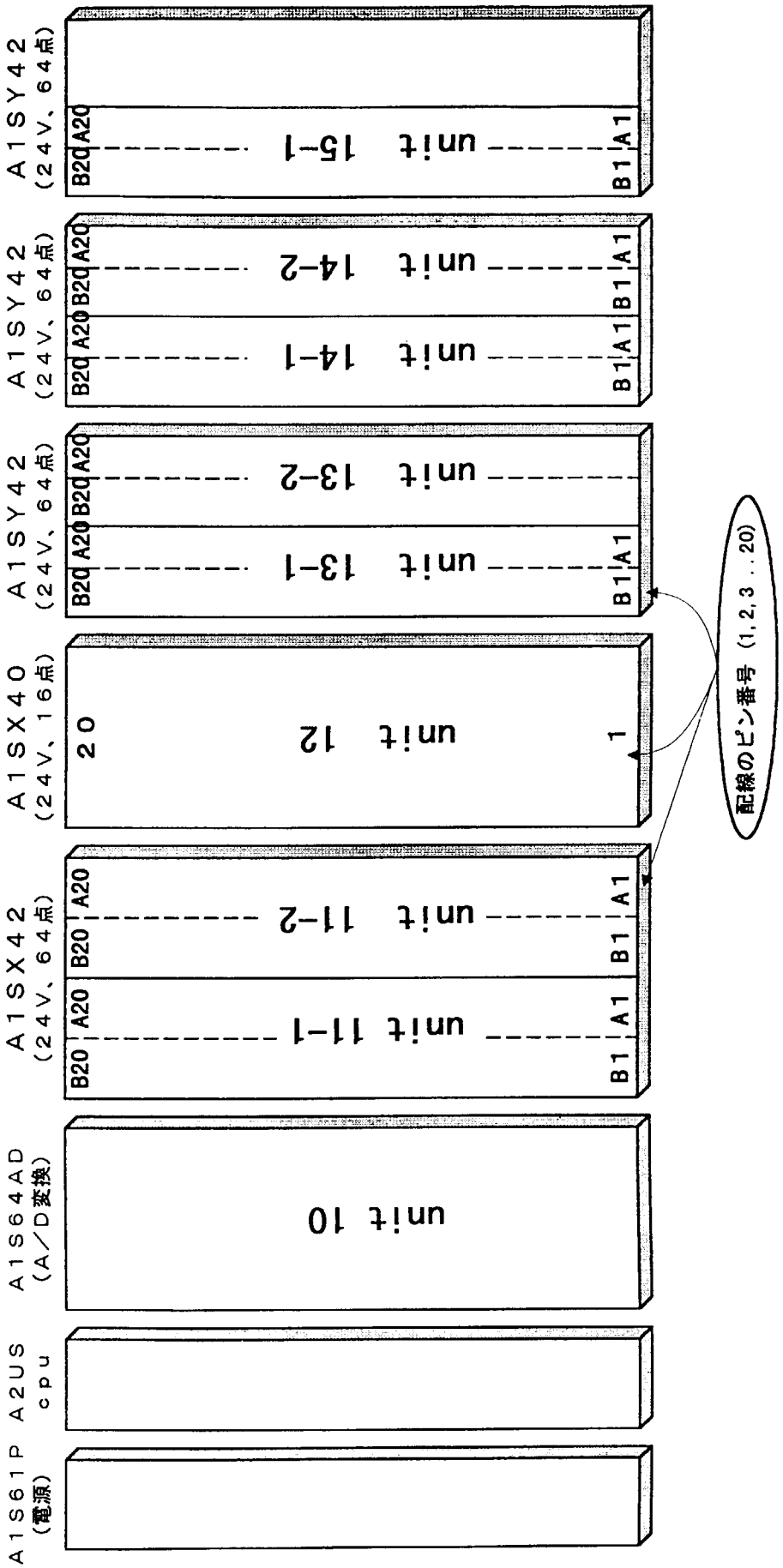


Fig. 5 Lay-out of I/O Units for PLC No.1

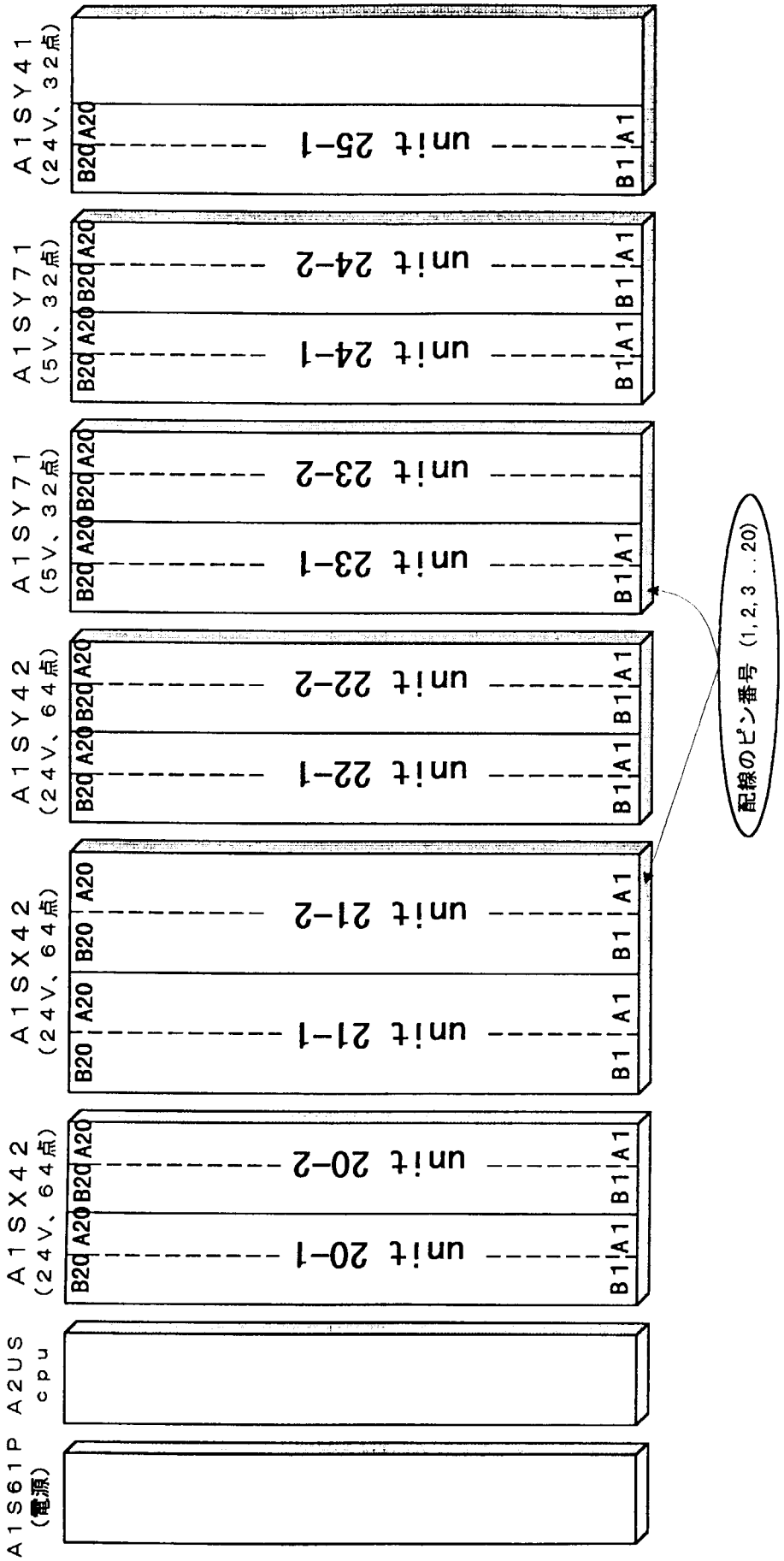


Fig. 6 Lay-out of I/O Units for PLC No.2

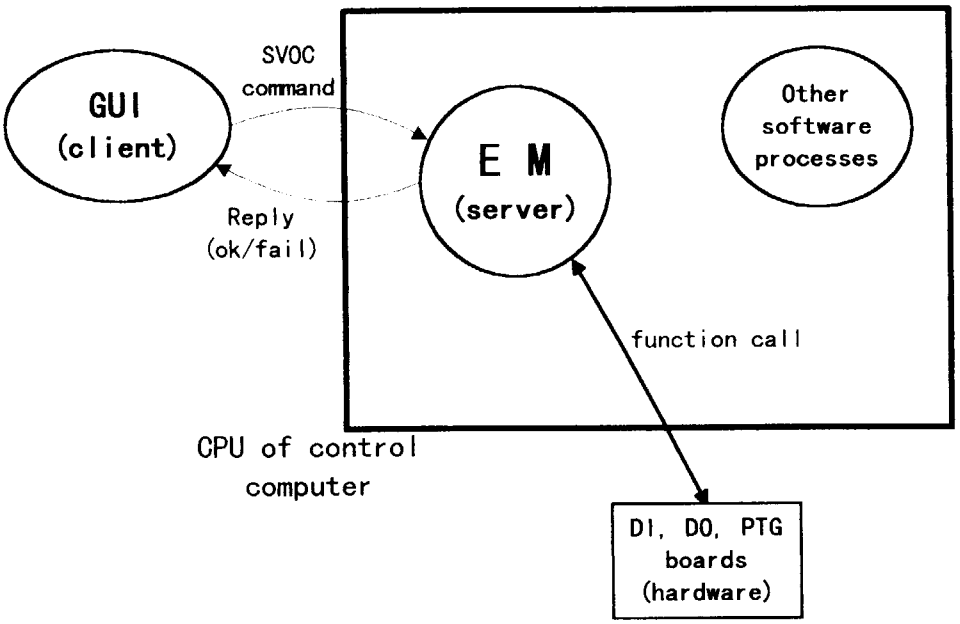


Fig. 7 Process EM on the Control Computer

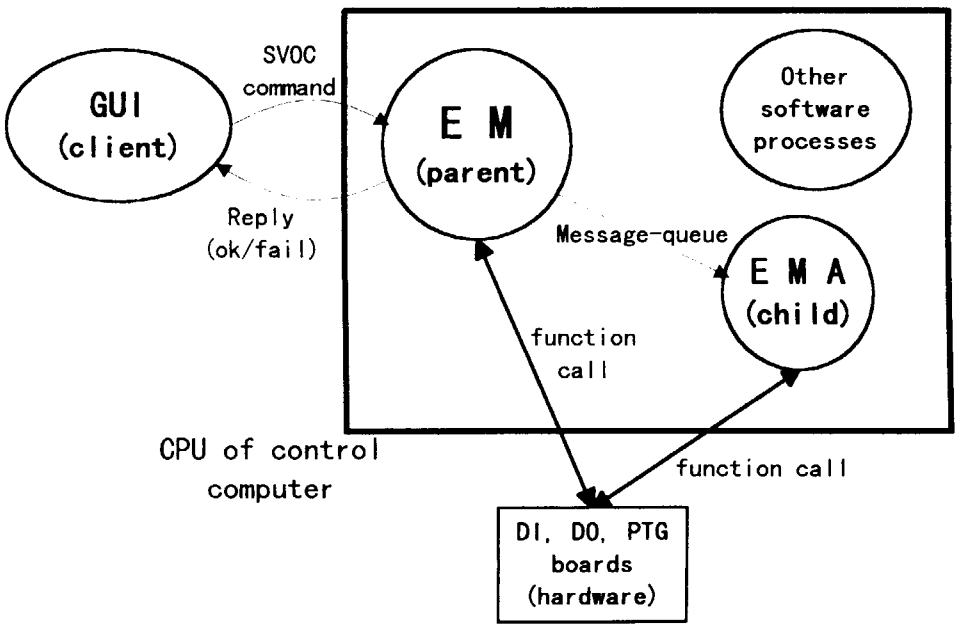


Fig. 8 Coexistence of EM and EMA on the same CPU

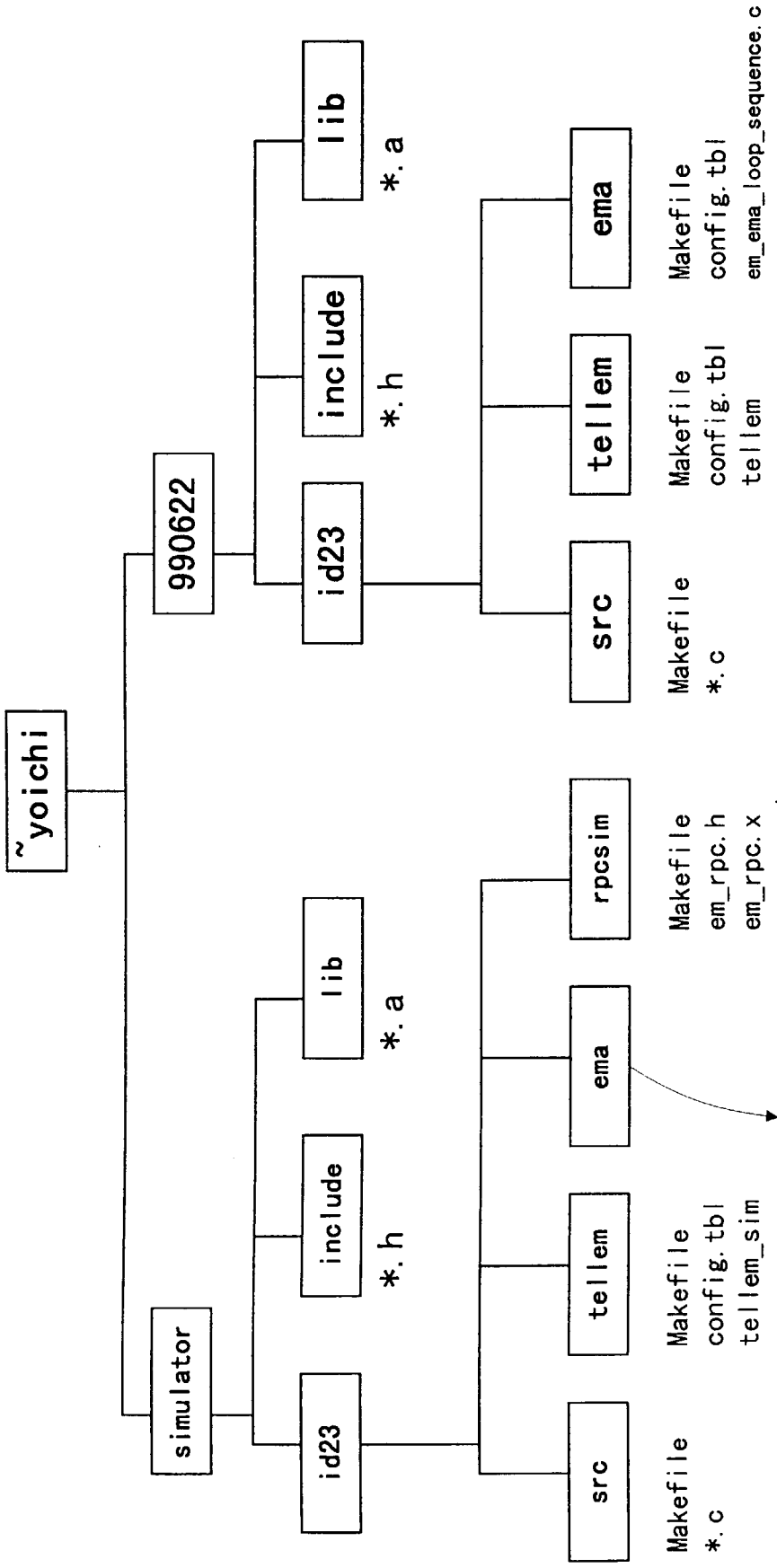


Fig. 9 Programming Environment

tellemの実行環境：

/prj/bin/id23rt/test/bl_id23_steer0.tbl　(補正電磁石の励磁テーブル)
/prj/bin/id23rt/test/bl_id23_dummy.tbl　(ダミーの励磁テーブル)
/prj/bin/id23rt/test/config.tbl　(デバイス構成表)
/prj/bin/id23rt/test/ema_rt*　(EMA)
/prj/bin/id23rt/test/tellem*　(EM)

rpc_emの実行環境：

/home/bl/blcntl/bl_id23_steer0.tbl　(補正電磁石の励磁テーブル)
/prj/bin/id23rt/em/config.tbl　(デバイス構成表)
/prj/bin/id23rt/em/emal_rt*　(EMA)
/prj/bin/id23rt/em/rpc_em*　(EM)

poller-collectorの実行環境：

/prj/bin/id23rt/poller_fast/config.tbl　(デバイス構成表)
/prj/bin/id23rt/poller_fast/pc_po_main*　(ポーラ・コレクタ)

ＳＲＡＭログのダンプ方法：

/prj/bin/em_sram_init_0*　(初期化、1 MByte)
/prj/bin/em_sram_dump_0* > /home/bl/blcntl/study/ファイル名　(ダンプ)
(ftpでkanbaiへ転送してから、データ解析すること)

ＤＲＡＭログのダンプ方法：

/prj/bin/em_dram_init_0*　(初期化、1 MByte)
/prj/bin/em_dram_dump_0* > /home/bl/blcntl/study/ファイル名　(ダンプ)
(ftpでkanbaiへ転送してから、データ解析すること)

Fig.10 Execution Environment at "id23rt" Control Computer

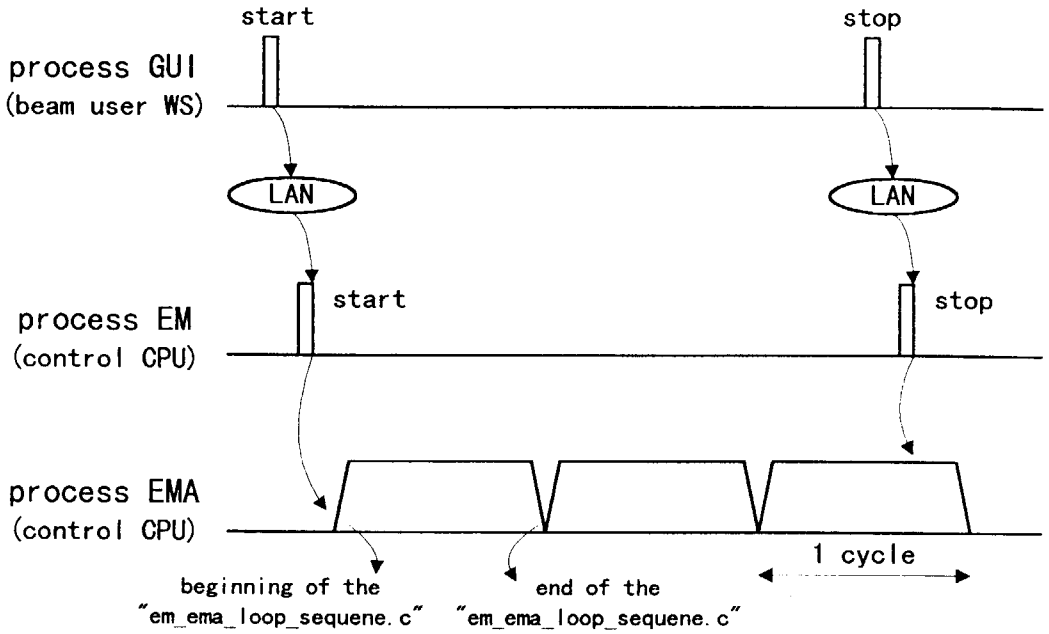


Fig. 11 Timing-chart of "start" and "stop" of EMA

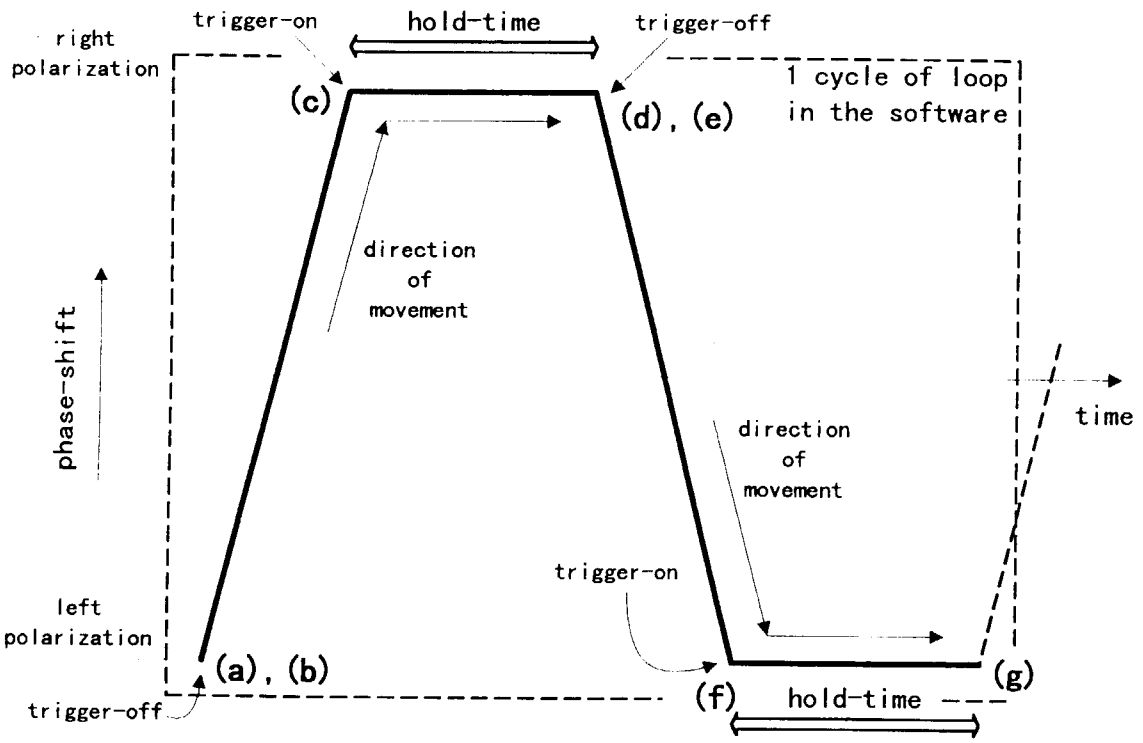


Fig. 12 Schematic of a Cycle of Periodic Movement

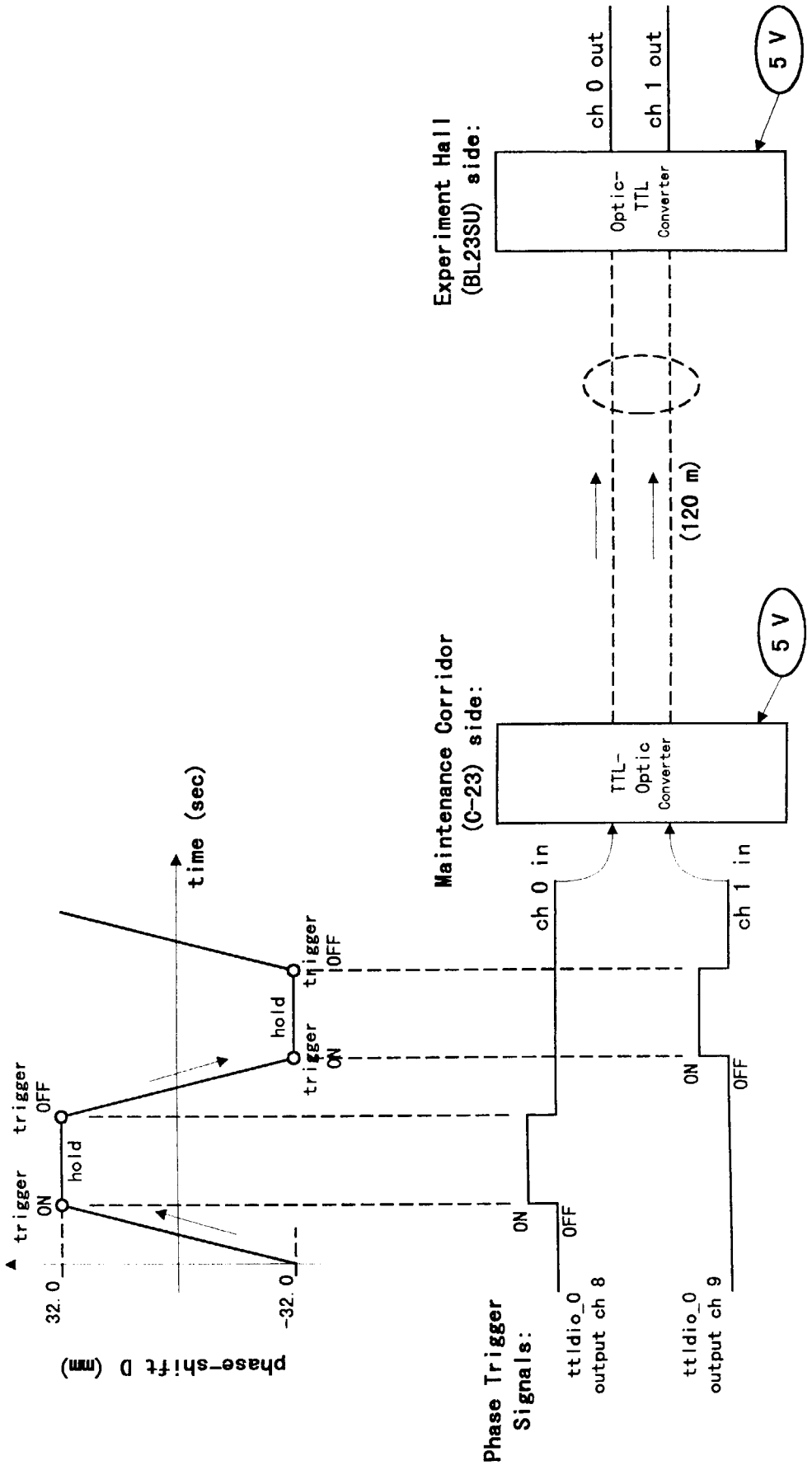


Fig. 13 Phase Trigger Signals during Periodic Phase Movement

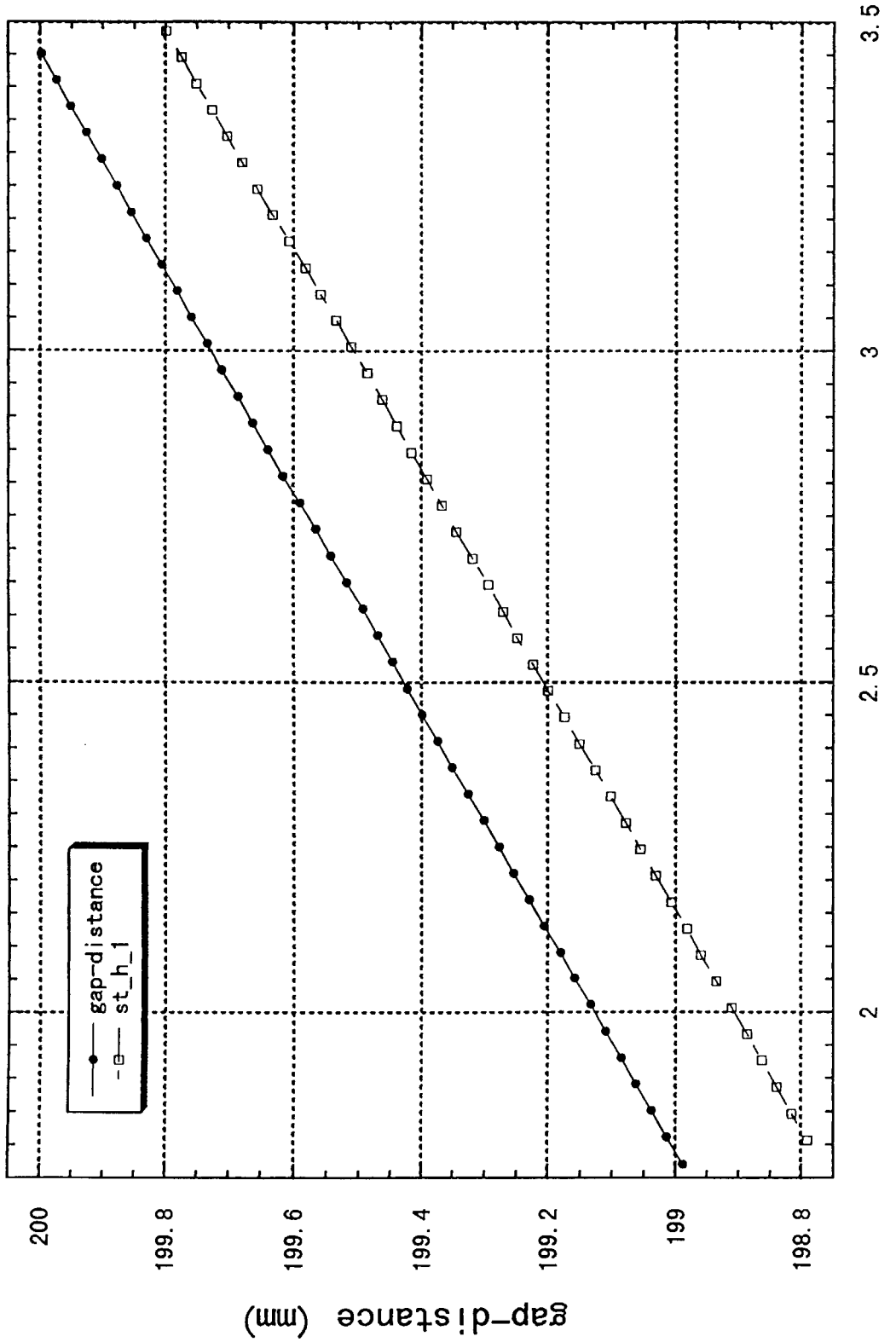


Fig. 14 Gap Movement (the frequency of excitation of steering Magnets is 42 Hz)

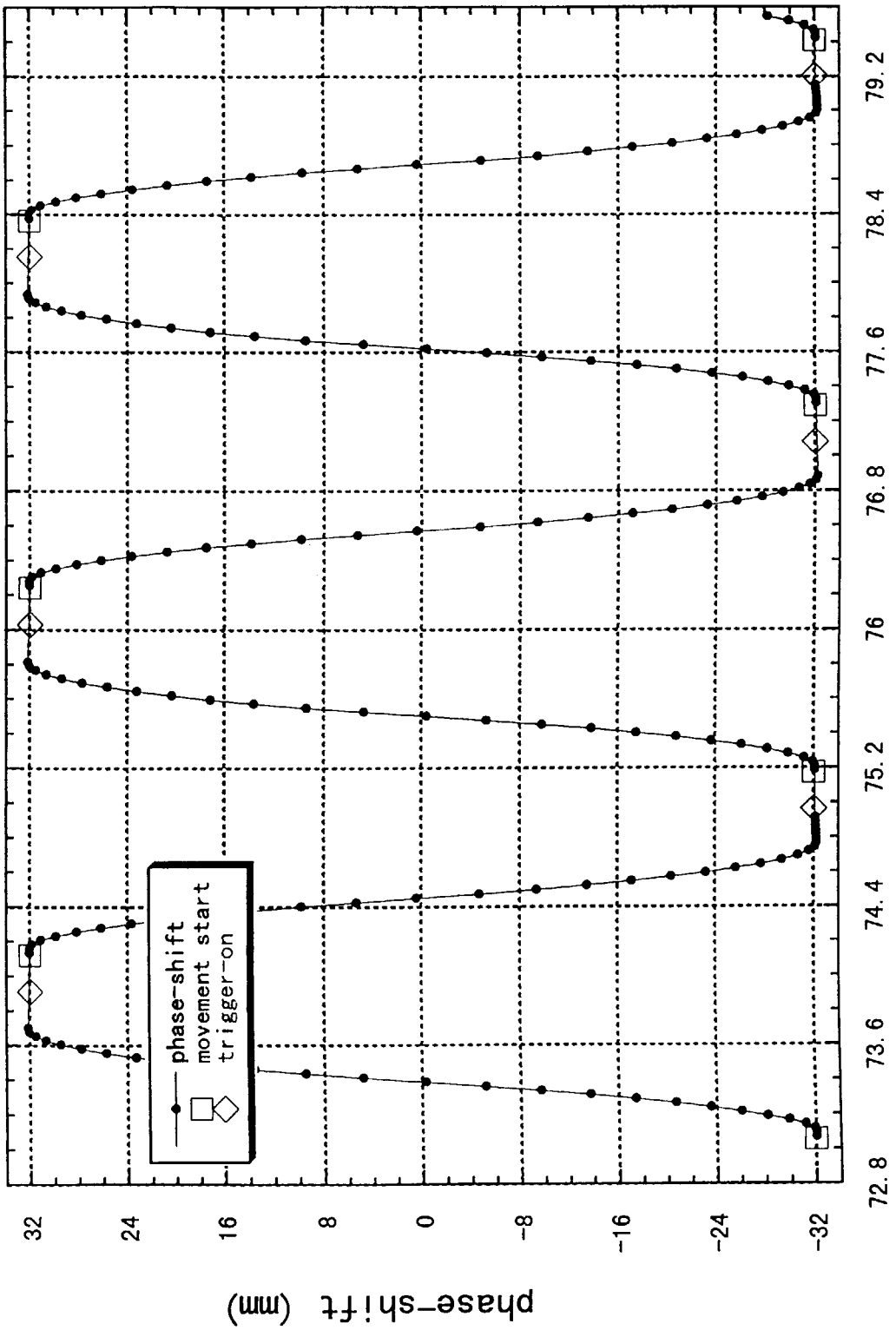


Fig. 15 Periodic Phase Movement (the frequency of excitation of steering Magnets is 42 Hz)

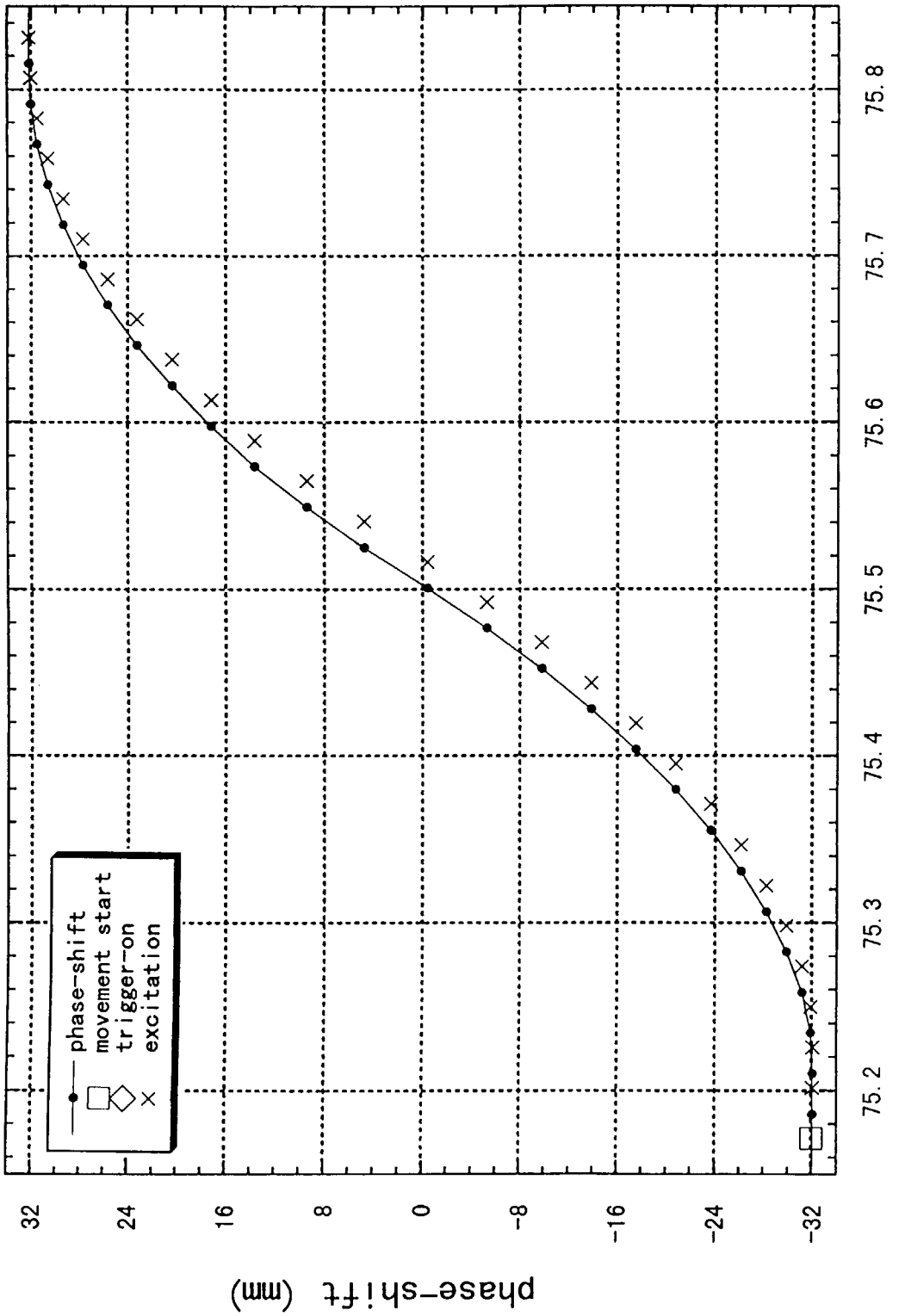


Fig. 16 Excitation of Steering Magnets at a period of 24 msec (42 Hz)

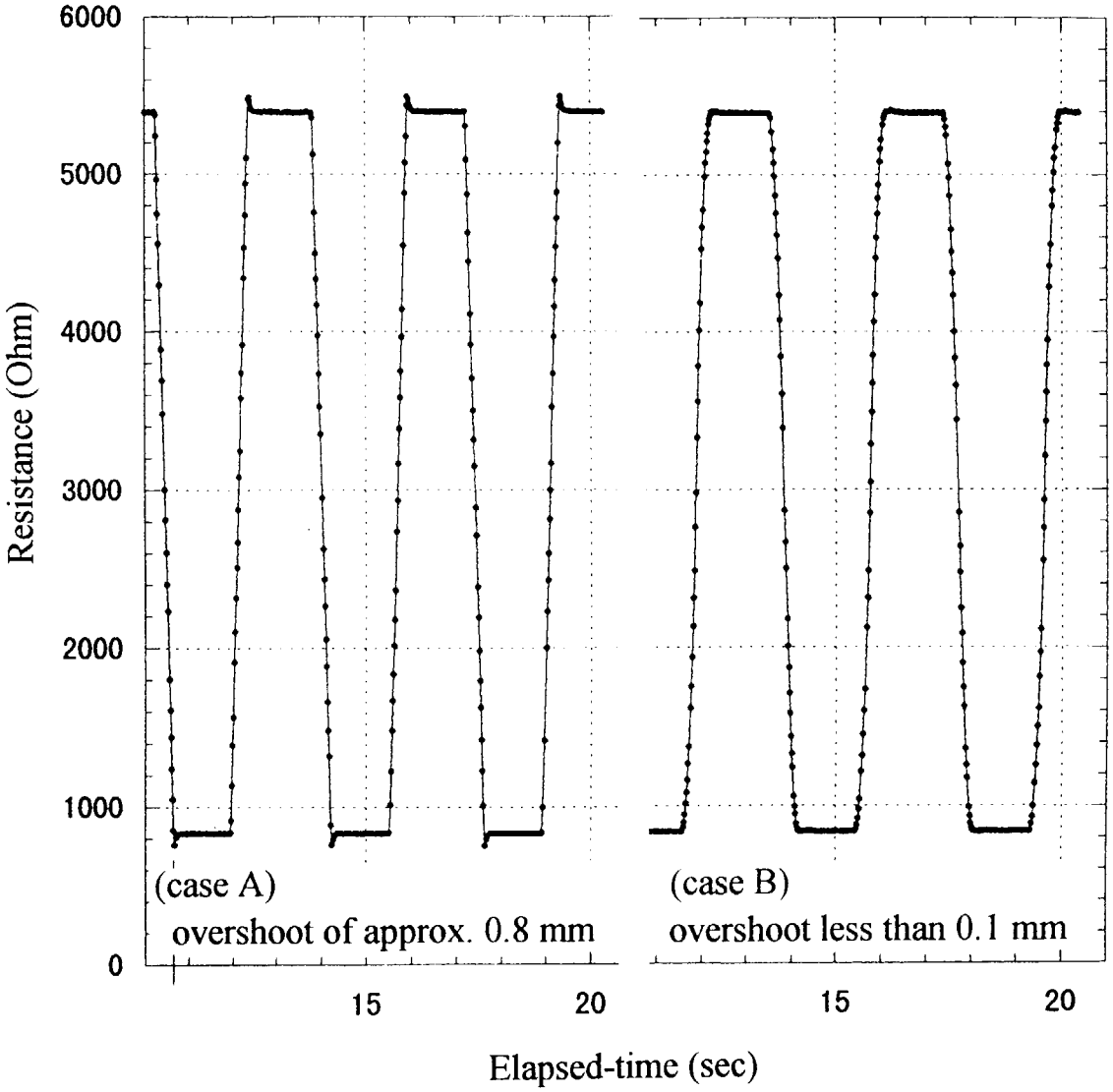


Fig. 17 Resistance Variation as a Function of Time

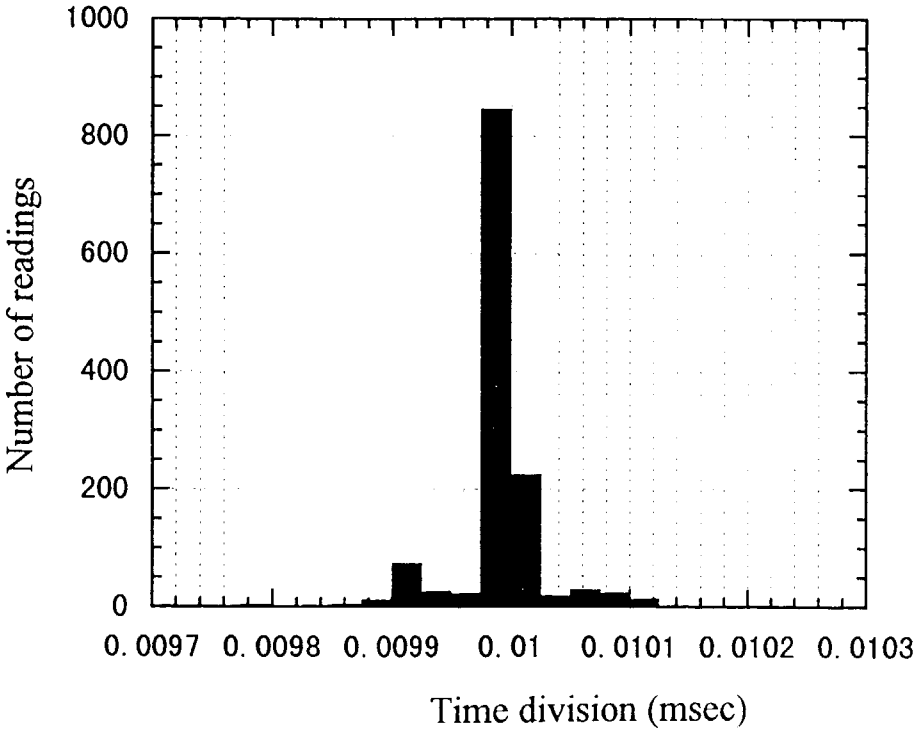


Fig. 18 Deviation of Sampling Time observed in the Measurement of Excitation Currents

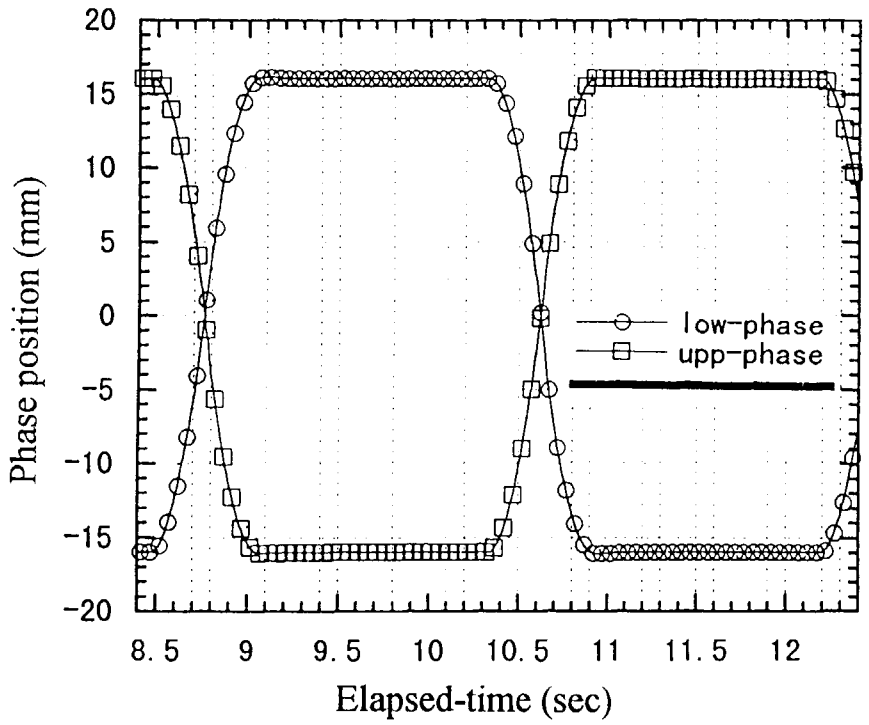


Fig. 19 Phase Position as a Function of Time

List of Appendices

Appendix A: SVOC Commands and Related Data for Database System

O	V	C	Signal Type	Fun1	Fun2	Fun3
bl_id23_gap	put	%fmm		em_id23_gap_move_put_abs	none	em_std_ret
bl_id23_gap_relat	put	%fmm		em_id23_gap_move_put	none	em_std_ret
bl_id23_gap	put	initial		em_id23_gap_init_put	none	em_std_ret
bl_id23_gap	put	preset		em_id23_gap_preset_put	none	em_std_ret
bl_id23_gap_rot	get	position	mm	em_id23_position_gap_rot_get	none	em_id23_position_conv_get
bl_id23_gap	get	position	mm	em_id23_position_gap_lin_get	none	em_id23_position_conv_get
bl_id23_gap	get	status	bl_id23_gap_status	em_id23_gap_status_get	none	em_id23_status_conv_get
bl_id23_hold	put	%dus		em_id23_hold	none	em_std_ret
bl_id23_phase	put	%fmm		em_id23_phase_move_put	em_id23_phase_conv_put	em_std_ret
bl_id23_phase_upper	put	%fmm		em_id23_phase_move_one_side_put	em_id23_phase_conv_put	em_std_ret
bl_id23_phase_lower	put	%fmm		em_id23_phase_move_one_side_put	em_id23_phase_conv_put	em_std_ret
bl_id23_phase_upper	put	initial		em_id23_phase_init_put	none	em_std_ret
bl_id23_phase_lower	put	initial		em_id23_phase_init_put	none	em_std_ret
bl_id23_phase_upper	put	preset		em_id23_phase_preset_put	none	em_std_ret
bl_id23_phase_lower	put	preset		em_id23_phase_preset_put	none	em_std_ret
bl_id23_phase	put	reset		em_id23_phase_reset	none	em_std_ret
bl_id23_phase_upper_rot	get	position	mm	em_id23_position_phase_u_rot_get	none	em_id23_position_conv_get
bl_id23_phase_upper	get	position	mm	em_id23_position_phase_u_lin_get	none	em_id23_position_conv_get
bl_id23_phase_lower_rot	get	position	mm	em_id23_position_phase_l_rot_get	none	em_id23_position_conv_get
bl_id23_phase_lower	get	position	mm	em_id23_position_phase_l_lin_get	none	em_id23_position_conv_get
bl_id23_phase	get	position	mm	em_id23_phase_position_get	none	em_idcom_conv_get_pol
bl_id23_phase	get	status	bl_id23_phase_status	em_id23_phase_status_get	none	em_id23_status_conv_get
bl_id23_phase_brake	put	on		em_id23_phase_brake_on_put	none	em_std_ret
bl_id23_phase_brake	put	off		em_id23_phase_brake_off_put	none	em_std_ret
bl_id23_phase_trigger_[r,l]	put	on		em_id23_phase_trigger_on_put	none	em_std_ret
bl_id23_phase_trigger_[r,l]	put	off		em_id23_phase_trigger_off_put	none	em_std_ret
bl_id23_phase_trigger_[r,l]	get	status	em_id23_phase_trigger_status	em_id23_phase_trigger_status_get	none	em_id23_status_conv_get
bl_id23_pattern	put	create		em_id23_pattern_create	none	em_std_ret
bl_id23_pattern	put	destroy		em_id23_pattern_destroy	none	em_std_ret
bl_id23_pattern	put	start		em_id23_pattern_start	none	em_std_ret
bl_id23_pattern	put	stop		em_id23_pattern_stop	none	em_std_ret
bl_id23_pattern_phase	put	%fmm_%fmm		em_id23_pattern_phase_put	none	em_std_ret
bl_id23_pattern_hold	put	%dmsec_%dmsec		em_id23_pattern_hold_put	none	em_std_ret
bl_id23_pattern	get	status	em_id23_pattern_status	em_id23_pattern_status_get	none	em_id23_pattern_status_conv_get
bl_id23_pattern_phase	get	position		em_id23_pattern_phase_get	none	em_id23_pattern_phase_conv_get
bl_id23_pattern_hold	get	time		em_id23_pattern_hold_get	none	em_id23_pattern_hold_conv_get
bl_id23_lc_table	put	%s		em_id23_lc_table_thru	em_id23_lc_table_set	em_std_ret
bl_id23_lc_[h,v]	set	%fA		em_id23_st_curr_set	none	em_std_ret
bl_id23_lc_[h,v]	put	on		em_id23_st_power_on_put	none	em_std_ret
bl_id23_lc_[h,v]	put	off		em_id23_st_power_off_put	none	em_std_ret

bl_id23_lc_[h,v]	put	reset		em_id23_st_err_reset	none	em_std_ret
bl_id23_lc_[h,v]	get	status	bl_id23_lc_status	em_id23_st_status_get	none	em_id23_status_conv_get
bl_id23_lc_[h,v]	get	current	A	em_id23_st_adc_curr_get	none	em_id23_st_curr_conv_get
bl_id23_lc_dac_[h,v]	get	current	A	em_id23_st_dac_curr_get	none	em_id23_st_curr_conv_get
bl_id23_st_[h,v]_[1-4]	set	%fA		em_id23_st_curr_set	none	em_std_ret
bl_id23_st	put	exec		em_id23_st_curr_put	em_id23_st_curr_exec	em_std_ret
bl_id23_st_[h,v]_[1-4]	put	on		em_id23_st_power_on_put	none	em_std_ret
bl_id23_st_[h,v]_[1-4]	put	off		em_id23_st_power_off_put	none	em_std_ret
bl_id23_st_[h,v]_[1-4]	put	reset		em_id23_st_err_reset	none	em_std_ret
bl_id23_st_[h,v]_[1-4]	get	status	bl_id23_st_status	em_id23_st_status_get	none	em_id23_status_conv_get
bl_id23_st_[h,v]_[1-4]	get	current	A	em_id23_st_adc_curr_get	none	em_id23_st_curr_conv_get
bl_id23_st_dac_[h,v]_[1-4]	get	current	A	em_id23_st_dac_curr_get	none	em_id23_st_curr_conv_get
bl_id23_rfbpm_[1-4]_[x,y]	get	voltage	V	em_id23_rfbpm_position_get	none	em_id23_rfbpm_conv_get
bl_id23_rfbpm_[1-4]_[x,y]	get	position	mm	em_id23_rfbpm_position_get	none	em_id23_rfbpm_convmm_get
bl_id23_rfbpm	get	status	bl_id23_rfbpm_status	em_id23_rfbpm_status_get	none	em_id23_status_conv_get
bl_id23_ivg_[1-2]	get	pressure	Pa	em_id23_vac_get	none	em_id23_vac_conv_get
bl_id23_ivg_[1-2]	get	status	bl_id23_ivg_status	em_id23_vac_status_get	none	em_id23_status_conv_get
bl_id23_sip	get	status	bl_id23_sip_status	em_id23_sip_status_get	none	em_id23_status_conv_get
bl_id23_lmt_gap	get	status	bl_id23_lmt_gap_status	em_id23_lmt_gap_status_get	none	em_id23_status_conv_get
bl_id23_lmt_phase	get	status	bl_id23_lmt_phase_status	em_id23_lmt_phase_status_get	none	em_id23_status_conv_get
bl_id23_lmt_emergency	get	status	bl_id23_lmt_emergency_status	em_id23_lmt_emergency_status_get	none	em_id23_status_conv_get
bl_id23_rfbpm_intlk	put	reset		em_idcom_do_put_pulse	none	em_std_ret
bl_id23_rfbpm_intlk	get	status	bl_id23_rfbpm_intlk_status	em_idcom_get_status	none	em_idcom_status_ret
bl_id23_rfbpm_intlk_gapa	put	reset		em_idcom_do_put_pulse	none	em_std_ret
bl_id23_rfbpm_intlk_gapa	get	status	bl_id23_rfbpm_intlk_gapa_status	em_idcom_get_status	none	em_idcom_status_ret
bl_id23_pl_1_id23rt	get	clock	s	em_get_clock	none	em_return_clock
bl_id23_pl_1_id23rt	get	clock_ms	ms	em_get_clock_ms	none	em_return_clock

signal type	bit number	bit info
bl_id23_gap_status	1	0:normal, 1:over-heat-alarm
bl_id23_phase_status	1	0:normal, 1:upp-amp-alarm
	2	0:upp-servo-no-ready, 1:ready
	3	0:upp-posi-no-complet, 1:normal
	4	0:normal, 1:upp-warning
	5	0:normal, 1:upp-brake-on
	6	0:normal, 1:low-amp-alarm
	7	0:low-servo-no-ready, 1:normal
	8	0:low-posi-no-complet, 1:normal
	9	0:normal, 1:low-warning
	10	0:normal, 1:low-brake-on
bl_id23_phase_trigger_status	1	0:trigger-off, 1:trigger-on
bl_id23_lc_status	1	0:line-off, 1:line-on
	2	0:normal, 1:fan-stop
	3	0:normal, 1:over-current
	4	0:normal, 1:over-voltage
	5	0:normal, 1:over-temperature
	6	0:local, 1:remote
	7	0:normal, 1:failure
	8	0:power-off, 1:power-on
bl_id23_st_status	1	0:line-off, 1:line-on
	2	0:normal, 1:fan-stop
	3	0:normal, 1:over-current
	4	0:normal, 1:over-voltage
	5	0:normal, 1:over-temperature
	6	0:local, 1:remote
	7	0:normal, 1:failure
	8	0:power-off, 1:power-on

bl_id23_rfbpm_status	1	0:pll-lock, 1:pll-unlock(mon1)
	2	0:pll-lock, 1:pll-unlock(mon2)
	3	0:pll-lock, 1:pll-unlock(mon3)
	4	0:pll-lock, 1:pll-unlock(mon4)
bl_id23_ivg_status	1	0:filament-off, 1:filament-on
	2	0:normal, 1:degas-on
	3	0:local, 1:remote
	4	0:setpoint1-off, 1:setpoint1-on
	5	0:setpoint2-off, 1:setpoint2-on
	6	0:gauge1, 1:gauge2
	7	0:normal, 1:alarm
	8	0:power-off, 1:power-on
bl_id23_sip_status	1	0:setpoint1-off, 1:setpoint1-on
	2	0:setpoint2-off, 1:setpoint2-on
	3	0:local, 1:remote
	4	0:normal, 1:failure
	5	0:normal, 1:over-load
	6	0:HV-off, 1:HV-on
bl_id23_lmt_gap_status	1	0:up-upstr-openLS1-on, 1:normal
	2	0:up-upstr-closLS5-on, 1:normal
	3	0:up-dowst-openLS2-on, 1:normal
	4	0:up-dowst-closLS6-on, 1:normal
	5	0:lo-upstr-openLS3-on, 1:normal
	6	0:lo-upstr-closeLS7-on, 1:normal
	7	0:lo-dowst-openLS4-on, 1:normal
	8	0:lo-dowst-closeLS8-on, 1:normal
	9	0:normal, 1:gap-refereceLS23-on
bl_id23_lmt_phase_status	1	0:row1-dowstLS9-on, 1:normal
	2	0:row1-upstLS10-on, 1:normal
	3	0:row2-dowstLS11-on, 1:normal

	4	0:row2-upstrLS12-on, 1:normal
	5	0:row3-dowstLS13-on, 1:normal
	6	0:row3-upstrLS14-on, 1:normal
	7	0:row4-dowstLS15-on, 1:normal
	8	0:row4-upstrLS16-on, 1:normal
	9	0:normal, 1:up-referenceLS24-on
	10	0:normal, 1:lo-referenceLS25-on
bl_id23_lmt_emergency_status	1	0:CSensor-uppLS17-on, 1:normal
	2	0:CSensor-lowLS18-on, 1:normal
	3	0:emergency-panel-on, 1:normal
	4	0:emergency-ID-on, 1:normal
	5	0:emergency-hall-on, 1:normal
	6	0:remote, 1:local
bl_id23_rfbpm_intlk_status	1	0:normal, 1:beam-shift2_X+on
	2	0:normal, 1:beam-shift2_X-on
	3	0:normal, 1:beam-shift2_Y+on
	4	0:normal, 1:beam-shift2_Y-on
	5	0:normal, 1:beam-shift4_X+on
	6	0:normal, 1:beam-shift4_X-on
	7	0:normal, 1:beam-shift4_Y+on
	8	0:normal, 1:beam-shift4_Y-on
	9	0:GFO, 1:not-GFO
	10	0:normal, 1:RC<1mA
	11	0:bpm-enable, 1:bpm-disable
	12	0:normal, 1:beam-shift-on
	13	0:beam-abort-required, 1:normal
bl_id23_rfbpm_intlk_gapa_status	1	0:not-all-GFO, 1:all-GFO
	2	0:notGFO&MODE, 1:GFO&MODE
	3	0:first-abort, 1:not-first-abort
	4	0:abort-disable, 1:abort-enable

sig_type_name	condition_number(1)	standard_bit	warning_bit	alert_bit
bl_id23_gap_status	1	0x0	0x1	0x0
bl_id23_phase_status	1	0xc6	0x3ff	0x0
bl_id23_phase_trigger_status	1	0x0	0x1	0x0
bl_id23_lc_status	1	0xa1	0x7f	0x0
bl_id23_st_status	1	0xa1	0x7f	0x0
bl_id23_rfbpm_status	1	0x0	0xf	0x0
bl_id23_ivg_status	1	0x99	0xff	0x0
bl_id23_sip_status	1	0x23	0x3f	0x0
bl_id23_lmt_gap_status	1	0xff	0x1ff	0x0
bl_id23_lmt_phase_status	1	0xff	0x3ff	0x0
bl_id23_lmt_emergency_status	1	0x1f	0x3f	0x0
bl_id23_rfbpm_intlk_status	1	0x1000	0x1800	0x0
bl_id23_rfbpm_intlk_gapa_status	1			
bl_id23_gap_status	2	0x0	0x1	0x0
bl_id23_phase_status	2	0xc6	0x3ff	0x0
bl_id23_phase_trigger_status	2	0x0	0x1	0x0
bl_id23_lc_status	2	0xa1	0x7f	0x0
bl_id23_st_status	2	0xa1	0x7f	0x0
bl_id23_rfbpm_status	2	0x0	0xf	0x0
bl_id23_ivg_status	2	0x99	0xff	0x0
bl_id23_sip_status	2	0x23	0x3f	0x0
bl_id23_lmt_gap_status	2	0xff	0x1ff	0x0
bl_id23_lmt_phase_status	2	0xff	0x3ff	0x0
bl_id23_lmt_emergency_status	2	0x1f	0x3f	0x0
bl_id23_rfbpm_intlk_status	2	0x1000	0x1800	0x0
bl_id23_rfbpm_intlk_gapa_status	2			

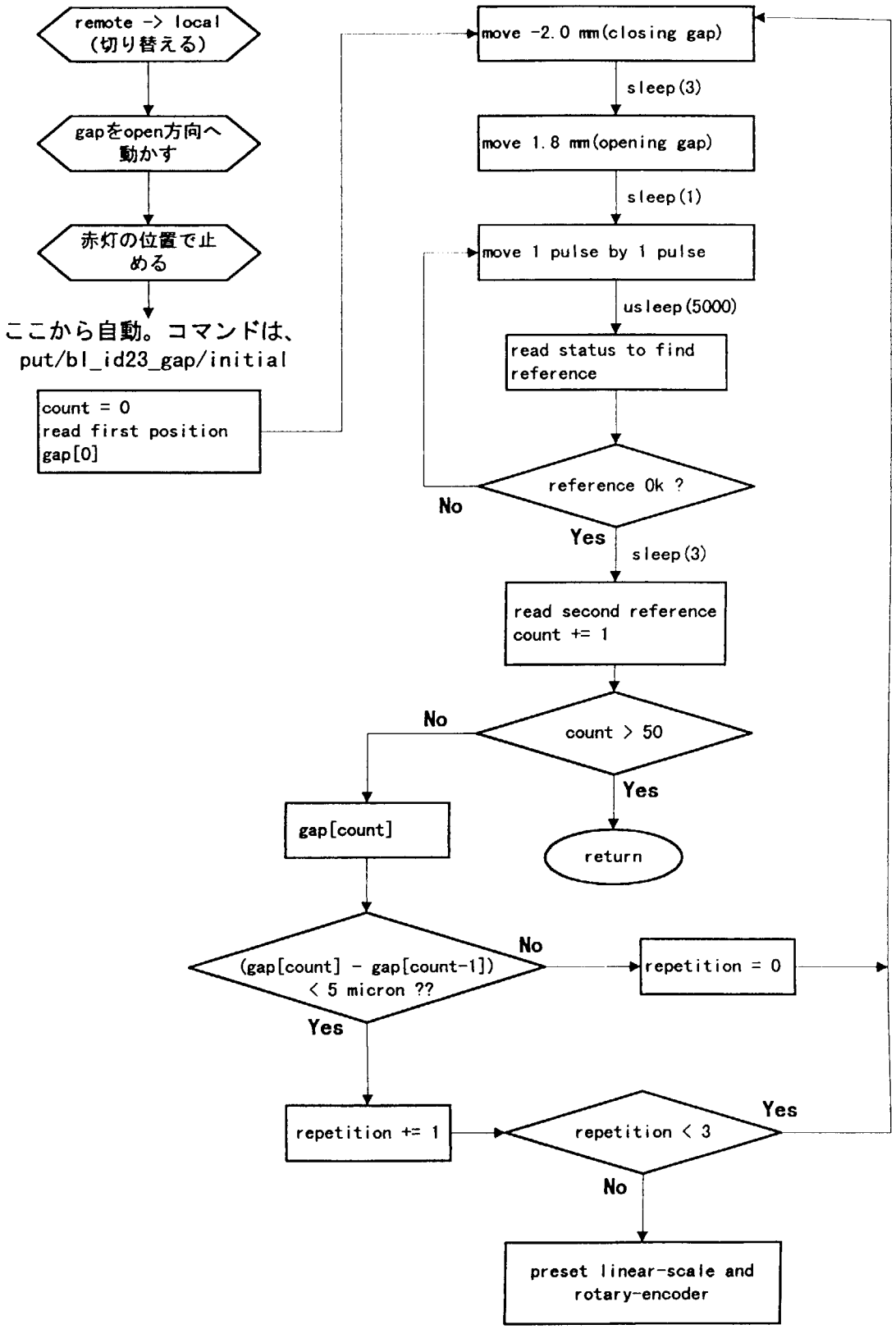
Appendix B: Status Information (reply of the command get/bl_id23.../status)

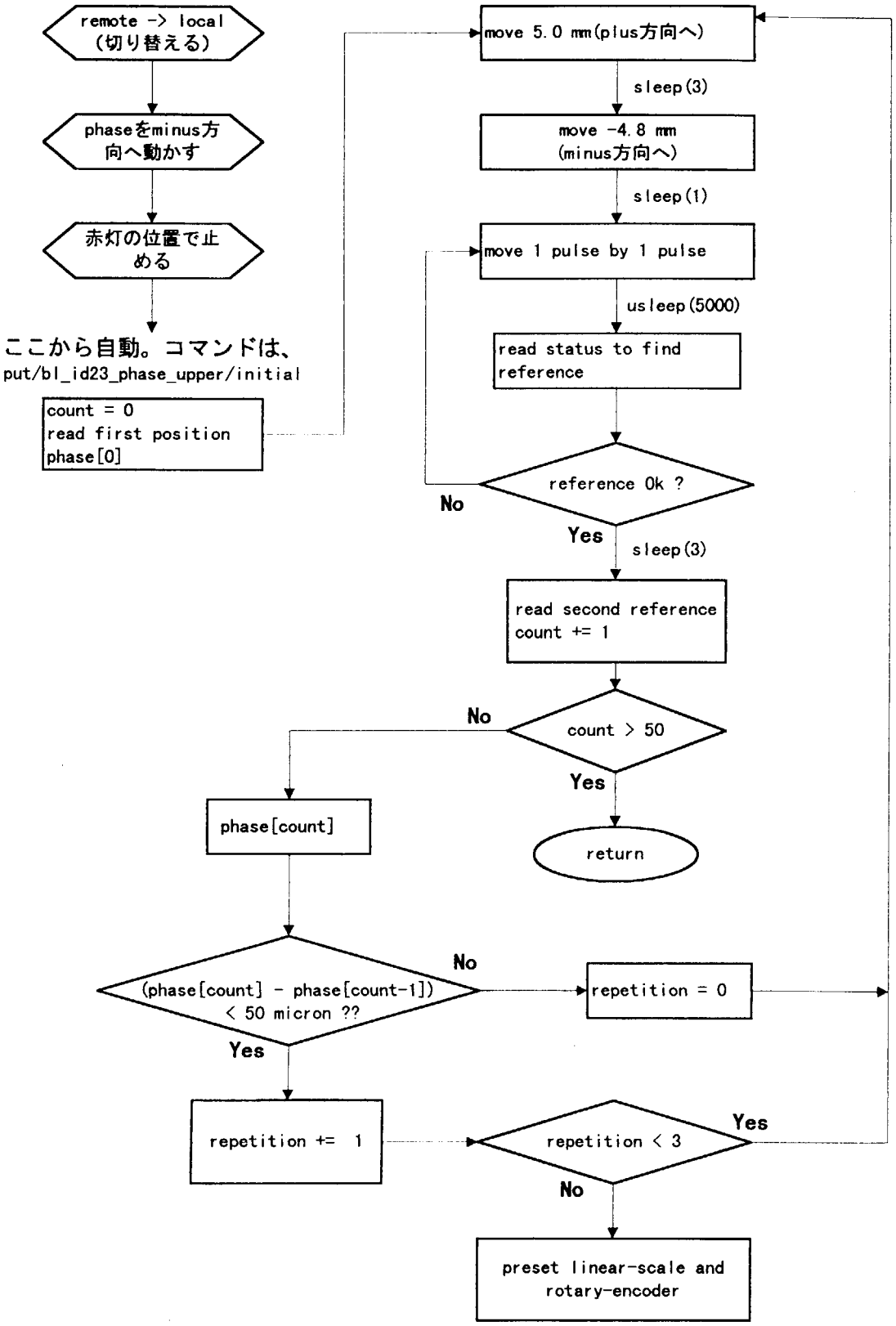
Signal Type	Device Name	Port	Channel	Bit no.	Bit Information	Normal State
bl_id23_gap_status	di_1(AVME-9421)	A	4	1	0:normal, 1:overheat-alarm	0x0
bl_id23_phase_status	di_1(AVME-9421)	A	5	1	0:normal, 1:upp-amp-alarm	0xc6(0011000110)
	di_1(AVME-9421)	A	6	2	0:upp-servo-no-ready, 1:ready	
	di_1(AVME-9421)	A	7	3	0:upp-posi-no-complet, 1:normal	
	di_1(AVME-9421)	A	8	4	0:normal, 1:upp-warning	
	di_1(AVME-9421)	A	9	5	0:normal, 1:upp-brake-on	
	di_1(AVME-9421)	A	10	6	0:normal, 1:low-amp-alarm	
	di_1(AVME-9421)	A	11	7	0:low-servo-no-ready, 1:normal	
	di_1(AVME-9421)	A	12	8	0:low-posi-no-complet, 1:normal	
	di_1(AVME-9421)	A	13	9	0:normal, 1:low-warning	
	di_1(AVME-9421)	A	14	10	0:normal, 1:low-brake-on	
bl_id23_phase_trigger_status	ttldio_0(HIMV-630)	0	8(9)	1	0:trigger-off, 1:trigger-on	0x0
bl_id23_lc_status	rio_0(MELTAC-C)	18	c	1	0:line-off, 1:line-on	0xa1(10100001)
	rio_0(MELTAC-C)	19	c	2	0:normal, 1:fan-stop	
	rio_0(MELTAC-C)	20	a	3	0:normal, 1:overcurrent	
	rio_0(MELTAC-C)	20	c	4	0:normal, 1:overvoltage	
	rio_0(MELTAC-C)	21	a	5	0:normal, 1:overtemperature	
	rio_0(MELTAC-C)	19	a	6	0:local, 1:remote	
	rio_0(MELTAC-C)	21	c	7	0:normal, 1:failure	
	rio_0(MELTAC-C)	18	a	8	0:power-off, 1:power-on	

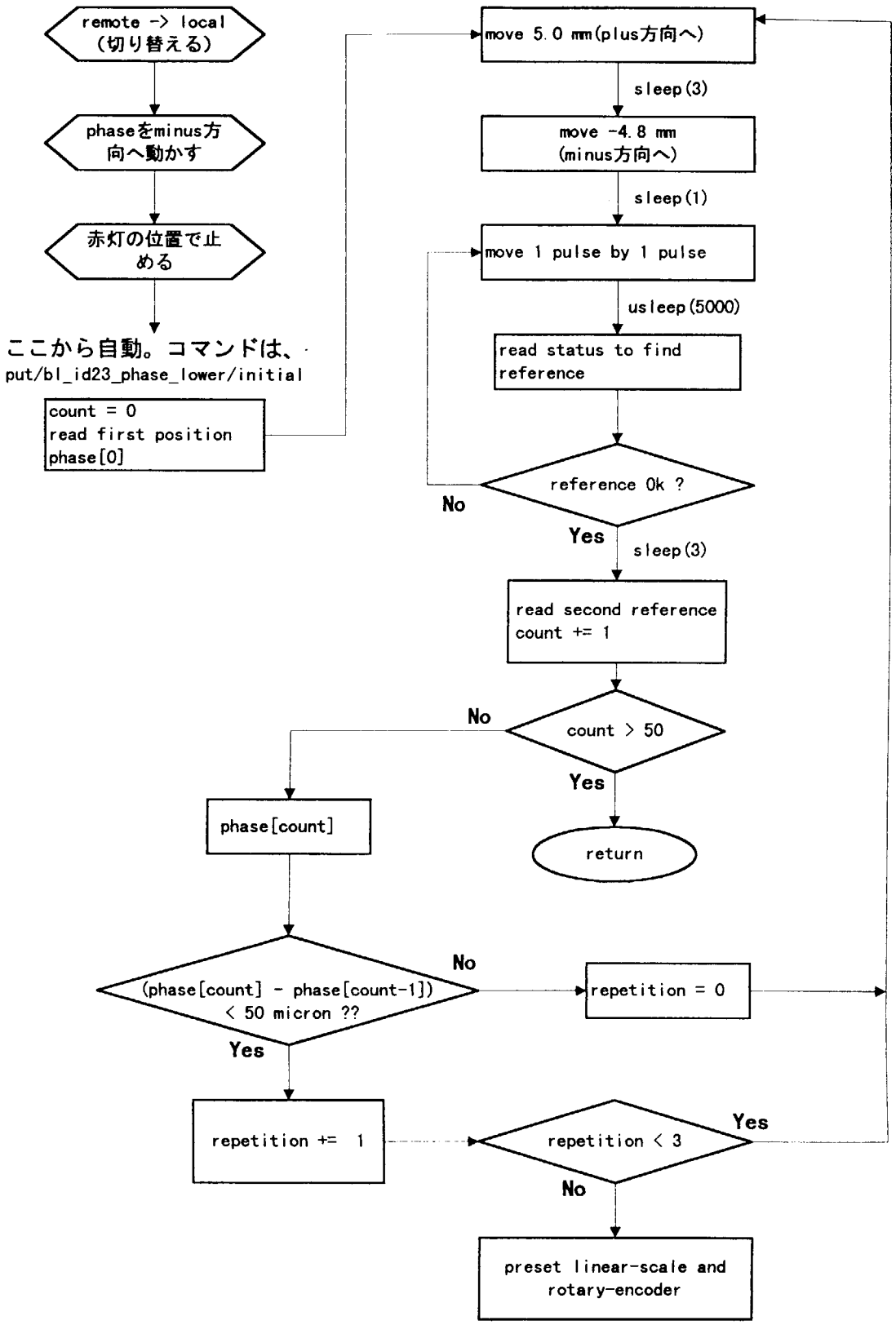
Signal Type	Device Name	Port	Channel	Bit no.	Bit Information	Normal State
bl_id23_st_status	rio_0(MELTAC-C)	18	c	1	0:line-off, 1:line-on	0xa1 (10100001)
	rio_0(MELTAC-C)	19	c	2	0:normal, 1:fan-stop	
	rio_0(MELTAC-C)	20	a	3	0:normal, 1:overcurrent	
	rio_0(MELTAC-C)	20	c	4	0:normal, 1:overvoltage	
	rio_0(MELTAC-C)	21	a	5	0:normal, 1:overtemperature	
	rio_0(MELTAC-C)	19	a	6	0:local, 1:remote	
	rio_0(MELTAC-C)	21	c	7	0:normal, 1:failure	
	rio_0(MELTAC-C)	18	a	8	0:power-off, 1:power-on	
bl_id23_rfbpm_status	ttldio_0(HIMV-630)	2	0	1	0:pll-lock, 1:pll-unlock (mon1)	0x0
	ttldio_0(HIMV-630)	2	1	2	0:pll-lock, 1:pll-unlock (mon2)	
	ttldio_0(HIMV-630)	2	2	3	0:pll-lock, 1:pll-unlock (mon3)	
	ttldio_0(HIMV-630)	2	3	4	0:pll-lock, 1:pll-unlock (mon4)	
bl_id23_ivg_status	di_2(AVME-9421)	C	0(8)	1	0:filament-off, 1:filament-on	0x99 (10011001)
	di_2(AVME-9421)	C	1(9)	2	0:normal, 1:degas-on	
	di_2(AVME-9421)	C	2(10)	3	0:local, 1:remote	
	di_2(AVME-9421)	C	3(11)	4	0:setpoint1-off, 1:setpoint1-on	
	di_2(AVME-9421)	C	4(12)	5	0:setpoint2-off, 1:setpoint2-on	
	di_2(AVME-9421)	C	5(13)	6	0:gauge1, 1:gauge2	
	di_2(AVME-9421)	C	6(14)	7	0:normal, 1:alarm	
	di_2(AVME-9421)	C	7(15)	8	0:power-off, 1:power-on	
bl_id23_sip_status	di_2(AVME-9421)	D	0	1	0:setpoint1-off, 1:setpoint1-on	0x23 (100011)
	di_2(AVME-9421)	D	1	2	0:setpoint2-off, 1:setpoint2-on	
	di_2(AVME-9421)	D	2	3	0:local, 1:remote	
	di_2(AVME-9421)	D	3	4	0:normal, 1:failure	
	di_2(AVME-9421)	D	4	5	0:normal, 1:overload	
	di_2(AVME-9421)	D	5	6	0:HV-off, 1:HV-on	

Signal Type	Device Name	Port	Channel	Bit no.	Bit Information	Normal State
bl_id23_lmt_gap_status	di_1(AVME-9421)	B	0	1	0:up-upstr-openLS1-on, 1:normal	0xFF(011111111)
	di_1(AVME-9421)	B	1	2	0:up-upstr-closLS5-on, 1:normal	
	di_1(AVME-9421)	B	2	3	0:up-dowst-openLS2-on, 1:normal	
	di_1(AVME-9421)	B	3	4	0:up-dowst-closLS6-on, 1:normal	
	di_1(AVME-9421)	B	4	5	0:lo-upstr-openLS3-on, 1:normal	
	di_1(AVME-9421)	B	5	6	0:lo-upstr-closeLS7-on, 1:normal	
	di_1(AVME-9421)	B	6	7	0:lo-dowst-openLS4-on, 1:normal	
	di_1(AVME-9421)	B	7	8	0:lo-dowst-closeLS8-on, 1:normal	
	di_1(AVME-9421)	C	6	9	0:normal, 1:gap-refereceLS23-on	
bl_id23_lmt_phase_status	di_1(AVME-9421)	B	8	1	0:row1-dowstLS9-on, 1:normal	0xFF(0011111111)
	di_1(AVME-9421)	B	9	2	0:row1-upstLS10-on, 1:normal	
	di_1(AVME-9421)	B	10	3	0:row2-dowstLS11-on, 1:normal	
	di_1(AVME-9421)	B	11	4	0:row2-upstrLS12-on, 1:normal	
	di_1(AVME-9421)	B	12	5	0:row3-dowstLS13-on, 1:normal	
	di_1(AVME-9421)	B	13	6	0:row3-upstrLS14-on, 1:normal	
	di_1(AVME-9421)	B	14	7	0:row4-dowstLS15-on, 1:normal	
	di_1(AVME-9421)	B	15	8	0:row4-upstrLS16-on, 1:normal	
	di_1(AVME-9421)	C	7	9	0:normal, 1:up-referenceLS24-on	
	di_1(AVME-9421)	C	8	10	0:normal, 1:lo-referenceLS25-on	
bl_id23_lmt_emergency_status	di_1(AVME-9421)	C	0	1	0:CSensor-uppLS17-on, 1:normal	0x1F(011111)
	di_1(AVME-9421)	C	1	2	0:CSensor-lowLS18-on, 1:normal	
	di_1(AVME-9421)	A	1	3	0:emergency-panel-on, 1:normal	
	di_1(AVME-9421)	A	2	4	0:emergency-ID-on, 1:normal	
	di_1(AVME-9421)	A	3	5	0:emergency-hall-on, 1:normal	
	di_1(AVME-9421)	A	4	6	0:remote, 1:local	

Appendix C: Flowchart of Automation to get Reference Position (for Gap, Upper-phase and Lower-phase)




remote -> local
(切り替える)
phaseをminus方向へ動かす
赤灯の位置で止める
ここから自動。コマンドは、
put/bl_id23_phase_upper/initial
count = 0
read first position
phase[0]
move 5.0 mm(plus方向へ)
sleep(3)
move -4.8 mm
(minus方向へ)
sleep(1)
move 1 pulse by 1 pulse
usleep(5000)
read status to find reference
reference Ok ?
No
Yes
sleep(3)
read second reference
count += 1
count > 50
No
Yes
return
phase[count]
(phase[count] - phase[count-1])
< 50 micron ??
No
Yes
repetition = 0
repetition += 1
repetition < 3
Yes
No
preset linear-scale and
rotary-encoder


remote -> local
(切り替える)
phaseをminus方向へ動かす
赤灯の位置で止める
ここから自動。コマンドは、
put/bl_id23_phase_lower/initial
count = 0
read first position
phase[0]
move 5.0 mm(plus方向へ)
sleep(3)
move -4.8 mm
(minus方向へ)
sleep(1)
move 1 pulse by 1 pulse
usleep(5000)
read status to find reference
reference Ok ?
No
Yes
sleep(3)
read second reference
count += 1
count > 50
No
Yes
return
phase[count]
(phase[count] - phase[count-1])
< 50 micron ??
No
repetition = 0
Yes
repetition += 1
repetition < 3
Yes
No
preset linear-scale and
rotary-encoder

ギャップの原点出し

（1）盤1で、remoteからlocalへ切り替える
（2）ギャップをオープンしながら、reference point checkが緑灯から赤灯する瞬間でギャップ駆動を止める。必ず、オープン方向へ駆動のこと。
（3）原点をメモして、5 mm程度戻す。同じ動作を１０回ほど繰り返す。最大値と最小値の差が、１０ミクロンになるまで繰り返すこと。
（4）リニアスケールの原点をセットする。プリセット値：304.565 mm
（5）ロータリエンコーダの原点をセットする。プリセット値：304565

upper_phase位相の原点出し

（1）upper_phaseをマイナス方向へ動かしながら、reference point checkが緑灯から赤灯する瞬間で位相駆動を止める。必ず、マイナス方向へ駆動のこと。
（2）原点をメモして、5 mm程度戻す。同じ動作を１０回ほど繰り返す。最大値と最小値の差が、１０ミクロンになるまで繰り返すこと。
（3）リニアスケールの原点をセットする。プリセット値：-59.563 mm
（4）ロータリエンコーダの原点をセットする。プリセット値：5956

lower_phase位相の原点出し

（1）lower_phaseをマイナス方向へ動かしながら、reference point checkが緑灯から赤灯する瞬間で位相駆動を止める。必ず、マイナス方向へ駆動のこと。
（2）原点をメモして、5 mm程度戻す。同じ動作を１０回ほど繰り返す。最大値と最小値の差が、１０ミクロンになるまで繰り返すこと。
（3）リニアスケールの原点をセットする。プリセット値：-62.107 mm
（4）ロータリエンコーダの原点をセットする。プリセット値：6211

Appendix D: Configuration Table

```
#ID23,98-01-15  Hiramatsu revised that from IHI
#     98-10-14  Hiramatsu updated
#     99-01-07  Hiramatsu updated for gapa
#     99-03-16  Hiramatsu delete put/_lc_table/exec,kind_flg for man&table
#                         modify latchon,encoder_get,latchoff for new EMfr
#     99-04-16  Hiramatsu updated for hold-time
#     99-05-04  Hiramatsu added EMA
#     99-06-17  Hiramatsu revised for GUI-EMA

put/bl_id23_pattern
   create          em_id23_pattern_create
                   none
                   em_std_ret
   destroy         em_id23_pattern_destroy
                   none
                   em_std_ret
   start           em_id23_pattern_start
                   none
                   em_std_ret
   stop            em_id23_pattern_stop
                   none
                   em_std_ret
put/bl_id23_pattern_phase
   %fmm_%fmm       em_id23_pattern_phase_put
                   none
                   em_std_ret
put/bl_id23_pattern_hold
   %dmsec_%dmsec   em_id23_pattern_hold_put
                   none
                   em_std_ret
get/bl_id23_pattern
   status          em_id23_pattern_status_get
                   none
                   em_id23_pattern_status_conv_get
get/bl_id23_pattern_phase
   position        em_id23_pattern_phase_get
                   none
                   em_id23_pattern_phase_conv_get
get/bl_id23_pattern_hold
   time            em_id23_pattern_hold_get
                   none
                   em_id23_pattern_hold_conv_get
put/bl_id23_pattern_ema
   create          em_ema_start                ema_rt
                   none
                   em_std_ret
   destroy         em_ema_stop                 ema_rt
                   none
                   em_std_ret
   start           em_ema_command              ema_rt
                   none
                   em_ema_std_ret
   stop            em_ema_command              ema_rt
                   none
                   em_ema_std_ret
   phase_r_%fmm    em_ema_command              ema_rt
                   none
                   em_ema_std_ret
   phase_l_%fmm    em_ema_command              ema_rt
                   none
                   em_ema_std_ret
   hold_r_%dmsec   em_ema_command              ema_rt
                   none
                   em_ema_std_ret
   hold_l_%dmsec   em_ema_command              ema_rt
                   none
                   em_ema_std_ret
get/bl_id23_pattern_ema
   status          em_ema_command              ema_rt
                   none
                   em_ema_std_ret
   phase_r         em_ema_command              ema_rt
                   none
```

```
                em_ema_std_ret
   phase_l      em_ema_command              ema_rt
                none
                em_ema_std_ret
   hold_r       em_ema_command              ema_rt
                none
                em_ema_std_ret
   hold_l       em_ema_command              ema_rt
                none
                em_ema_std_ret
put/bl_id23_hold
   %dmsec       em_id23_hold
                none
                em_std_ret
get/bl_id23_gap
   position     em_id23_position_gap_lin_get /dev/ttldio_0 0 100 /dev/ttldi_0
                none
                em_id23_position_conv_get    1
   status       em_id23_gap_status_get       /dev/di_1 4
                none
                em_id23_status_conv_get
put/bl_id23_gap
   %fmm         em_id23_gap_move_put_abs
                none
                em_std_ret
   initial      em_id23_gap_init_put         /dev/ptg0350_0 5000 /dev/di_1 2
                none
                em_std_ret
   preset       em_id23_gap_preset_put       /dev/do_0 16 19 50000
                none
                em_std_ret
put/bl_id23_gap_relat
   %fmm         em_id23_gap_move_put         /dev/ptg0350_0 3000 1 80 0
                none
                em_std_ret
get/bl_id23_gap_rot
   position     em_id23_position_gap_rot_get /dev/ttldio_0 4 50000 /dev/di_0
                none
                em_id23_position_conv_get    0
get/bl_id23_phase
   position     em_id23_phase_position_get
                none
                em_idcom_conv_get_pol        f3mm 0.0 1.0 0.0
   status       em_id23_phase_status_get     /dev/di_1 0
                none
                em_id23_status_conv_get
put/bl_id23_phase
   %fmm         em_id23_phase_move_put       /dev/ptg0350_2 30000 1 30 0
                em_id23_phase_conv_put       0
                em_std_ret
   reset        em_id23_phase_reset          /dev/do_0 4 5 50000
                none
                em_std_ret
put/bl_id23_phase_brake
   off          em_id23_phase_brake_off_put    /dev/do_0 10000 0 1
                none
                em_std_ret
   on           em_id23_phase_brake_on_put     /dev/do_0 10000 0 1
                none
                em_std_ret
get/bl_id23_phase_lower
   position     em_id23_position_phase_l_lin_get  /dev/ttldio_0 2 100 /dev/ttldi_0
                none
                em_id23_position_conv_get         5
put/bl_id23_phase_lower
   %fmm         em_id23_phase_move_one_side_put   /dev/ptg0350_2 10000 1 2 1 0
                em_id23_phase_conv_put            2
                em_std_ret
   initial      em_id23_phase_init_put         /dev/ptg0350_2 1 5000 /dev/di_1 2
                none
                em_std_ret
   preset       em_id23_phase_preset_put       /dev/do_0 18 21 50000
                none
```

```
                em_std_ret
get/bl_id23_phase_lower_rot
   position     em_id23_position_phase_l_rot_get  /dev/ttldio_0 6 50000 /dev/di_0
                none
                em_id23_position_conv_get        3
put/bl_id23_phase_trigger_r
   off          em_id23_phase_trigger_off_put    /dev/ttldio_0 8
                none
                em_std_ret
   on           em_id23_phase_trigger_on_put     /dev/ttldio_0 8
                none
                em_std_ret
get/bl_id23_phase_trigger_r
   status       em_id23_phase_trigger_status_get  /dev/ttldio_0 8
                none
                em_id23_status_conv_get
put/bl_id23_phase_trigger_l
   off          em_id23_phase_trigger_off_put    /dev/ttldio_0 9
                none
                em_std_ret
   on           em_id23_phase_trigger_on_put     /dev/ttldio_0 9
                none
                em_std_ret
get/bl_id23_phase_trigger_l
   status       em_id23_phase_trigger_status_get  /dev/ttldio_0 9
                none
                em_id23_status_conv_get
get/bl_id23_phase_upper
   position     em_id23_position_phase_u_lin_get    /dev/ttldio_0 1 100 /dev/ttldi_0
                none
                em_id23_position_conv_get          4
put/bl_id23_phase_upper
   %fmm         em_id23_phase_move_one_side_put   /dev/ptg0350_2 10000 1 2 0 0
                em_id23_phase_conv_put            1
                em_std_ret
   initial      em_id23_phase_init_put        /dev/ptg0350_2 0 5000 /dev/di_1 2
                none
                em_std_ret
   preset       em_id23_phase_preset_put      /dev/do_0 17 20 50000
                none
                em_std_ret
get/bl_id23_phase_upper_rot
   position     em_id23_position_phase_u_rot_get  /dev/ttldio_0 5 50000 /dev/di_0
                none
                em_id23_position_conv_get         2
get/bl_id23_lc_h
   current      em_id23_st_adc_curr_get       /dev/rio_0 9
                none
                em_id23_st_curr_conv_get      0.00046876431 -15.36
   status       em_id23_st_status_get         /dev/rio_0 9
                none
                em_id23_status_conv_get
put/bl_id23_lc_h
   off          em_id23_st_power_off_put      /dev/rio_0 9 1 50000 32767
                none
                em_std_ret
   on           em_id23_st_power_on_put       /dev/rio_0 9 0 50000 32767
                none
                em_std_ret
   reset        em_id23_st_err_reset          /dev/rio_0 9 2 50000
                none
                em_std_ret
set/bl_id23_lc_h
   %fA          em_id23_st_curr_set           8
                none
                em_std_ret
get/bl_id23_lc_v
   current      em_id23_st_adc_curr_get       /dev/rio_0 10
                none
                em_id23_st_curr_conv_get      0.00046876431 -15.36
   status       em_id23_st_status_get         /dev/rio_0 10
                none
                em_id23_status_conv_get
```

```
put/bl_id23_lc_v
   off           em_id23_st_power_off_put      /dev/rio_0 10 1 50000 32767
                 none
                 em_std_ret
   on            em_id23_st_power_on_put       /dev/rio_0 10 0 50000 32767
                 none
                 em_std_ret
   reset         em_id23_st_err_reset          /dev/rio_0 10 2 50000
                 none
                 em_std_ret
set/bl_id23_lc_v
   %fA           em_id23_st_curr_set           9
                 none
                 em_std_ret
get/bl_id23_lc_dac_h
   current       em_id23_st_dac_curr_get       /dev/rio_0 9
                 none
                 em_id23_st_curr_conv_get      0.00045777764 -15.0
get/bl_id23_lc_dac_v
   current       em_id23_st_dac_curr_get       /dev/rio_0 10
                 none
                 em_id23_st_curr_conv_get      0.00045777764 -15.0
put/bl_id23_lc_table
   %s            em_id23_lc_table_thru         /home/bl/blcntl/bl_id23_
                 em_id23_lc_table_set
                 em_std_ret
put/bl_id23_st
   exec          em_id23_st_curr_put           /dev/rio_0 1 2 3 4 5 6 7 8 9 10
                 em_id23_st_curr_exec          2184.466667 32767.0
                 em_std_ret
get/bl_id23_st_h_1
   current       em_id23_st_adc_curr_get       /dev/rio_0 1
                 none
                 em_id23_st_curr_conv_get      0.00046876431 -15.36
   status        em_id23_st_status_get         /dev/rio_0 1
                 none
                 em_id23_status_conv_get
put/bl_id23_st_h_1
   off           em_id23_st_power_off_put      /dev/rio_0 1 1 50000 32767
                 none
                 em_std_ret
   on            em_id23_st_power_on_put       /dev/rio_0 1 0 50000 32767
                 none
                 em_std_ret
   reset         em_id23_st_err_reset          /dev/rio_0 1 2 50000
                 none
                 em_std_ret
set/bl_id23_st_h_1
   %fA           em_id23_st_curr_set           0
                 none
                 em_std_ret
get/bl_id23_st_dac_h_1
   current       em_id23_st_dac_curr_get       /dev/rio_0 1
                 none
                 em_id23_st_curr_conv_get      0.00045777764 -15.0
get/bl_id23_st_v_1
   current       em_id23_st_adc_curr_get       /dev/rio_0 2
                 none
                 em_id23_st_curr_conv_get      0.00046876431 -15.36
   status        em_id23_st_status_get         /dev/rio_0 2
                 none
                 em_id23_status_conv_get
put/bl_id23_st_v_1
   off           em_id23_st_power_off_put      /dev/rio_0 2 1 50000 32767
                 none
                 em_std_ret
   on            em_id23_st_power_on_put       /dev/rio_0 2 0 50000 32767
                 none
                 em_std_ret
   reset         em_id23_st_err_reset          /dev/rio_0 2 2 50000
                 none
                 em_std_ret
set/bl_id23_st_v_1
```

```
   %fA             em_id23_st_curr_set          1
                   none
                   em_std_ret
get/bl_id23_st_dac_v_1
   current         em_id23_st_dac_curr_get      /dev/rio_0 2
                   none
                   em_id23_st_curr_conv_get     0.00045777764 -15.0
get/bl_id23_st_h_2
   current         em_id23_st_adc_curr_get      /dev/rio_0 3
                   none
                   em_id23_st_curr_conv_get     0.00046876431 -15.36
   status          em_id23_st_status_get        /dev/rio_0 3
                   none
                   em_id23_status_conv_get
put/bl_id23_st_h_2
   off             em_id23_st_power_off_put     /dev/rio_0 3 1 50000 32767
                   none
                   em_std_ret
   on              em_id23_st_power_on_put      /dev/rio_0 3 0 50000 32767
                   none
                   em_std_ret
   reset           em_id23_st_err_reset         /dev/rio_0 3 2 50000
                   none
                   em_std_ret
set/bl_id23_st_h_2
   %fA             em_id23_st_curr_set          2
                   none
                   em_std_ret
get/bl_id23_st_dac_h_2
   current         em_id23_st_dac_curr_get      /dev/rio_0 3
                   none
                   em_id23_st_curr_conv_get     0.00045777764 -15.0
get/bl_id23_st_v_2
   current         em_id23_st_adc_curr_get      /dev/rio_0 4
                   none
                   em_id23_st_curr_conv_get     0.00046876431 -15.36
   status          em_id23_st_status_get        /dev/rio_0 4
                   none
                   em_id23_status_conv_get
put/bl_id23_st_v_2
   off             em_id23_st_power_off_put     /dev/rio_0 4 1 50000 32767
                   none
                   em_std_ret
   on              em_id23_st_power_on_put      /dev/rio_0 4 0 50000 32767
                   none
                   em_std_ret
   reset           em_id23_st_err_reset         /dev/rio_0 4 2 50000
                   none
                   em_std_ret
set/bl_id23_st_v_2
   %fA             em_id23_st_curr_set          3
                   none
                   em_std_ret
get/bl_id23_st_dac_v_2
   current         em_id23_st_dac_curr_get      /dev/rio_0 4
                   none
                   em_id23_st_curr_conv_get     0.00045777764 -15.0
get/bl_id23_st_h_3
   current         em_id23_st_adc_curr_get      /dev/rio_0 5
                   none
                   em_id23_st_curr_conv_get     0.00046876431 -15.36
   status          em_id23_st_status_get        /dev/rio_0 5
                   none
                   em_id23_status_conv_get
put/bl_id23_st_h_3
   off             em_id23_st_power_off_put     /dev/rio_0 5 1 50000 32767
                   none
                   em_std_ret
   on              em_id23_st_power_on_put      /dev/rio_0 5 0 50000 32767
                   none
                   em_std_ret
   reset           em_id23_st_err_reset         /dev/rio_0 5 2 50000
                   none
```

```
                em_std_ret
set/bl_id23_st_h_3
   %fA          em_id23_st_curr_set          4
                none
                em_std_ret
get/bl_id23_st_dac_h_3
   current      em_id23_st_dac_curr_get      /dev/rio_0 5
                none
                em_id23_st_curr_conv_get     0.00045777764 -15.0
get/bl_id23_st_v_3
   current      em_id23_st_adc_curr_get      /dev/rio_0 6
                none
                em_id23_st_curr_conv_get     0.00046876431 -15.36
   status       em_id23_st_status_get        /dev/rio_0 6
                none
                em_id23_status_conv_get
put/bl_id23_st_v_3
   off          em_id23_st_power_off_put     /dev/rio_0 6 1 50000 32767
                none
                em_std_ret
   on           em_id23_st_power_on_put      /dev/rio_0 6 0 50000 32767
                none
                em_std_ret
   reset        em_id23_st_err_reset         /dev/rio_0 6 2 50000
                none
                em_std_ret
set/bl_id23_st_v_3
   %fA          em_id23_st_curr_set          5
                none
                em_std_ret
get/bl_id23_st_dac_v_3
   current      em_id23_st_dac_curr_get      /dev/rio_0 6
                none
                em_id23_st_curr_conv_get     0.00045777764 -15.0
get/bl_id23_st_h_4
   current      em_id23_st_adc_curr_get      /dev/rio_0 7
                none
                em_id23_st_curr_conv_get     0.00046876431 -15.36
   status       em_id23_st_status_get        /dev/rio_0 7
                none
                em_id23_status_conv_get
put/bl_id23_st_h_4
   off          em_id23_st_power_off_put     /dev/rio_0 7 1 50000 32767
                none
                em_std_ret
   on           em_id23_st_power_on_put      /dev/rio_0 7 0 50000 32767
                none
                em_std_ret
   reset        em_id23_st_err_reset         /dev/rio_0 7 2 50000
                none
                em_std_ret
set/bl_id23_st_h_4
   %fA          em_id23_st_curr_set          6
                none
                em_std_ret
get/bl_id23_st_dac_h_4
   current      em_id23_st_dac_curr_get      /dev/rio_0 7
                none
                em_id23_st_curr_conv_get     0.00045777764 -15.0
get/bl_id23_st_v_4
   current      em_id23_st_adc_curr_get      /dev/rio_0 8
                none
                em_id23_st_curr_conv_get     0.00046876431 -15.36
   status       em_id23_st_status_get        /dev/rio_0 8
                none
                em_id23_status_conv_get
put/bl_id23_st_v_4
   off          em_id23_st_power_off_put     /dev/rio_0 8 1 50000 32767
                none
                em_std_ret
   on           em_id23_st_power_on_put      /dev/rio_0 8 0 50000 32767
                none
                em_std_ret
```

```
   reset          em_id23_st_err_reset          /dev/rio_0 8 2 50000
                  none
                  em_std_ret
set/bl_id23_st_v_4
   %fA            em_id23_st_curr_set           7
                  none
                  em_std_ret
get/bl_id23_st_dac_v_4
   current        em_id23_st_dac_curr_get       /dev/rio_0 8
                  none
                  em_id23_st_curr_conv_get      0.00045777764 -15.0
get/bl_id23_rfbpm
   status         em_id23_rfbpm_status_get      /dev/ttldio_0 2
                  none
                  em_id23_status_conv_get
get/bl_id23_rfbpm_1_x
   position       em_id23_rfbpm_position_get    /dev/adc311_0 0 5 20
                  none
                  em_id23_rfbpm_convmm_get      0.000305175 -10.0
   voltage        em_id23_rfbpm_position_get    /dev/adc311_0 0 5 20
                  none
                  em_id23_rfbpm_conv_get        0.000305175 -10.0
get/bl_id23_rfbpm_1_y
   position       em_id23_rfbpm_position_get    /dev/adc311_0 1 5 20
                  none
                  em_id23_rfbpm_convmm_get      0.000305175 -10.0
   voltage        em_id23_rfbpm_position_get    /dev/adc311_0 1 5 20
                  none
                  em_id23_rfbpm_conv_get        0.000305175 -10.0
get/bl_id23_rfbpm_2_x
   position       em_id23_rfbpm_position_get    /dev/adc311_0 2 5 20
                  none
                  em_id23_rfbpm_convmm_get      0.000305175 -10.0
   voltage        em_id23_rfbpm_position_get    /dev/adc311_0 2 5 20
                  none
                  em_id23_rfbpm_conv_get        0.000305175 -10.0
get/bl_id23_rfbpm_2_y
   position       em_id23_rfbpm_position_get    /dev/adc311_0 3 5 20
                  none
                  em_id23_rfbpm_convmm_get      0.000305175 -10.0
   voltage        em_id23_rfbpm_position_get    /dev/adc311_0 3 5 20
                  none
                  em_id23_rfbpm_conv_get        0.000305175 -10.0
get/bl_id23_rfbpm_3_x
   position       em_id23_rfbpm_position_get    /dev/adc311_0 4 5 20
                  none
                  em_id23_rfbpm_convmm_get      0.000305175 -10.0
   voltage        em_id23_rfbpm_position_get    /dev/adc311_0 4 5 20
                  none
                  em_id23_rfbpm_conv_get        0.000305175 -10.0
get/bl_id23_rfbpm_3_y
   position       em_id23_rfbpm_position_get    /dev/adc311_0 5 5 20
                  none
                  em_id23_rfbpm_convmm_get      0.000305175 -10.0
   voltage        em_id23_rfbpm_position_get    /dev/adc311_0 5 5 20
                  none
                  em_id23_rfbpm_conv_get        0.000305175 -10.0
get/bl_id23_rfbpm_4_x
   position       em_id23_rfbpm_position_get    /dev/adc311_0 6 5 20
                  none
                  em_id23_rfbpm_convmm_get      0.000305175 -10.0
   voltage        em_id23_rfbpm_position_get    /dev/adc311_0 6 5 20
                  none
                  em_id23_rfbpm_conv_get        0.000305175 -10.0
get/bl_id23_rfbpm_4_y
   position       em_id23_rfbpm_position_get    /dev/adc311_0 7 5 20
                  none
                  em_id23_rfbpm_convmm_get      0.000305175 -10.0
   voltage        em_id23_rfbpm_position_get    /dev/adc311_0 7 5 20
                  none
                  em_id23_rfbpm_conv_get        0.000305175 -10.0
get/bl_id23_ivg_1
   pressure       em_id23_vac_get               /dev/di_2 0
```

```
                none
                em_id23_vac_conv_get
   status       em_id23_vac_status_get        /dev/di_2 2 0
                none
                em_id23_status_conv_get
get/bl_id23_ivg_2
   pressure     em_id23_vac_get               /dev/di_2 1
                none
                em_id23_vac_conv_get
   status       em_id23_vac_status_get        /dev/di_2 2 1
                none
                em_id23_status_conv_get
get/bl_id23_sip
   status       em_id23_sip_status_get        /dev/di_2 3
                none
                em_id23_status_conv_get
get/bl_id23_lmt_emergency
   status       em_id23_lmt_emergency_status_get  /dev/di_1 2 0
                none
                em_id23_status_conv_get
get/bl_id23_lmt_gap
   status       em_id23_lmt_gap_status_get    /dev/di_1 1 2
                none
                em_id23_status_conv_get
get/bl_id23_lmt_phase
   status       em_id23_lmt_phase_status_get  /dev/di_1 1 2
                none
                em_id23_status_conv_get
put/bl_id23_rfbpm_intlk
   reset        em_idcom_do_put_pulse         /dev/dio602a_0 0 10
                none
                em_std_ret
get/bl_id23_rfbpm_intlk
   status       em_idcom_get_status           /dev/dio602a_0 0 1 2 3 4 5 6 7 8 9 10 11 12
                none
                em_idcom_status_ret
put/bl_id23_rfbpm_intlk_gapa
   reset        em_idcom_do_put_pulse         /dev/dio602a_0 1 10
                none
                em_std_ret
get/bl_id23_rfbpm_intlk_gapa
   status       em_idcom_get_status           /dev/dio602a_1 0 1 2 3 4
                none
                em_idcom_status_ret
get/bl_id23_pl_1_id23rt
   clock        em_get_clock
                none
                em_return_clock    s
   clock_ms     em_get_clock_ms
                none
                em_return_clock    ms
```

Appendix E: Correction Table of 10 Steering Magnets

```
# Magnetic Field Correction Table  98-10-2  updated during machine study
# Correction Table  99-05-08  added for 4 AC-steering magnets (dummy data)
<phase_h>
-121.,-40.,-30.,-20.,-10., 0. ,10., 20., 30., 40.,121.
</phase_h>
#
<Ic_h>
35. ,2.000,2.000,2.300,2.200,1.900,1.700,1.900,2.100,2.300,1.800,1.800
36. ,2.000,2.000,2.300,2.200,1.900,1.700,1.900,2.100,2.300,1.800,1.800
40. ,2.201,2.201,2.436,2.365,2.120,1.950,2.120,2.289,2.444,2.065,2.065
50. ,2.522,2.522,2.653,2.630,2.471,2.350,2.471,2.591,2.675,2.488,2.488
60. ,2.803,2.803,2.843,2.862,2.778,2.700,2.778,2.856,2.877,2.859,2.859
80. ,2.850,2.850,2.875,2.900,2.829,2.758,2.829,2.900,2.910,2.920,2.920
90. ,2.893,2.893,2.919,2.944,2.872,2.800,2.872,2.944,2.954,2.964,2.964
100.,2.893,2.893,2.919,2.944,2.872,2.800,2.872,2.944,2.954,2.964,2.964
120.,2.265,2.265,2.285,2.305,2.248,2.192,2.248,2.305,2.313,2.321,2.321
150.,1.860,1.860,1.876,1.893,1.846,1.800,1.846,1.893,1.899,1.906,1.906
300.,0.0  ,0.   ,0.   ,0.   ,0.   ,0.   ,0.   ,0.   ,0.   ,0.   ,0.0
350.,0.0  ,0.   ,0.   ,0.   ,0.   ,0.   ,0.   ,0.   ,0.   ,0.   ,0.0
</Ic_h>
#
<phase_v>
-121.,-60.,-50.,-40.,-30.,-20.,-10.,-5., 0., 5., 10., 20., 30., 40., 50., 60.,121.
</phase_v>
#
<Ic_v>
35. ,-2.59,-2.59,-2.59,-2.59,-2.59,-2.59,-2.59,-2.59,-2.59,-2.59,-2.59,-2.59,-2.59,-2.59,-2.59,-2.59,-2.59,-2.59,-2.59
36. ,-2.59,-2.59,-2.59,-2.59,-2.59,-2.59,-2.59,-2.59,-2.59,-2.59,-2.59,-2.59,-2.59,-2.59,-2.59,-2.59,-2.59,-2.59,-2.59
40. ,-2.03,-2.03,-2.03,-2.03,-2.03,-2.03,-2.03,-2.03,-2.03,-2.03,-2.03,-2.03,-2.03,-2.03,-2.03,-2.03,-2.03,-2.03,-2.03
45. ,-1.39,-1.39,-1.39,-1.39,-1.39,-1.39,-1.39,-1.39,-1.39,-1.39,-1.39,-1.39,-1.39,-1.39,-1.39,-1.39,-1.39,-1.39,-1.39
50. ,-0.98,-0.98,-0.98,-0.98,-0.98,-0.98,-0.98,-0.98,-0.98,-0.98,-0.98,-0.98,-0.98,-0.98,-0.98,-0.98,-0.98,-0.98,-0.98
55. ,-0.64,-0.64,-0.64,-0.64,-0.64,-0.64,-0.64,-0.64,-0.64,-0.64,-0.64,-0.64,-0.64,-0.64,-0.64,-0.64,-0.64,-0.64,-0.64
60. ,-0.41,-0.41,-0.41,-0.41,-0.41,-0.41,-0.41,-0.41,-0.41,-0.41,-0.41,-0.41,-0.41,-0.41,-0.41,-0.41,-0.41,-0.41,-0.41
70. ,0.089,0.089,0.089,0.089,0.089,0.089,0.089,0.089,0.089,0.089,0.089,0.089,0.089,0.089,0.089,0.089,0.089,0.089,0.089
80. ,0.386,0.386,0.386,0.386,0.386,0.386,0.386,0.386,0.386,0.386,0.386,0.386,0.386,0.386,0.386,0.386,0.386,0.386,0.386
90. ,0.561,0.561,0.561,0.561,0.561,0.561,0.561,0.561,0.561,0.561,0.561,0.561,0.561,0.561,0.561,0.561,0.561,0.561,0.561
100.,0.61 ,0.61 ,0.61 ,0.61 ,0.61 ,0.61 ,0.61 ,0.61 ,0.61 ,0.61 ,0.61 ,0.61 ,0.61 ,0.61 ,0.61 ,0.61 ,0.61 ,0.61 ,0.61
110.,0.566,0.566,0.566,0.566,0.566,0.566,0.566,0.566,0.566,0.566,0.566,0.566,0.566,0.566,0.566,0.566,0.566,0.566,0.566
120.,0.687,0.687,0.687,0.687,0.687,0.687,0.687,0.687,0.687,0.687,0.687,0.687,0.687,0.687,0.687,0.687,0.687,0.687,0.687
150.,0.674,0.674,0.674,0.674,0.674,0.674,0.674,0.674,0.674,0.674,0.674,0.674,0.674,0.674,0.674,0.674,0.674,0.674,0.674
200.,0.461,0.461,0.461,0.461,0.461,0.461,0.461,0.461,0.461,0.461,0.461,0.461,0.461,0.461,0.461,0.461,0.461,0.461,0.461
250.,0.241,0.241,0.241,0.241,0.241,0.241,0.241,0.241,0.241,0.241,0.241,0.241,0.241,0.241,0.241,0.241,0.241,0.241,0.241
300.,0.0  ,0.   ,0.   ,0.   ,0.   ,0.   ,0.   ,0.   ,0.   ,0.   ,0.   ,0.   ,0.   ,0.   ,0.   ,0.   ,0.   ,0.   ,0.0
350.,0.0  ,0.   ,0.   ,0.   ,0.   ,0.   ,0.   ,0.   ,0.   ,0.   ,0.   ,0.   ,0.   ,0.   ,0.   ,0.   ,0.   ,0.   ,0.0
</Ic_v>
#
<s1phase_h>
-121.,-40.,-30.,-20.,-10., 0. ,10., 20., 30., 40.,121.
</s1phase_h>
#
<s1_h>
35. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
36. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
40. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
```

```
50. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
60. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
80. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
90. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
100.,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
120.,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
150.,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
300.,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
350.,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
</s1_h>
#
<s1phase_v>
-121.,-40.,-30.,-20.,-10., 0. ,10., 20., 30., 40.,121.
</s1phase_v>
#
<s1_v>
35. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
36. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
40. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
50. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
60. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
80. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
90. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
100.,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
120.,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
150.,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
300.,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
350.,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
</s1_v>
#
<s2phase_h>
-121.,-40.,-30.,-20.,-10., 0. ,10., 20., 30., 40.,121.
</s2phase_h>
#
<s2_h>
35. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
36. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
40. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
50. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
60. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
80. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
90. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
100.,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
120.,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
150.,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
300.,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
350.,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
</s2_h>
#
<s2phase_v>
-121.,-40.,-30.,-20.,-10., 0. ,10., 20., 30., 40.,121.
</s2phase_v>
#
<s2_v>
35. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
36. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
40. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
50. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
60. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
80. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
90. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
100.,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
120.,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
150.,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
300.,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
350.,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
</s2_v>
#
<s3phase_h>
-121.,-40.,-30.,-20.,-10., 0. ,10., 20., 30., 40.,121.
</s3phase_h>
#
<s3_h>
35. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
36. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
```

```
40. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
50. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
60. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
80. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
90. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
100.,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
120.,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
150.,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
300.,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
350.,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
</s3_h>
#
<s3phase_v>
-121.,-40.,-30.,-20.,-10., 0. ,10., 20., 30., 40.,121.
</s3phase_v>
#
<s3_v>
35. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
36. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
40. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
50. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
60. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
80. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
90. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
100.,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
120.,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
150.,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
300.,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
350.,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
</s3_v>
#
<s4phase_h>
-121.,-40.,-30.,-20.,-10., 0. ,10., 20., 30., 40.,121.
</s4phase_h>
#
<s4_h>
35. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
36. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
40. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
50. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
60. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
80. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
90. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
100.,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
120.,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
150.,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
300.,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
350.,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
</s4_h>
#
<s4phase_v>
-121.,-40.,-30.,-20.,-10., 0. ,10., 20., 30., 40.,121.
</s4phase_v>
#
<s4_v>
35. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
36. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
40. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
50. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
60. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
80. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
90. ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
100.,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
120.,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
150.,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
300.,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
350.,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0  ,0.0
</s4_v>
#
#
```

Appendix F: Makefile for Simulator (tellem, ema_rt and rpc_sim)

```
# Library Makefile for ID23
#         97-12-10 Hiramatsu  created
#         99-04-03 Hiramatsu, for simulator
#         99-05-07 Hiramatsu, for EMA
#
# Compiler parameter
INC             = -I/HP-RT/usr/include -I/prj/sr/include  ¥
                  -I../../include       -I/prj/bl/include
DEBFLAG         = -g -DEM_DEBUG_PRINTF
SIMFLAG         = -DEM_DEBUG_SIM
TELLEMFLAG      = -DEM_DEBUG_TELLEM
CCRT            = ccrt  -w -Ae
MAKE            = make

# Library parameter
LIBNAME         = em_id23
TARGETDIR       = ../../lib

# Source and Objects File
SRCS            = *.c
OBJS            = *.o
RMOBJS          = em_poller_init.o em_poller_terminate.o

# Target Rule
#all : librt_$(LIBNAME).a librt_$(LIBNAME)_deb.a
all :  librt_$(LIBNAME)_sim.a

librt_$(LIBNAME).a: $(SRCS)
                @echo $@ " making start"
                @rm -f *.o
                $(CCRT) $(TELLEMFLAG) -c $(INC) $(SRCS)
                $(MAKE) lib NAME=$@
                @echo $@ " making end"

librt_$(LIBNAME)_deb.a: $(SRCS)
                @echo $@ " making start"
                @rm -f *.o
                $(CCRT) $(TELLEMFLAG) $(DEBFLAG) -c $(INC) $(SRCS)
                $(MAKE) lib NAME=$@
                @echo $@ " making end"

librt_$(LIBNAME)_sim.a: $(SRCS)
                @echo $@ " making start"
                @rm -f *.o
                $(CCRT) -c $(SIMFLAG) $(DEBFLAG) $(INC) $(SRCS)
                $(MAKE) lib NAME=$@
                @echo $@ " making end"

lib::
                @rm -f $(RMOBJS)
                @rm -f $(TARGETDIR)/$(NAME)
                ar cru $(TARGETDIR)/$(NAME) $(OBJS)
                @rm -f $(OBJS)

clean::         @rm -f $(OBJS) core
```

```
# Makefile for tellem & EMA
#          99-05-07  Hiramatsu  created
#
# Compiler parameter
EQUIPLIB     = id23
EQUIP        = ID23
INCPATH      = -I/prj/sr/include -I../../include -I/prj/bl/include
LIBPATH      = -L/prj/sr/lib     -L../../lib     -L/prj/bl/lib
LIBPATHRT    = -L/HP-RT/lib   -L/HP-RT/usr/lib
LIBPATHDEB   = -L/prj/sr/em/frame/lib
TEMPLATEPATH = /prj/sr/em/ema/src
RTDDLIB      = -lrt_devapi_rio    -lrt_drvlib_rio     -lrt_devapi_di     ¥
               -lrt_devapi_do     -lrt_devapi_ttldi   -lrt_devapi_ttldio ¥
               -lrt_drvlib_ttldio -lrt_devapi_ptg0350 -lrt_devapi_adc311 ¥
               -lrt_devapi_dio_himv602a  -lrt_devapi_com
LSRRT        = -lrt_cp  -lrt_util
RTSYSLIB     = -lp -lm -lc -lrpc -lbsd
LRTREAL      = -lrt_em_$(EQUIPLIB)       -lrt_em_frame      -lrt_em_idcom ¥
               -lrt_em_$(EQUIPLIB)       -lrt_em_frame_template
LRTREALDEB   = -lrt_em_$(EQUIPLIB)_deb  -lrt_em_frame      -lrt_em_idcom ¥
               -lrt_em_$(EQUIPLIB)_deb  -lrt_em_frame_template
LRTSIM       = -lrt_em_$(EQUIPLIB)_sim  -lrt_em_frame_sim  -lrt_em_idcom ¥
               -lrt_em_$(EQUIPLIB)_sim  -lrt_em_frame_template
LRTEMA       = -lrt_em_$(EQUIPLIB)  -lrt_em_ema    -lrt_em_frame ¥
               -lrt_em_idcom  -lrt_em_$(EQUIPLIB)  -lrt_em_frame_template
LRTEMASIM    = -lrt_em_$(EQUIPLIB)_sim  -lrt_em_ema_sim    -lrt_em_frame_sim ¥
               -lrt_em_idcom  -lrt_em_$(EQUIPLIB)_sim  -lrt_em_frame_template
DFLAG        = -D$(EQUIP)
CCRT         = ccrt -w -Ae $(INCPATH) $(LIBPATH) $(LIBPATHRT) $(DFLAG)

# Executable file
#EXEFILE     = tellem  tellem_deb
EXEFILE      = tellem_sim
#EXEFILE      = ema_rt

# Source and Object file
SRCS         = *.c
OBJS         = *.o

# Precompiler parameter
DEV_CONFIG   = config.tbl
FUNC_PTR_SRC = em_config_get_func_ptr.c
PRECOMP_PERL = pre_em_config_get_func_ptr.pl

# Target
all: $(FUNC_PTR_SRC) $(EXEFILE)

tellem:;
        @echo $@ "Making..."
        $(CCRT) $(SRCS) -o $@  $(LRTEMA) ¥
        $(LRTREAL) $(RTDDLIB) $(LSRRT) $(RTSYSLIB)

tellem_deb:;
        @echo $@ "Making..."
        $(CCRT) $(SRCS) -o $@  $(LRTEMA) ¥
        $(LIBPATHDEB) $(LRTREALDEB) $(RTDDLIB) $(LSRRT) $(RTSYSLIB)

tellem_sim:;
        @echo $@ "Making..."
        $(CCRT) $(SRCS) -o $@  $(LRTEMASIM) ¥
        $(LRTSIM) $(RTDDLIB) $(LSRRT) $(RTSYSLIB)

ema_rt:;
        @echo $@ "Making..."
        $(CCRT) $(SRCS) $(TEMPLATEPATH)/em_ema_main.c -o $@ ¥
        $(LRTEMASIM) $(RTDDLIB) $(LSRRT) $(RTSYSLIB)

$(FUNC_PTR_SRC):;
        @echo $@ "Making..."
        @perl /prj/sr/em/frame/build/$(PRECOMP_PERL) ¥
        $(DEV_CONFIG) $(EQUIPLIB) ema

clean:; @rm -f $(OBJS) $(FUNC_PTR_SRC) $(EXEFILE) core
```

```
# Makefile for tellem & EMA
#          99-05-07  Hiramatsu  created
#
# Compiler parameter
EQUIPLIB     = id23
EQUIP        = ID23
INCPATH      = -I/prj/sr/include -I../../include -I/prj/bl/include
LIBPATH      = -L/prj/sr/lib     -L../../lib     -L/prj/bl/lib
LIBPATHRT    = -L/HP-RT/lib   -L/HP-RT/usr/lib
LIBPATHDEB   = -L/prj/sr/em/frame/lib
TEMPLATEPATH = /prj/sr/em/ema/src
RTDDLIB      = -lrt_devapi_rio    -lrt_drvlib_rio     -lrt_devapi_di     ¥
               -lrt_devapi_do     -lrt_devapi_ttldi   -lrt_devapi_ttldio ¥
               -lrt_drvlib_ttldio -lrt_devapi_ptg0350 -lrt_devapi_adc311 ¥
               -lrt_devapi_dio_himv602a  -lrt_devapi_com
LSRRT        = -lrt_cp  -lrt_util
RTSYSLIB     = -lp -lm -lc -lrpc -lbsd
LRTREAL      = -lrt_em_$(EQUIPLIB)      -lrt_em_frame      -lrt_em_idcom ¥
               -lrt_em_$(EQUIPLIB)      -lrt_em_frame_template
LRTREALDEB   = -lrt_em_$(EQUIPLIB)_deb  -lrt_em_frame      -lrt_em_idcom ¥
               -lrt_em_$(EQUIPLIB)_deb  -lrt_em_frame_template
LRTSIM       = -lrt_em_$(EQUIPLIB)_sim  -lrt_em_frame_sim  -lrt_em_idcom ¥
               -lrt_em_$(EQUIPLIB)_sim  -lrt_em_frame_template
LRTEMA       = -lrt_em_$(EQUIPLIB)  -lrt_em_ema    -lrt_em_frame ¥
               -lrt_em_idcom  -lrt_em_$(EQUIPLIB)  -lrt_em_frame_template
LRTEMASIM    = -lrt_em_$(EQUIPLIB)_sim  -lrt_em_ema_sim    -lrt_em_frame_sim ¥
               -lrt_em_idcom  -lrt_em_$(EQUIPLIB)_sim  -lrt_em_frame_template
DFLAG        = -D$(EQUIP)
CCRT         = ccrt -w -Ae $(INCPATH) $(LIBPATH) $(LIBPATHRT) $(DFLAG)

# Executable file
#EXEFILE     = tellem  tellem_deb
#EXEFILE      = tellem_sim
EXEFILE      = ema_rt

# Source and Object file
SRCS         = *.c
OBJS         = *.o

# Precompiler parameter
DEV_CONFIG   = config.tbl
FUNC_PTR_SRC = em_config_get_func_ptr.c
PRECOMP_PERL = pre_em_config_get_func_ptr.pl

# Target
all: $(FUNC_PTR_SRC) $(EXEFILE)

tellem:;
        @echo $@ "Making..."
        $(CCRT) $(SRCS) -o $@          $(LRTEMA) ¥
        $(LRTREAL) $(RTDDLIB) $(LSRRT) $(RTSYSLIB)

tellem_deb:;
        @echo $@ "Making..."
        $(CCRT) $(SRCS) -o $@  $(LRTEMA) ¥
        $(LIBPATHDEB) $(LRTREALDEB) $(RTDDLIB) $(LSRRT) $(RTSYSLIB)

tellem_sim:;
        @echo $@ "Making..."
        $(CCRT) $(SRCS) -o $@  $(LRTEMASIM) ¥
        $(LRTSIM) $(RTDDLIB) $(LSRRT) $(RTSYSLIB)

ema_rt:;
        @echo $@ "Making..."
        $(CCRT) $(SRCS) $(TEMPLATEPATH)/em_ema_main.c -o $@ ¥
        $(LRTEMASIM) $(RTDDLIB) $(LSRRT) $(RTSYSLIB)

$(FUNC_PTR_SRC):;
        @echo $@ "Making..."
        @perl /prj/sr/em/frame/build/$(PRECOMP_PERL) ¥
        $(DEV_CONFIG) $(EQUIPLIB) ema

clean:; @rm -f $(OBJS) $(FUNC_PTR_SRC) $(EXEFILE) core
```

```
#  Makefile for rpc_sim
#    99-06-11  Hiramatsu created

EQUIPLIB  = id23
INCPATH   = -I/prj/sr/include  -I../../include -I/prj/bl/include
LIBPATH   = -L/prj/sr/lib      -L../../lib     -L/prj/bl/lib
LIBPATHRT = -L/HP-RT/lib       -L/HP-RT/usr/lib
LIBPATHDD = -L/prj/sr/dev/lib
RTDDLIB   = -lrt_devapi_rio  -lrt_drvlib_rio    -lrt_devapi_di          ¥
            -lrt_devapi_do   -lrt_devapi_ttldi  -lrt_devapi_ttldio      ¥
            -lrt_drvlib_ttldio  -lrt_devapi_ptg0350  -lrt_devapi_adc311 ¥
            -lrt_devapi_dio_himv602a  -lrt_devapi_com
LSRRT     = -lrt_cp  -lrt_util
RTSYSLIB  = -lp -lm -lc -lrpc -lbsd
LRTSIM    = -lrt_em_$(EQUIPLIB)_sim  -lrt_em_frame_sim  -lrt_em_idcom  ¥
            -lrt_em_$(EQUIPLIB)_sim  -lrt_em_frame_template
LRTEMASIM = -lrt_em_$(EQUIPLIB)_sim  -lrt_em_ema_sim  -lrt_em_frame_sim  ¥
            -lrt_em_idcom  -lrt_em_$(EQUIPLIB)_sim  -lrt_em_frame_template
MAKE      = make
CCRT      = ccrt -Ae -w $(INCPATH) $(LIBPATH) $(LIBPATHRT) $(LIBPATHDD)
EXEFILE   = rpc_sim
SRCS      = *.c
OBJS      = *.o

DEV_CONFIG   = config.tbl
FUNC_PTR_SRC = em_config_get_func_ptr.c
PRECOMP_PERL = pre_em_config_get_func_ptr.pl

all: $(FUNC_PTR_SRC) $(EXEFILE)

rpc_sim:; @echo $@ "Making..."
        $(MAKE) rpc_prepare
        $(MAKE) cc_em_rpc_svc  COMPILER=ccrt
        $(CCRT) $(SRCS)  -o  $@  em_rpc_svc.o   $(LRTEMASIM) ¥
        $(LRTSIM) $(RTDDLIB) $(LSRRT) $(RTSYSLIB)

$(FUNC_PTR_SRC):;  @echo $@ "Making..."
        @perl /prj/sr/em/frame/build/$(PRECOMP_PERL) ¥
        $(DEV_CONFIG) $(EQUIPLIB) ema

clean:; @rm -f $(OBJS) $(FUNC_PTR_SRC) $(EXEFILE) core

rpc_prepare: ;
        @echo "Prepare RPC..."
        cp  /prj/sr/em/api/src/em_rpc.x  em_rpc.x
        rpcgen  em_rpc.x
        rm  em_rpc_clnt.c

cc_em_rpc_svc:;
        @echo "Compile em_rpc_svc..."
        $(COMPILER) $(INCPATH) -c  em_rpc_svc.c
        rm  em_rpc_svc.c
```

Appendix G: Makefile for Local Test (tellem and ema_rt)

```
# Library Makefile for ID23
#         97-12-10 Hiramatsu  created
#         99-04-03 Hiramatsu, for simulator
#         99-05-07 Hiramatsu, for EMA
#
# Compiler parameter
INC             = -I/HP-RT/usr/include -I/prj/sr/include  ¥
                  -I../../include       -I/prj/bl/include
DEBFLAG         = -g -DEM_DEBUG_PRINTF
SIMFLAG         = -DEM_DEBUG_SIM
TELLEMFLAG      = -DEM_DEBUG_TELLEM
CCRT            = ccrt  -w -Ae
MAKE            = make

# Library parameter
LIBNAME         = em_id23
TARGETDIR       = ../../lib

# Source and Objects File
SRCS            = *.c
OBJS            = *.o
RMOBJS          = em_poller_init.o em_poller_terminate.o

# Target Rule
all : librt_$(LIBNAME).a
#all :  librt_$(LIBNAME)_sim.a

librt_$(LIBNAME).a: $(SRCS)
                @echo $@ " making start"
                @rm -f *.o
                $(CCRT) $(TELLEMFLAG) -c $(INC) $(SRCS)
                $(MAKE) lib NAME=$@
                @echo $@ " making end"

librt_$(LIBNAME)_deb.a: $(SRCS)
                @echo $@ " making start"
                @rm -f *.o
                $(CCRT) $(TELLEMFLAG) $(DEBFLAG) -c $(INC) $(SRCS)
                $(MAKE) lib NAME=$@
                @echo $@ " making end"

librt_$(LIBNAME)_sim.a: $(SRCS)
                @echo $@ " making start"
                @rm -f *.o
                $(CCRT) -c $(SIMFLAG) $(INC) $(SRCS)
                $(MAKE) lib NAME=$@
                @echo $@ " making end"

lib:;
                @rm -f $(RMOBJS)
                @rm -f $(TARGETDIR)/$(NAME)
                ar cru $(TARGETDIR)/$(NAME) $(OBJS)
                @rm -f $(OBJS)

clean:;         @rm -f $(OBJS) core
```

```
# Makefile for tellem & EMA
#          99-05-07  Hiramatsu  created
#
# Compiler parameter
EQUIPLIB     = id23
EQUIP        = ID23
INCPATH      = -I/prj/sr/include -I../../include -I/prj/bl/include
LIBPATH      = -L/prj/sr/lib     -L../../lib     -L/prj/bl/lib
LIBPATHRT    = -L/HP-RT/lib   -L/HP-RT/usr/lib
LIBPATHDEB   = -L/prj/sr/em/frame/lib
TEMPLATEPATH = /prj/sr/em/ema/src
DEBFLAG      = -g -DEM_DEBUG_PRINTF
SIMFLAG      = -DEM_DEBUG_SIM
RTDDLIB      = -lrt_devapi_rio    -lrt_drvlib_rio     -lrt_devapi_di     ¥
               -lrt_devapi_do     -lrt_devapi_ttldi   -lrt_devapi_ttldio ¥
               -lrt_drvlib_ttldio -lrt_devapi_ptg0350 -lrt_devapi_adc311 ¥
               -lrt_devapi_dio_himv602a  -lrt_devapi_com
LSRRT        = -lrt_cp  -lrt_util
RTSYSLIB     = -lp -lm -lc -lrpc -lbsd
LRTREAL      = -lrt_em_$(EQUIPLIB)       -lrt_em_frame      -lrt_em_idcom ¥
               -lrt_em_$(EQUIPLIB)       -lrt_em_frame_template
LRTREALDEB   = -lrt_em_$(EQUIPLIB)_deb  -lrt_em_frame      -lrt_em_idcom ¥
               -lrt_em_$(EQUIPLIB)_deb  -lrt_em_frame_template
LRTSIM       = -lrt_em_$(EQUIPLIB)_sim  -lrt_em_frame_sim  -lrt_em_idcom ¥
               -lrt_em_$(EQUIPLIB)_sim  -lrt_em_frame_template
LRTEMA       = -lrt_em_$(EQUIPLIB)  -lrt_em_ema    -lrt_em_frame ¥
               -lrt_em_idcom  -lrt_em_$(EQUIPLIB)  -lrt_em_frame_template
LRTEMASIM    = -lrt_em_$(EQUIPLIB)_sim  -lrt_em_ema_sim    -lrt_em_frame_sim ¥
               -lrt_em_idcom  -lrt_em_$(EQUIPLIB)_sim  -lrt_em_frame_template
DFLAG        = -D$(EQUIP) -DEM_LOG_SRAM=0
CCRT         = ccrt -w -Ae $(INCPATH) $(LIBPATH) $(LIBPATHRT) $(DFLAG)

# Executable file
EXEFILE     = tellem
#EXEFILE      = tellem_sim
#EXEFILE      = ema_rt

# Source and Object file
SRCS         = *.c
OBJS         = *.o

# Precompiler parameter
DEV_CONFIG   = config.tbl
FUNC_PTR_SRC = em_config_get_func_ptr.c
PRECOMP_PERL = pre_em_config_get_func_ptr.pl

# Target
all: $(FUNC_PTR_SRC) $(EXEFILE)

tellem:;
        @echo $@ "Making..."
        $(CCRT) $(SRCS) -o $@          $(LRTEMA) ¥
        $(LRTREAL) $(RTDDLIB) $(LSRRT) $(RTSYSLIB)

tellem_deb:;
        @echo $@ "Making..."
        $(CCRT) $(DEBFLAG) $(SRCS) -o $@  $(LRTEMA) ¥
        $(LIBPATHDEB) $(LRTREALDEB) $(RTDDLIB) $(LSRRT) $(RTSYSLIB)

tellem_sim:;
        @echo $@ "Making..."
        $(CCRT) $(SIMFLAG) $(DEBFLAG) $(SRCS) -o $@  $(LRTEMASIM) ¥
        $(LRTSIM) $(RTDDLIB) $(LSRRT) $(RTSYSLIB)

ema_rt:;
        @echo $@ "Making..."
        $(CCRT) $(SRCS) $(TEMPLATEPATH)/em_ema_main.c -o $@ ¥
        $(LRTEMASIM) $(RTDDLIB) $(LSRRT) $(RTSYSLIB)

$(FUNC_PTR_SRC):;
        @echo $@ "Making..."
        @perl /prj/sr/em/frame/build/$(PRECOMP_PERL) ¥
        $(DEV_CONFIG) $(EQUIPLIB) ema

clean:; @rm -f $(OBJS) $(FUNC_PTR_SRC) $(EXEFILE) core
```

```
# Makefile for tellem & EMA
#          99-05-07  Hiramatsu  created
#
# Compiler parameter
EQUIPLIB     = id23
EQUIP        = ID23
INCPATH      = -I/prj/sr/include -I../../include -I/prj/bl/include
LIBPATH      = -L/prj/sr/lib     -L../../lib     -L/prj/bl/lib
LIBPATHRT    = -L/HP-RT/lib   -L/HP-RT/usr/lib
LIBPATHDEB   = -L/prj/sr/em/frame/lib
TEMPLATEPATH = /prj/sr/em/ema/src
DEBFLAG      = -g -DEM_DEBUG_PRINTF
SIMFLAG      = -DEM_DEBUG_SIM
RTDDLIB      = -lrt_devapi_rio    -lrt_drvlib_rio     -lrt_devapi_di     ¥
               -lrt_devapi_do     -lrt_devapi_ttldi   -lrt_devapi_ttldio ¥
               -lrt_drvlib_ttldio -lrt_devapi_ptg0350 -lrt_devapi_adc311 ¥
               -lrt_devapi_dio_himv602a  -lrt_devapi_com
LSRRT        = -lrt_cp  -lrt_util
RTSYSLIB     = -lp -lm -lc -lrpc -lbsd
LRTREAL      = -lrt_em_$(EQUIPLIB)       -lrt_em_frame      -lrt_em_idcom ¥
               -lrt_em_$(EQUIPLIB)       -lrt_em_frame_template
LRTREALDEB   = -lrt_em_$(EQUIPLIB)_deb  -lrt_em_frame      -lrt_em_idcom ¥
               -lrt_em_$(EQUIPLIB)_deb  -lrt_em_frame_template
LRTSIM       = -lrt_em_$(EQUIPLIB)_sim  -lrt_em_frame_sim  -lrt_em_idcom ¥
               -lrt_em_$(EQUIPLIB)_sim  -lrt_em_frame_template
LRTEMA       = -lrt_em_$(EQUIPLIB)  -lrt_em_ema    -lrt_em_frame ¥
               -lrt_em_idcom  -lrt_em_$(EQUIPLIB)  -lrt_em_frame_template
LRTEMASIM    = -lrt_em_$(EQUIPLIB)_sim  -lrt_em_ema_sim    -lrt_em_frame_sim ¥
               -lrt_em_idcom  -lrt_em_$(EQUIPLIB)_sim  -lrt_em_frame_template
DFLAG        = -D$(EQUIP) -DEM_LOG_SRAM=0
CCRT         = ccrt -w -Ae $(INCPATH) $(LIBPATH) $(LIBPATHRT) $(DFLAG)

# Executable file
#EXEFILE     = tellem  tellem_deb
#EXEFILE      = tellem_sim
EXEFILE      = ema_rt

# Source and Object file
SRCS         = *.c
OBJS         = *.o

# Precompiler parameter
DEV_CONFIG   = config.tbl
FUNC_PTR_SRC = em_config_get_func_ptr.c
PRECOMP_PERL = pre_em_config_get_func_ptr.pl

# Target
all: $(FUNC_PTR_SRC) $(EXEFILE)

tellem:;
        @echo $@ "Making..."
        $(CCRT) $(SRCS) -o $@          $(LRTEMA) ¥
        $(LRTREAL) $(RTDDLIB) $(LSRRT) $(RTSYSLIB)

tellem_deb:;
        @echo $@ "Making..."
        $(CCRT) $(DEBFLAG) $(SRCS) -o $@  $(LRTEMA) ¥
        $(LIBPATHDEB) $(LRTREALDEB) $(RTDDLIB) $(LSRRT) $(RTSYSLIB)

tellem_sim:;
        @echo $@ "Making..."
        $(CCRT) $(SIMFLAG) $(DEBFLAG) $(SRCS) -o $@  $(LRTEMASIM) ¥
        $(LRTSIM) $(RTDDLIB) $(LSRRT) $(RTSYSLIB)

ema_rt:;
        @echo $@ "Making..."
        $(CCRT) $(SRCS) $(TEMPLATEPATH)/em_ema_main.c -o $@ ¥
        $(LRTEMA) $(RTDDLIB) $(LSRRT) $(RTSYSLIB)

$(FUNC_PTR_SRC):;
        @echo $@ "Making..."
        @perl /prj/sr/em/frame/build/$(PRECOMP_PERL) ¥
        $(DEV_CONFIG) $(EQUIPLIB) ema

clean:; @rm -f $(OBJS) $(FUNC_PTR_SRC) $(EXEFILE) core
```

Makefiles for EM(rpc_em_rt*), EMA(ema1_rt*) and Poller-collector(pc_po_main*)

```
#       EM executable Makefile
#       Created Taketani Feb-12-97
#       99-09-12   Hiramatsu   adapted for ID23

DEV_CONFIG = config.tbl
INCPATH    = -I./ -I/prj/sr/include
LIBPATH    = -L/prj/sr/lib
LIBPATHRT  = -L/HP-RT/lib -L/HP-RT/usr/lib
TEMPLATEPATH = /prj/sr/em/frame/build
UXSRLIB    = -lsr_cp -lsr_util
RTSRLIB    = -lrt_cp -lrt_util
UXSYSLIB   = -lm
RTSYSLIB   = -lp -lm -lc -lrpc -lbsd
RTDDLIB    = -lrt_devapi_com -lrt_devapi_rio -lrt_drvlib_rio ¥
           -lrt_devapi_di -lrt_devapi_do -lrt_devapi_ptg ¥
           -lrt_devapi_ai -lrt_devapi_ttldi -lrt_devapi_ttldio ¥
           -lrt_devapi_gpib -lrt_devapi_ai_advme2602 -lrt_devapi_dio_himv602a ¥
           -lrt_devapi_ai_ics130 -lrt_devapi_rpv100 ¥
           -lrt_drvlib_gpib -lrt_drvlib_ttldio ¥
           -lrt_devapi_com
MAKE       = make -f /HP-RT/prj/bin/build_cpu/Makefile.em
DFLAG      = -D$(EQUIP) -g
SIMFLAG    = -DEM_DEBUG_SIM
CCUX       = cc   -Ae -w $(INCPATH) $(SIMFLAG) $(DFLAG) $(LIBPATH)
CCRT       = ccrt -Ae -w $(INCPATH) $(DFLAG) $(LIBPATH) $(LIBPATHRT)
EXEFILE    =  tellem_rt rpc_em_rt
SRCS       = *.c /prj/sr/em/frame/src/em_log_open_sram.c
OBJS       = *.o
FUNC_PTR_SRC = em_config_get_func_ptr.c
PRECOMP_PERL = pre_em_config_get_func_ptr.pl
DEBFLAG    = -g
all: $(FUNC_PTR_SRC) $(EXEFILE)

#RPC EM (controlled through AS)
rpc_em_rt::
        @echo $@ "Making..."
        @rm -f tellem.c
        $(MAKE) rpc_prepare
        $(MAKE) cc_em_rpc_svc COMPILER=ccrt
        $(CCRT) -DEM_LOG_SRAM=0 $(SRCS) -o $@ em_rpc_svc.o    ¥
        -lrt_em_$(EQUIP) -lrt_em_gpib -lrt_em_ai -lrt_em_ema  ¥
        -lrt_em_frame -lrt_em_$(EQUIP) -lrt_em_frame_template ¥
        $(RTDDLIB) $(RTSRLIB) $(RTSYSLIB)

#tellem (locally and interactive)
tellem_rt::
        @echo $@ "Makinfg..."
        $(MAKE) rpc_prepare_tellem
        $(CCRT)  -DEM_LOG_SRAM=0 $(SRCS) $(TEMPLATEPATH)/tellem.c -o $@ ¥
        -lrt_em_$(EQUIP) -lrt_em_gpib -lrt_em_ai -lrt_em_ema  ¥
        -lrt_em_frame -lrt_em_$(EQUIP) -lrt_em_frame_template ¥
        $(RTDDLIB) $(RTSRLIB) $(RTSYSLIB)

#RPC EM HP-RT simulator (controlled through AS)
rpc_em_rt_sim::
        @echo $@ "Making..."
        @rm -f tellem.c
        $(MAKE) rpc_prepare
        $(MAKE) cc_em_rpc_svc COMPILER=ccrt
        $(CCRT) $(SIMFLAG) $(SRCS) -o $@ em_rpc_svc.o  ¥
        -lrt_em_$(EQUIP)_sim -lrt_em_gpib_sim -lrt_em_ai_sim -lrt_em_ema_sim ¥
        -lrt_em_frame_sim -lrt_em_$(EQUIP)_sim -lrt_em_frame_template ¥
        $(RTSRLIB) $(RTSYSLIB)

#tellem HP-RT simulator(locally and interactive)
tellem_rt_sim::
        @echo $@ "Making..."
        $(MAKE) rpc_prepare_tellem
        $(CCRT) $(SIMFLAG) $(SRCS) $(TEMPLATEPATH)/tellem.c -o $@ ¥
```

```
        -lrt_em_$(EQUIP)_sim -lrt_em_gpib_sim -lrt_em_ai_sim -lrt_em_ema_sim ¥
        -lrt_em_frame_sim -lrt_em_$(EQUIP)_sim -lrt_em_frame_template ¥
        $(RTSRLIB) $(RTSYSLIB)

#RPC EM HP-UX simulator (controled through AS)
rpc_em_ux_sim:;
        @echo $@ "Making..."
        @rm -f tellem.c
        $(MAKE) rpc_prepare
        $(MAKE) cc_em_rpc_svc COMPILER=cc
        $(CCUX) $(SRCS) -o $@ em_rpc_svc.o  ¥
        -lsr_em_$(EQUIP)_sim -lsr_em_ema_sim ¥
        -lsr_em_frame_sim -lsr_em_$(EQUIP)_sim -lsr_em_frame_template ¥
        $(UXSRLIB) $(UXSYSLIB)

#tellem HP-UX simulator(locally and interactive)
tellem_ux_sim:;
        @echo $@ "Making..."
        $(MAKE) rpc_prepare_tellem
        $(CCUX) $(SRCS) $(TEMPLATEPATH)/tellem.c -o $@ ¥
        -lsr_em_$(EQUIP)_sim -lsr_em_ema_sim ¥
        -lsr_em_frame_sim -lsr_em_$(EQUIP)_sim -lsr_em_frame_template ¥
        $(UXSRLIB) $(UXSYSLIB)

$(FUNC_PTR_SRC):;
        @echo $@ "Making..."
        @perl /prj/sr/em/frame/build/$(PRECOMP_PERL) ¥
        $(DEV_CONFIG) $(EQUIP) ema gpib ai

clean:;
        @rm -f $(OBJS) $(FUNC_PTR_SRC) $(EXEFILE) core em_rpc.h ¥
        em_rpc_clnt.c em_rpc_svc.c em_rpc_xdr.c em_rpc.x

cc_em_rpc_svc:;
        @echo "Compile em_rpc_svc..."
        $(COMPILER) $(INCPATH) $(DFLAG) -c em_rpc_svc.c
        rm em_rpc_svc.c em_rpc.x

rpc_prepare: ;
        @echo "Prepare RPC..."
        cp /prj/sr/em/api/src/em_rpc.x em_rpc.x
        rpcgen em_rpc.x
        rm em_rpc_clnt.c

rpc_prepare_tellem: ;
        @echo "Prepare RPC..."
        cp /prj/sr/em/api/src/em_rpc.x em_rpc.x
        rpcgen em_rpc.x
        rm em_rpc_clnt.c em_rpc_svc.c em_rpc_xdr.c em_rpc.x
```

```
#         EM executable Makefile
#         Created Taketani Feb-12-97
#         99-09-12  Hiramatsu  adapted for ID23

DEV_CONFIG = config.tbl
INCPATH    = -I./ -I/prj/sr/include
LIBPATH    = -L/prj/sr/lib
LIBPATHRT  = -L/HP-RT/lib -L/HP-RT/usr/lib
TEMPLATEPATH = /prj/sr/em/ema/src
UXSRLIB    = -lsr_cp -lsr_util
RTSRLIB    = -lrt_cp -lrt_util
UXSYSLIB   = -lm
RTSYSLIB   = -lp -lm -lc -lrpc -lbsd
RTDDLIB    = -lrt_devapi_com -lrt_devapi_rio -lrt_drvlib_rio ¥
           -lrt_devapi_di -lrt_devapi_do -lrt_devapi_ptg ¥
           -lrt_devapi_ai -lrt_devapi_ttldi -lrt_devapi_ttldio ¥
           -lrt_devapi_gpib -lrt_devapi_ai_advme2602 -lrt_devapi_dio_himv602a ¥
           -lrt_devapi_rpv100 ¥
           -lrt_drvlib_gpib -lrt_drvlib_ttldio ¥
           -lrt_devapi_com
MAKE       = make -f /HP-RT/prj/bin/build_cpu/Makefile.ema
DFLAG      = -D$(EQUIP) -DEM_DEBUG_RF_TMP_TTL
SIMFLAG    = -DEM_DEBUG_SIM
CCUX       = cc   -Ae -w $(INCPATH) $(SIMFLAG) $(DFLAG) $(LIBPATH)
CCRT       = ccrt -Ae -w $(INCPATH) $(DFLAG) $(LIBPATH) $(LIBPATHRT)
EXEFILE    = ema1_rt
SRCS       = *.c /prj/sr/em/frame/src/em_log_open_sram.c
OBJS       = *.o
FUNC_PTR_SRC = em_config_get_func_ptr.c
PRECOMP_PERL = pre_em_config_get_func_ptr.pl
DEBFLAG    = -g
all: $(FUNC_PTR_SRC) $(EXEFILE)

#tellem (locally and interactive)
ema1_rt:;
        @echo $@ "Making..."
        $(MAKE) rpc_prepare_tellem
        $(CCRT) -DEM_LOG_SRAM=2 $(SRCS) $(TEMPLATEPATH)/em_ema_main.c -o $@ ¥
        -lrt_em_$(EQUIP) -lrt_em_gpib -lrt_em_ai -lrt_em_ema ¥
        -lrt_em_frame -lrt_em_$(EQUIP) -lrt_em_frame_template ¥
        $(RTDDLIB) $(RTSRLIB) $(RTSYSLIB)
        @cp -f $(EXEFILE) ../em

$(FUNC_PTR_SRC):;
        @echo $@ "Making..."
        $(MAKE) rpc_prepare_tellem
        @perl /prj/sr/em/frame/build/$(PRECOMP_PERL) $(DEV_CONFIG) $(EQUIP) ema gpib ai

clean:;
        @rm -f $(OBJS) $(FUNC_PTR_SRC) $(EXEFILE) em_rpc_clnt.c
        em_rpc_svc.c em_rpc_xdr.c em_rpc.x em_rpc.h

rpc_prepare_tellem: ;
        @echo "Prepare RPC..."
        cp /prj/sr/em/api/src/em_rpc.x em_rpc.x
        rpcgen em_rpc.x
        rm em_rpc_clnt.c em_rpc_svc.c em_rpc_xdr.c em_rpc.x
```

```
#  Makefile for Poller-collector
#  99-09-12  Hiramatsu  adapted for ID23

DEV_CONFIG = config.tbl
INCPATH    = -I./ -I/prj/sr/include -I/HP-RT/usr/include -I/usr/include/sys
LIBPATH    = -L/prj/sr/lib -L/HP-RT/lib -L/HP-RT/usr/lib
DEBUG      = -DDEBUG
CFLAGS     = ${DEBUG}
CC         = /HP-RT/hpux/bin/ccrt -Ae ${INCPATH} -g
CCRPC      = /HP-RT/hpux/bin/ccrt -Ae -c ${INCPATH} -g
LIB        = -lrtpcpol -lrtpccom -lrt_util
CONFSRC    = em_config_get_func_ptr.c
CONFOBJ    = em_config_get_func_ptr.o
LRTREAL    = -lrt_em_$(EQUIP) -lrt_em_gpib_sim -lrt_em_ai_sim -lrt_em_ema ¥
             -lrt_em_frame_poller -lrt_em_$(EQUIP) ¥
             -lrt_em_frame_template
RTDDLIB    = -lrt_devapi_com -lrt_devapi_rio -lrt_drvlib_rio     ¥
             -lrt_devapi_di -lrt_devapi_do -lrt_devapi_ptg       ¥
             -lrt_devapi_ai -lrt_devapi_ttldi -lrt_devapi_ttldio ¥
             -lrt_devapi_gpib -lrt_devapi_ai_advme2602           ¥
             -lrt_devapi_dio_himv602a -lrt_devapi_rpv100         ¥
             -lrt_drvlib_gpib -lrt_drvlib_ttldio -lrt_devapi_com
LSRRT      = -lrt_cp -lrt_util
RTSYSLIB   = -lp -lm -lc -lbsd
EMLIB      = ${LRTREAL} ${RTDDLIB} ${LSRRT} ${RTSYSLIB}

PRECOMP_PERL = pre_em_config_get_func_ptr.pl

# Poller main
MOD     = pc_po_main
SRC_FIL = /prj/sr/polcol/pol/src/pc_po_main.c ¥
          /prj/sr/em/frame/src/em_log_open_sram.c

all     : ${CONFSRC} em_rpc.h ${MOD}

${CONFSRC} : ${DEV_CONFIG}
             @echo $@ "Making..."
             @perl /prj/sr/em/frame/build/${PRECOMP_PERL} ¥
             ${DEV_CONFIG} ${EQUIP} ema gpib ai

pc_po_main : ${SRC_FIL} ${CONFOBJ}
             ${CC} -o $@ -DEM_LOG_SRAM=${SRAM_NO} ${CFLAGS}    ¥
             ${SRC_FIL} ${CONFSRC} ${LIBPATH} ${LIB} ${EMLIB} ¥
             rm -f *.o

${CONFOBJ} : ${CONFSRC}
             ${CCRPC} ${CONFSRC} ${CFLAGS}

clean :;
        rm -f *.o ${MOD} ${CONFSRC} em_rpc.h em_rpc_clnt.c ¥
        em_rpc_svc.c em_rpc_xdr.c em_rpc.x

em_rpc.h:
        @echo "Prepare RPC..."
        cp /prj/sr/em/api/src/em_rpc.x em_rpc.x
        rpcgen em_rpc.x
        rm em_rpc_clnt.c em_rpc_svc.c em_rpc_xdr.c em_rpc.x
```

Appendix H: Header Files

```
/*(@header_begin)*****************************************************
 *  (C Header)                                                       *
 *      Name = em_id23                                               *
 *  (Purpose)                                                        *
 *     Header file for ID23                                          *
 *  (Relation)                                                       *
 *                                                                   *
 *  (History)                                                        *
 *      18-Nov-1996      T.Fukui/SPring-8  for id23                  *
 *      97-Dec-17        Hiramatsu         revised                   *
 *****************************************************(@header_end)*/
#ifndef BL_INCLUDE_EM_ID23
  #define BL_INCLUDE_EM_ID23

#include </HP-RT/usr/include/sys/types.h>
#include </HP-RT/usr/include/stat.h>
#include </HP-RT/usr/include/sys/shmmap.h>
#include </HP-RT/usr/include/fcntl.h>
#include </HP-RT/usr/include/string.h>
#include </HP-RT/usr/include/stdlib.h>
#include </HP-RT/usr/include/stdio.h>
#include </HP-RT/usr/include/rpc/rpc.h>

#include "sr_all.h"
#include "sr_em.h"
#include "dev_api.h"
#include "em_idcom.h"

#include "em_id23_define.h"
#include "em_id23_typedef.h"
#include "em_id23_func.h"
#include "em_id23_err_define.h"

#endif
```

```
 1 /*(@header_begin)*************************************************
 2 *  (C Header)                                                    *
 3 *      Name = em_id23_define                                     *
 4 *  (Purpose)                                                     *
 5 *       Macro constants are defined                              *
 6 *  (Relation)                                                    *
 7 *                                                                *
 8 *  (History)                                                     *
 9 *      Jul-04-1996  H.Funayama/iST     created                   *
10 *      98-01-16     Hiramatsu          revised                   *
11 *      99-01-15     Hiramatsu          updated                   *
12 *      99-04-23     Hiramatsu          updated                   *
13 *      99-07-05     Hiramatsu          join lc_ and st_curr_put  *
14 *************************************************(@header_end)*/
15 #ifndef    EM_ID23_DEFINE
16   #define  EM_ID23_DEFINE
17
18 #define EM_ID23_BCD_1                          0x000F
19 #define EM_ID23_BCD_2                          0x00F0
20 #define EM_ID23_BCD_3                          0x0F00
21 #define EM_ID23_BCD_4                          0xF000
22 #define EM_ID23_EMERGENCY_CNT_1_BIT            0x0001
23 #define EM_ID23_EMERGENCY_CNT_1_SET_BIT        0x0001
24 #define EM_ID23_EMERGENCY_CNT_2_BIT            0x0002
25 #define EM_ID23_EMERGENCY_CNT_2_SET_BIT        0x0002
26 #define EM_ID23_EMERGENCY_INPUT_PORT_1             1
27 #define EM_ID23_EMERGENCY_INPUT_PORT_2             2
28 #define EM_ID23_EMERGENCY_LOCAL_BIT            0x0008
29 #define EM_ID23_EMERGENCY_LOCAL_SET_BIT        0x0020
30 #define EM_ID23_EMERGENCY_PB_1_BIT             0x0001
31 #define EM_ID23_EMERGENCY_PB_1_SET_BIT         0x0004
32 #define EM_ID23_EMERGENCY_PB_2_BIT             0x0002
33 #define EM_ID23_EMERGENCY_PB_2_SET_BIT         0x0008
34 #define EM_ID23_EMERGENCY_PB_3_BIT             0x0004
35 #define EM_ID23_EMERGENCY_PB_3_SET_BIT         0x0010
36 #define EM_ID23_GAP_CONV_MOVE_DATA                 0
37 #define EM_ID23_GAP_FULL_OPEN                  300.0
38 #define EM_ID23_GAP_INIT_DELAY_TIME                1
39 #define EM_ID23_GAP_INIT_INPUT_PORT                3
40 #define EM_ID23_GAP_LIN                            1
41 #define EM_ID23_PUT_POSITION                       1
42 #define EM_ID23_GET_POSITION                       0
43 #define EM_ID23_GAP_LMT_HOME_BIT               0x0040
44 #define EM_ID23_GAP_LMT_HOME_SET_BIT           0x0100
45 #define EM_ID23_GAP_LMT_INPUT_PORT_1               1
46 #define EM_ID23_GAP_LMT_INPUT_PORT_2               2
47 #define EM_ID23_GAP_LMT_LW_DW_CL_BIT           0x0080
48 #define EM_ID23_GAP_LMT_LW_DW_CL_SET_BIT       0x0080
49 #define EM_ID23_GAP_LMT_LW_DW_OP_BIT           0x0008
50 #define EM_ID23_GAP_LMT_LW_DW_OP_SET_BIT       0x0040
51 #define EM_ID23_GAP_LMT_LW_UP_CL_BIT           0x0040
52 #define EM_ID23_GAP_LMT_LW_UP_CL_SET_BIT       0x0020
53 #define EM_ID23_GAP_LMT_LW_UP_OP_BIT           0x0004
54 #define EM_ID23_GAP_LMT_LW_UP_OP_SET_BIT       0x0010
55 #define EM_ID23_GAP_LMT_UP_DW_CL_BIT           0x0020
56 #define EM_ID23_GAP_LMT_UP_DW_CL_SET_BIT       0x0008
57 #define EM_ID23_GAP_LMT_UP_DW_OP_BIT           0x0002
58 #define EM_ID23_GAP_LMT_UP_DW_OP_SET_BIT       0x0004
59 #define EM_ID23_GAP_LMT_UP_UP_CL_BIT           0x0010
60 #define EM_ID23_GAP_LMT_UP_UP_CL_SET_BIT       0x0002
61 #define EM_ID23_GAP_LMT_UP_UP_OP_BIT           0x0001
62 #define EM_ID23_GAP_LMT_UP_UP_OP_SET_BIT       0x0001
63 #define EM_ID23_GAP_MAX_NUM                    319.9
```

```
64 #define EM_ID23_GAP_MIN_NUM                          34.1
65 #define EM_ID23_GAP_MOVE_ACC_CYCLE                      3
66 #define EM_ID23_GAP_MOVE_CONV_COEFF                5000.0
67 #define EM_ID23_GAP_MOVE_V_MAX                          1
68 #define EM_ID23_GAP_MOVE_V_MIN                          2
69 #define EM_ID23_GAP_MOVE_WTIME                          4
70 #define EM_ID23_GAP_OPEN_ACC_CYCLE                      3
71 #define EM_ID23_GAP_OPEN_V_MAX                          1
72 #define EM_ID23_GAP_OPEN_V_MIN                          2
73 #define EM_ID23_GAP_PRESET_DELAY_TIME                   3
74 #define EM_ID23_GAP_PRESET_LIN                          1
75 #define EM_ID23_GAP_PRESET_ROT                          2
76 #define EM_ID23_GAP_ROT                                 0
77 #define EM_ID23_GAP_STATUS_INPUT_BIT                    1
78 #define EM_ID23_GET_CURR                                0
79 #define EM_ID23_GET_TABLE                               0
80 #define EM_ID23_I_L_GAP_LMT_STATUS                   0xff
81 #define EM_ID23_I_L_GAP_STATUS                       0x01
82 #define EM_ID23_I_L_PHASE_STATUS                     0x231
83 #define EM_ID23_I_L_SIP_STATUS                       0x01
84 #define EM_ID23_LC_CURR_EXEC_CONV_VALUE_1               0
85 #define EM_ID23_LC_CURR_EXEC_CONV_VALUE_2               1
86 #define EM_ID23_LC_CURR_EXEC_KIND                       2
87 #define EM_ID23_LC_CURR_EXEC_MANUAL                     0
88 #define EM_ID23_LC_CURR_EXEC_TABLE                      1
89 #define EM_ID23_LC_CURR_PUT_KIND                        3
90 #define EM_ID23_LC_CURR_PUT_MANUAL                      0
91 #define EM_ID23_LC_CURR_PUT_TABLE                       1
92 #define EM_ID23_LC_TABLE_SET_FILE_MAX                 256
93 #define EM_ID23_LC_TABLE_SET_FILE_NAME                  0
94 #define EM_ID23_LC_TABLE_THRU_FILE_MAX                256
95 #define EM_ID23_LC_TABLE_THRU_FILE_NAME                 0
96 #define EM_ID23_PHASE_BRAKE_DELAY_TIME                  1
97 #define EM_ID23_PHASE_BRAKE_L_AMP_BIT                   5
98 #define EM_ID23_PHASE_BRAKE_L_ON_BIT                    3
99 #define EM_ID23_PHASE_BRAKE_U_AMP_BIT                   4
100 #define EM_ID23_PHASE_BRAKE_U_ON_BIT                    2
101 #define EM_ID23_PHASE_CONV_MOVE_BOTH                    0
102 #define EM_ID23_PHASE_CONV_MOVE_DATA                    0
103 #define EM_ID23_PHASE_CONV_MOVE_KIND                    0
104 #define EM_ID23_PHASE_CONV_MOVE_LOWER                   2
105 #define EM_ID23_PHASE_CONV_MOVE_UPPER                   1
106 #define EM_ID23_PHASE_INIT_DELAY_TIME                   2
107 #define EM_ID23_PHASE_INIT_INPUT_PORT                   4
108 #define EM_ID23_PHASE_INIT_KIND                         1
109 #define EM_ID23_PHASE_LMT_DW_1_BIT                 0x0200
110 #define EM_ID23_PHASE_LMT_DW_1_SET_BIT             0x0002
111 #define EM_ID23_PHASE_LMT_DW_2_BIT                 0x0800
112 #define EM_ID23_PHASE_LMT_DW_2_SET_BIT             0x0008
113 #define EM_ID23_PHASE_LMT_DW_3_BIT                 0x2000
114 #define EM_ID23_PHASE_LMT_DW_3_SET_BIT             0x0020
115 #define EM_ID23_PHASE_LMT_DW_4_BIT                 0x8000
116 #define EM_ID23_PHASE_LMT_DW_4_SET_BIT             0x0080
117 #define EM_ID23_PHASE_LMT_INPUT_PORT_1                  1
118 #define EM_ID23_PHASE_LMT_INPUT_PORT_2                  2
119 #define EM_ID23_PHASE_LMT_LW_HOME_BIT              0x0100
120 #define EM_ID23_PHASE_LMT_LW_HOME_SET_BIT          0x0200
121 #define EM_ID23_PHASE_LMT_UP_1_BIT                 0x0100
122 #define EM_ID23_PHASE_LMT_UP_1_SET_BIT             0x0001
123 #define EM_ID23_PHASE_LMT_UP_2_BIT                 0x0400
124 #define EM_ID23_PHASE_LMT_UP_2_SET_BIT             0x0004
125 #define EM_ID23_PHASE_LMT_UP_3_BIT                 0x1000
126 #define EM_ID23_PHASE_LMT_UP_3_SET_BIT             0x0010
```

```
127 #define EM_ID23_PHASE_LMT_UP_4_BIT              0x4000
128 #define EM_ID23_PHASE_LMT_UP_4_SET_BIT          0x0040
129 #define EM_ID23_PHASE_LMT_UP_HOME_BIT           0x0080
130 #define EM_ID23_PHASE_LMT_UP_HOME_SET_BIT       0x0100
131 #define EM_ID23_PHASE_LOWER                          1
132 #define EM_ID23_PHASE_L_LIN                          5
133 #define EM_ID23_PHASE_L_ROT                          3
134 #define EM_ID23_PHASE_L_SIGN_BIT_ROT            0x8000
135 #define EM_ID23_PHASE_MAX_NUM                    122.0
136 #define EM_ID23_PHASE_MIN_NUM                   -122.0
137 #define EM_ID23_PHASE_MOVE_ACC_CYCLE                 3
138 #define EM_ID23_PHASE_MOVE_CONV_COEFF            200.0
139 #define EM_ID23_PHASE_MOVE_L_PULS_NUM                1
140 #define EM_ID23_PHASE_MOVE_ONE_SIDE_ACC_CYCLE        3
141 #define EM_ID23_PHASE_MOVE_ONE_SIDE_PHASE_SIDE       4
142 #define EM_ID23_PHASE_MOVE_ONE_SIDE_PHASE_UPPER      0
143 #define EM_ID23_PHASE_MOVE_ONE_SIDE_V_MAX            1
144 #define EM_ID23_PHASE_MOVE_ONE_SIDE_V_MIN            2
145 #define EM_ID23_PHASE_MOVE_ONE_SIDE_WTIME            5
146 #define EM_ID23_PHASE_MOVE_U_PULS_NUM                0
147 #define EM_ID23_PHASE_MOVE_V_MAX                     1
148 #define EM_ID23_PHASE_MOVE_V_MIN                     2
149 #define EM_ID23_PHASE_MOVE_WTIME                     4
150 #define EM_ID23_PHASE_PRESET_DELAY_TIME              3
151 #define EM_ID23_PHASE_PRESET_LIN                     1
152 #define EM_ID23_PHASE_PRESET_ROT                     2
153 #define EM_ID23_PHASE_RESET_DELAY_TIME               3
154 #define EM_ID23_PHASE_RESET_OUT_BIT_1                1
155 #define EM_ID23_PHASE_RESET_OUT_BIT_2                2
156 #define EM_ID23_PHASE_STATUS_INPUT_PORT              1
157 #define EM_ID23_PHASE_TRIGGER_BIT_NUM                1
158 #define EM_ID23_PHASE_UPPER                          0
159 #define EM_ID23_PHASE_U_LIN                          4
160 #define EM_ID23_PHASE_U_ROT                          2
161 #define EM_ID23_PHASE_U_SIGN_BIT_ROT            0x4000
162 #define EM_ID23_POSITION_CONV_KIND                   0
163 #define EM_ID23_PUT_CURR                             1
164 #define EM_ID23_PUT_TABLE                            1
165 #define EM_ID23_RFBPM_CONV_GET_AI_DATA               0
166 #define EM_ID23_RFBPM_CONV_GET_VALUE_1               0
167 #define EM_ID23_RFBPM_CONV_GET_VALUE_2               1
168 #define EM_ID23_RFBPM_POSITION_AVE_NUM               2
169 #define EM_ID23_RFBPM_POSITION_NUMBER                1
170 #define EM_ID23_RFBPM_POSITION_SAMP_TIM              3
171 #define EM_ID23_RFBPM_STATUS_CHECK              0x0f
172 #define EM_ID23_RFBPM_STATUS_INPUT_PORT              1
173 #define EM_ID23_R_SHIFT_12                          12
174 #define EM_ID23_R_SHIFT_4                            4
175 #define EM_ID23_R_SHIFT_8                            8
176 #define EM_ID23_SIGN_BIT_LIN                    0x10
177 #define EM_ID23_SIP_STATUS_INPUT_PORT                1
178 #define EM_ID23_ST_ADC_CURR_GET_SLAVE                1
179 #define EM_ID23_ST_CURR_CLEAR_DATA_NUM               0
180 #define EM_ID23_ST_CURR_CONV_GET_AI_DATA             0
181 #define EM_ID23_ST_CURR_CONV_GET_VALUE_1             0
182 #define EM_ID23_ST_CURR_CONV_GET_VALUE_2             1
183 #define EM_ID23_ST_CURR_EXEC_CONV_VALUE_1            0
184 #define EM_ID23_ST_CURR_EXEC_CONV_VALUE_2            1
185 #define EM_ID23_ST_CURR_SET_DATA                     0
186 #define EM_ID23_ST_CURR_SET_ST_NUM                   0
187 #define EM_ID23_ST_DAC_CURR_GET_SLAVE                1
188 #define EM_ID23_ST_ERR_RESET_BIT                     2
189 #define EM_ID23_ST_ERR_RESET_DELAY                   3
```

```
190 #define EM_ID23_ST_ERR_RESET_SLAVE                1
191 #define EM_ID23_ST_NUM                           10
192 #define EM_ID23_ST_POWER_PUT_ODIGIT               4
193 #define EM_ID23_ST_POWER_PUT_BIT                  2
194 #define EM_ID23_ST_POWER_PUT_DELAY                3
195 #define EM_ID23_ST_POWER_PUT_SLAVE                1
196 #define EM_ID23_ST_STATUS_CURR                0x0010
197 #define EM_ID23_ST_STATUS_CURR_BIT            0x0004
198 #define EM_ID23_ST_STATUS_FAN                 0x0008
199 #define EM_ID23_ST_STATUS_FAN_BIT             0x0002
200 #define EM_ID23_ST_STATUS_FAULT               0x0080
201 #define EM_ID23_ST_STATUS_FAULT_BIT           0x0040
202 #define EM_ID23_ST_STATUS_GET_SLAVE               1
203 #define EM_ID23_ST_STATUS_ON                  0x0002
204 #define EM_ID23_ST_STATUS_ON2                 0x0001
205 #define EM_ID23_ST_STATUS_ON2_BIT             0x0080
206 #define EM_ID23_ST_STATUS_ON_BIT              0x0001
207 #define EM_ID23_ST_STATUS_REMOTE              0x0004
208 #define EM_ID23_ST_STATUS_REMOTE_BIT          0x0020
209 #define EM_ID23_ST_STATUS_TEMP                0x0040
210 #define EM_ID23_ST_STATUS_TEMP_BIT            0x0010
211 #define EM_ID23_ST_STATUS_VOLT                0x0020
212 #define EM_ID23_ST_STATUS_VOLT_BIT            0x0008
213 #define EM_ID23_TABLE_R_MAX_DATA                500
214 #define EM_ID23_TIME_OUT_NUM                  18000
215 #define EM_ID23_VAC_GET_PORT                      1
216 #define EM_ID23_VAC_STATUS_INPUT_PORT             1
217 #define EM_ID23_VAC_STATUS_KIND                   2
218 #define EM_ID23_VAC_STATUS_KIND_1                 0
219 #define EM_ID23_VAC_STATUS_KIND_2                 1
220 #define EM_ID23_VAC_STATUS_SHIFT_NUM              8
221
222 #endif
```

```
/*(@header_begin)*********************************************************
*  (C Header)                                                             *
*      Name = em_id23_typedef                                             *
*  (Purpose)                                                              *
*     Header file                                                         *
*  (Relation)                                                             *
*                                                                         *
*  (History)                                                              *
*      Jul-12-1997      H.Funayama / iST                                  *
*      98-1-22          Hiramatsu    for long-steering (Bx & By)          *
*      99-04-16         Hiramatsu    introduced set_flg                   *
*      99-05-08         Hiramatsu    added 4 AC-steerings                 *
*************************************************************(@header_end)*/
#ifndef    EM_ID23_TYPEDEF
  #define  EM_ID23_TYPEDEF

    typedef struct {
       int    set_flg;          /* cellar flag PUT=1, GET=0  */
       int    current_digital;  /* present current(digital)  */
       double current_now;      /* present current           */
    } Em_id23_current;

    typedef struct {
       int   set_flg;           /* cellar flag PUT=1, GET=0  */
       double gapha;            /* gap, phase_up or phase_lo */
    } Em_id23_position;

    typedef struct {
       int    set_flg;          /* cellar flag PUT=1, GET=0  */
       double bx_gap[50];       /* long-steering magnet      */
       double bx_phase[50];
       double by_gap[50];
       double by_phase[50];
       double bx_table[50][50];
       double by_table[50][50];
       double s1x_gap[50];      /* AC-steering magnet no.1   */
       double s1x_phase[50];
       double s1y_gap[50];
       double s1y_phase[50];
       double s1x_table[50][50];
       double s1y_table[50][50];
       double s2x_gap[50];      /* AC-steering magnet no.2   */
       double s2x_phase[50];
       double s2y_gap[50];
       double s2y_phase[50];
       double s2x_table[50][50];
       double s2y_table[50][50];
       double s3x_gap[50];      /* AC-steering magnet no.3   */
       double s3x_phase[50];
       double s3y_gap[50];
       double s3y_phase[50];
       double s3x_table[50][50];
       double s3y_table[50][50];
       double s4x_gap[50];      /* AC-steering magnet no.4   */
       double s4x_phase[50];
       double s4y_gap[50];
       double s4y_phase[50];
       double s4x_table[50][50];
       double s4y_table[50][50];
    } Em_id23_table;

#endif
```

```
/*(@header_begin)*******************************************************
 *  (C Header)                                                          *
 *      Name = em_id23_func                                             *
 *  (Purpose)                                                           *
 *      Function prototype is declared here                             *
 *  (Relation)                                                          *
 *      Prototype syntax should be same as those of ANSI-C              *
 *  (History)                                                           *
 *      18-Nov-1996     T.Fukui/SPring-8  for id23                      *
 *      98-02-01        Hiramatsu         revised                       *
 *      99-01-15        Hiramatsu         updated                       *
 *      99-03-03        Hiramatsu         deleted init_counter          *
 *      99-05-05        Hiramatsu         add em_ema_loop_sequence      *
 *      99-06-18        Hiramatsu         revised for GUI-EMA           *
 *      99-07-05        Hiramatsu         join lc_ and st_cur_put       *
 *******************************************************(@header_end)*/
#ifndef    EM_ID23_FUNC
  #define  EM_ID23_FUNC

int em_cntl_init                    ( );
int em_cntl_terminate               ( );
int em_dev_init                     ( );
int em_dev_terminate                ( );
int em_id23_gap_conv_full_put       ( Em_data, Em_data, Em_data * );
int em_id23_gap_conv_put            ( Em_data, Em_data, Em_data * );
int em_id23_gap_init_put            ( Em_data, Em_data, Em_data * );
int em_id23_gap_move_put            ( Em_data, Em_data, Em_data * );
int em_id23_gap_move_put_abs        ( Em_data, Em_data, Em_data * );
int em_id23_gap_preset_put          ( Em_data, Em_data, Em_data * );
int em_id23_gap_status_get          ( Em_data, Em_data, Em_data * );
int em_id23_hold                    ( Em_data, Em_data, Em_data * );
int em_id23_lc_feed_forward         ( );
int em_id23_lc_feed_forward_parm    (double * );
int em_id23_lc_table_set            ( Em_data, Em_data, Em_data * );
int em_id23_lc_table_thru           ( Em_data, Em_data, Em_data * );
int em_id23_lmt_emergency_status_get( Em_data, Em_data, Em_data * );
int em_id23_lmt_gap_status_get      ( Em_data, Em_data, Em_data * );
int em_id23_lmt_phase_status_get    ( Em_data, Em_data, Em_data * );
int em_id23_phase_brake_off_put     ( Em_data, Em_data, Em_data * );
int em_id23_phase_brake_on_put      ( Em_data, Em_data, Em_data * );
int em_id23_phase_conv_put          ( Em_data, Em_data, Em_data * );
int em_id23_phase_init_put          ( Em_data, Em_data, Em_data * );
int em_id23_phase_move_one_side_put ( Em_data, Em_data, Em_data * );
int em_id23_phase_move_put          ( Em_data, Em_data, Em_data * );
int em_id23_phase_position_get      ( Em_data, Em_data, Em_data * );
int em_id23_phase_preset_put        ( Em_data, Em_data, Em_data * );
int em_id23_phase_reset             ( Em_data, Em_data, Em_data * );
int em_id23_phase_status_get        ( Em_data, Em_data, Em_data * );
int em_id23_phase_trigger_off_put   ( Em_data, Em_data, Em_data * );
int em_id23_phase_trigger_on_put    ( Em_data, Em_data, Em_data * );
int em_id23_phase_trigger_status_get( Em_data, Em_data, Em_data * );
int em_id23_pattern_create          ( Em_data, Em_data, Em_data * );
int em_id23_pattern_destroy         ( Em_data, Em_data, Em_data * );
int em_id23_pattern_start           ( Em_data, Em_data, Em_data * );
int em_id23_pattern_stop            ( Em_data, Em_data, Em_data * );
int em_id23_pattern_phase_put       ( Em_data, Em_data, Em_data * );
int em_id23_pattern_hold_put        ( Em_data, Em_data, Em_data * );
int em_id23_pattern_phase_get       ( Em_data, Em_data, Em_data * );
int em_id23_pattern_phase_conv_get  ( Em_data, Em_data, Em_svoc * );
int em_id23_pattern_hold_get        ( Em_data, Em_data, Em_data * );
int em_id23_pattern_hold_conv_get   ( Em_data, Em_data, Em_svoc * );
int em_id23_pattern_status_get      ( Em_data, Em_data, Em_data * );
int em_id23_pattern_status_conv_get ( Em_data, Em_data, Em_svoc * );
```

```
64 int em_id23_position_conv_get        ( Em_data, Em_data, Em_svoc * );
65 int em_id23_position_gap_lin_get     ( Em_data, Em_data, Em_data * );
66 int em_id23_position_gap_rot_get     ( Em_data, Em_data, Em_data * );
67 int em_id23_position_phase_l_lin_get( Em_data, Em_data, Em_data * );
68 int em_id23_position_phase_l_rot_get( Em_data, Em_data, Em_data * );
69 int em_id23_position_phase_u_lin_get( Em_data, Em_data, Em_data * );
70 int em_id23_position_phase_u_rot_get( Em_data, Em_data, Em_data * );
71 int em_id23_rfbpm_conv_get           ( Em_data, Em_data, Em_svoc * );
72 int em_id23_rfbpm_convmm_get         ( Em_data, Em_data, Em_svoc * );
73 int em_id23_rfbpm_position_get       ( Em_data, Em_data, Em_data * );
74 int em_id23_rfbpm_status_get         ( Em_data, Em_data, Em_data * );
75 int em_id23_rio_init                 ( );
76 int em_id23_rio_terminate            ( );
77 int em_id23_sip_status_get           ( Em_data, Em_data, Em_data * );
78 int em_id23_st_adc_curr_get          ( Em_data, Em_data, Em_data * );
79 int em_id23_st_curr_conv_get         ( Em_data, Em_data, Em_svoc * );
80 int em_id23_st_curr_exec             ( Em_data, Em_data, Em_data * );
81 int em_id23_st_curr_put              ( Em_data, Em_data, Em_data * );
82 int em_id23_st_curr_set              ( Em_data, Em_data, Em_data * );
83 int em_id23_st_dac_curr_get          ( Em_data, Em_data, Em_data * );
84 int em_id23_st_err_reset             ( Em_data, Em_data, Em_data * );
85 int em_id23_st_power_off_put         ( Em_data, Em_data, Em_data * );
86 int em_id23_st_power_on_put          ( Em_data, Em_data, Em_data * );
87 int em_id23_st_status_get            ( Em_data, Em_data, Em_data * );
88 int em_id23_status_conv_get          ( Em_data, Em_data, Em_svoc * );
89 int em_id23_ttldio_init              ( );
90 int em_id23_util_delay_time          (int );
91 int util_usleep                      (long );
92 int em_id23_util_position_conv       (int, const int *, double * );
93 int em_id23_util_st_curr_cellar      (int, int, Em_id23_current * );
94 int em_id23_util_table_cellar        (int, Em_id23_table * );
95 int em_id23_util_table_file_get      (char *, Em_id23_table * );
96 int em_id23_vac_conv_get             ( Em_data, Em_data, Em_svoc * );
97 int em_id23_vac_get                  ( Em_data, Em_data, Em_data * );
98 int em_id23_vac_status_get           ( Em_data, Em_data, Em_data * );
99
100 #endif
```

```
/*(@header_begin)************************************************
*  (C Header)                                                   *
*      Name = em_id23_err_define                                *
*  (Purpose)                                                    *
*      EM ID23 library error definition                         *
*  (Relation)                                                   *
*                                                               *
*  (History)                                                    *
*      18-Nov-1996      T.Fukui/SPring-8  for id23              *
*      97-Dec-17        Hiramatsu  revised                      *
*      99-01-15         Hiramatsu  updated                      *
*      99-04-14         Hiramatsu  updated                      *
************************************************(@header_end)*/
#ifndef    EM_ID23_ERR_DEFINE
  #define  EM_ID23_ERR_DEFINE

#define EM_ID23_CNTL_INIT_CURRENT_PUT_ERR                -317230010
#define EM_ID23_CNTL_INIT_CURRENT_SET_ERR                -317230020
#define EM_ID23_CNTL_INIT_FEED_FORWARD_ERR               -317230030
#define EM_ID23_CNTL_INIT_TABLE_GET_ERR                  -317230040
#define EM_ID23_CNTL_INIT_TABLE_STORE_ERR                -317230050
#define EM_ID23_EMA_LOOP_SEQUENCE_HOLD_LIMIT_ERR         -317230060
#define EM_ID23_EMA_LOOP_SEQUENCE_HOLD_L_GET_ERR         -317230070
#define EM_ID23_EMA_LOOP_SEQUENCE_HOLD_L_LOAD_ERR        -317230080
#define EM_ID23_EMA_LOOP_SEQUENCE_HOLD_R_GET_ERR         -317230090
#define EM_ID23_EMA_LOOP_SEQUENCE_HOLD_R_LOAD_ERR        -317230100
#define EM_ID23_EMA_LOOP_SEQUENCE_PHASE_LIMIT_ERR        -317230110
#define EM_ID23_EMA_LOOP_SEQUENCE_PHASE_L_ERR            -317230120
#define EM_ID23_EMA_LOOP_SEQUENCE_PHASE_L_GET_ERR        -317230130
#define EM_ID23_EMA_LOOP_SEQUENCE_PHASE_L_LOAD_ERR       -317230140
#define EM_ID23_EMA_LOOP_SEQUENCE_PHASE_R_ERR            -317230150
#define EM_ID23_EMA_LOOP_SEQUENCE_PHASE_R_GET_ERR        -317230160
#define EM_ID23_EMA_LOOP_SEQUENCE_PHASE_R_LOAD_ERR       -317230170
#define EM_ID23_EMA_LOOP_SEQUENCE_TRIGGER_L_OFF_ERR      -317230180
#define EM_ID23_EMA_LOOP_SEQUENCE_TRIGGER_L_ON_ERR       -317230190
#define EM_ID23_EMA_LOOP_SEQUENCE_TRIGGER_R_OFF_ERR      -317230200
#define EM_ID23_EMA_LOOP_SEQUENCE_TRIGGER_R_ON_ERR       -317230210
#define EM_ID23_EMERGENCY_ERR_ADR_NOT_SET                -317230220
#define EM_ID23_EMERGENCY_STATUS_ERR                     -317230230
#define EM_ID23_FEED_FORWARD_CALCULATION_ERR             -317230240
#define EM_ID23_GAP_INIT_ERR_ADR_PORT_NOT_SET            -317230250
#define EM_ID23_GAP_INIT_ERR_ADR_TIME_NOT_SET            -317230260
#define EM_ID23_GAP_INIT_ERR_FIRST_REFERENCE             -317230270
#define EM_ID23_GAP_INIT_ERR_SECOND_REFERENCE            -317230280
#define EM_ID23_GAP_INIT_ERR_STATUS_GET                  -317230290
#define EM_ID23_GAP_INIT_PUT_ERR                         -317230300
#define EM_ID23_GAP_INIT_PUT_ERR_COUNT_OVER              -317230301
#define EM_ID23_GAP_LMT_ERR_ADR_NOT_SET                  -317230310
#define EM_ID23_GAP_LMT_STATUS_ERR                       -317230320
#define EM_ID23_GAP_MOVE_ERR_ADR_NOT_SET                 -317230330
#define EM_ID23_GAP_MOVE_FEED_FORWARD_ERR                -317230340
#define EM_ID23_GAP_MOVE_PUT_ABS_ERR                     -317230350
#define EM_ID23_GAP_MOVE_PUT_ERR                         -317230360
#define EM_ID23_GAP_PRESENT_GET_ERR                      -317230370
#define EM_ID23_GAP_PRESET_ERR                           -317230380
#define EM_ID23_GAP_PRESET_ERR_ADR_NOT_SET               -317230390
#define EM_ID23_GAP_STATUS_ERR_ADR_NOT_SET               -317230400
#define EM_ID23_GAP_STATUS_GET_ERR                       -317230410
#define EM_ID23_HOLD_TIME_GET_ERR                        -317230420
#define EM_ID23_LC_CURR_ERR_DATA_NOT_SET                 -317230430
#define EM_ID23_LC_H_FEED_FORWARD_ERR_PUT                -317230440
#define EM_ID23_LC_TABLE_FILE_GET_ERR                    -317230450
#define EM_ID23_LC_TABLE_THRU_ERR_ADR_NOT_SET            -317230460
#define EM_ID23_LC_V_FEED_FORWARD_ERR_PUT                -317230470
```

```
 65 #define EM_ID23_PATTERN_CREATE_ERR                          -317230480
 66 #define EM_ID23_PATTERN_DESTROY_ERR                         -317230490
 67 #define EM_ID23_PATTERN_HOLD_CONV_GET_ERR                   -317230500
 68 #define EM_ID23_PATTERN_HOLD_GET_ERR                        -317230510
 69 #define EM_ID23_PATTERN_HOLD_PUT_ERR_HOLD_L                 -317230520
 70 #define EM_ID23_PATTERN_HOLD_PUT_ERR_HOLD_L_PUT             -317230530
 71 #define EM_ID23_PATTERN_HOLD_PUT_ERR_HOLD_R                 -317230540
 72 #define EM_ID23_PATTERN_HOLD_PUT_ERR_HOLD_R_PUT             -317230550
 73 #define EM_ID23_PATTERN_PHASE_CONV_GET_ERR                  -317230560
 74 #define EM_ID23_PATTERN_PHASE_GET_ERR                       -317230570
 75 #define EM_ID23_PATTERN_PHASE_PUT_ERR_PHASE_L               -317230580
 76 #define EM_ID23_PATTERN_PHASE_PUT_ERR_PHASE_L_PUT           -317230590
 77 #define EM_ID23_PATTERN_PHASE_PUT_ERR_PHASE_R               -317230600
 78 #define EM_ID23_PATTERN_PHASE_PUT_ERR_PHASE_R_PUT           -317230610
 79 #define EM_ID23_PATTERN_START_ERR                           -317230620
 80 #define EM_ID23_PATTERN_STATUS_CONV_GET_ERR                 -317230630
 81 #define EM_ID23_PATTERN_STATUS_GET_ERR                      -317230640
 82 #define EM_ID23_PATTERN_STATUS_GET_ERR_NO_VALUE_IN_C        -317230650
 83 #define EM_ID23_PATTERN_STATUS_GET_ERR_OUT                  -317230660
 84 #define EM_ID23_PATTERN_STOP_ERR                            -317230670
 85 #define EM_ID23_PHASE_BRAKE_L_ON_BIT_ERR                    -317230680
 86 #define EM_ID23_PHASE_BRAKE_OFF_ERR_ADR_NOT_SET             -317230690
 87 #define EM_ID23_PHASE_BRAKE_OFF_PUT_ERR                     -317230700
 88 #define EM_ID23_PHASE_BRAKE_ON_ERR_ADR_NOT_SET              -317230710
 89 #define EM_ID23_PHASE_BRAKE_ON_PUT_ERR                      -317230720
 90 #define EM_ID23_PHASE_BRAKE_U_ON_BIT_ERR                    -317230730
 91 #define EM_ID23_PHASE_CONV_ERR_CNV_NOT_SET                  -317230740
 92 #define EM_ID23_PHASE_CONV_MOVE_DATA_NOT_SET                -317230750
 93 #define EM_ID23_PHASE_CONV_PUT_ERR_UNSUPPORTED_MODE         -317230760
 94 #define EM_ID23_PHASE_FEED_FORWARD_ERR                      -317230770
 95 #define EM_ID23_PHASE_INIT_ERR_ADR_NOT_SET                  -317230780
 96 #define EM_ID23_PHASE_INIT_ERR_ADR_PORT_NOT_SET             -317230790
 97 #define EM_ID23_PHASE_INIT_ERR_ADR_TIME_NOT_SET             -317230800
 98 #define EM_ID23_PHASE_INIT_ERR_FIRST_REFERENCE_UP           -317230810
 99 #define EM_ID23_PHASE_INIT_ERR_FIRST_REFERENCE_LOW          -317230811
100 #define EM_ID23_PHASE_INIT_ERR_SECOND_REFERENCE_UP          -317230820
101 #define EM_ID23_PHASE_INIT_ERR_SECOND_REFERENCE_LOW         -317230821
102 #define EM_ID23_PHASE_INIT_ERR_STATUS_GET                   -317230830
103 #define EM_ID23_PHASE_INIT_PUT_ERR                          -317230840
104 #define EM_ID23_PHASE_INIT_PUT_ERR_COUNT_OVER               -317230842
105 #define EM_ID23_PHASE_LMT_ERR_ADR_NOT_SET                   -317230850
106 #define EM_ID23_PHASE_LMT_STATUS_ERR                        -317230860
107 #define EM_ID23_PHASE_L_POSITION_GET_ERR                    -317230870
108 #define EM_ID23_PHASE_L_PUT_ERR                             -317230880
109 #define EM_ID23_PHASE_MOVE_CHECK_ERR                        -317230890
110 #define EM_ID23_PHASE_MOVE_ERR_ADR_NOT_SET                  -317230900
111 #define EM_ID23_PHASE_MOVE_L_PULS_ERR                       -317230910
112 #define EM_ID23_PHASE_MOVE_L_PULS_NUM_ERR                   -317230920
113 #define EM_ID23_PHASE_MOVE_ONE_SIDE_ERR_ADR_NOT_SET         -317230930
114 #define EM_ID23_PHASE_MOVE_U_PULS_ERR                       -317230940
115 #define EM_ID23_PHASE_MOVE_U_PULS_NUM_ERR                   -317230950
116 #define EM_ID23_PHASE_PRESENT_GET_ERR                       -317230960
117 #define EM_ID23_PHASE_PRESET_ERR                            -317230970
118 #define EM_ID23_PHASE_PRESET_ERR_ADR_NOT_SET                -317230980
119 #define EM_ID23_PHASE_RESET_ERR                             -317230990
120 #define EM_ID23_PHASE_RESET_ERR_ADR_NOT_SET                 -317231000
121 #define EM_ID23_PHASE_STATUS_ERR_ADR_NOT_SET                -317231010
122 #define EM_ID23_PHASE_STATUS_GET_ERR                        -317231020
123 #define EM_ID23_PHASE_TRIGGER_OFF_ERR_ADR_NOT_SET           -317231030
124 #define EM_ID23_PHASE_TRIGGER_ON_ERR_ADR_NOT_SET            -317231040
125 #define EM_ID23_PHASE_TRIGGER_STATUS_ERR                    -317231050
126 #define EM_ID23_PHASE_U_POSITION_GET_ERR                    -317231060
127 #define EM_ID23_PHASE_U_PUT_ERR                             -317231070
128 #define EM_ID23_POSITION_CONV_ERR                           -317231080
```

```
129 #define EM_ID23_POSITION_CONV_ERR_CNV_NOT_SET          -317231090
130 #define EM_ID23_POSITION_CONV_ERR_DATA_KIND_DIFFER     -317231100
131 #define EM_ID23_POSITION_CONV_ERR_GAP_MAX_OVER         -317231110
132 #define EM_ID23_POSITION_CONV_ERR_GAP_MIN_OVER         -317231120
133 #define EM_ID23_POSITION_CONV_ERR_PHASE_MAX_OVER       -317231130
134 #define EM_ID23_POSITION_CONV_ERR_PHASE_MIN_OVER       -317231140
135 #define EM_ID23_POSITION_GET_ERR                       -317231150
136 #define EM_ID23_POSITION_GET_ERR_ADR_NOT_SET           -317231160
137 #define EM_ID23_RFBPM_CONV_GET_ERR                     -317231170
138 #define EM_ID23_RFBPM_CONV_GET_ERR_CNV_NOT_SET         -317231180
139 #define EM_ID23_RFBPM_POSITION_GET_ERR                 -317231190
140 #define EM_ID23_RFBPM_POSITION_GET_ERR_ADR_NOT_SET     -317231200
141 #define EM_ID23_RFBPM_STATUS_GET_ERR                   -317231210
142 #define EM_ID23_RFBPM_STATUS_GET_ERR_ADR_NOT_SET       -317231220
143 #define EM_ID23_RIO_INIT_ERR                           -317231230
144 #define EM_ID23_RIO_START_ERR                          -317231240
145 #define EM_ID23_RIO_TERMINATE_ERR                      -317231250
146 #define EM_ID23_SIP_STATUS_GET_ERR                     -317231260
147 #define EM_ID23_SIP_STATUS_GET_ERR_ADR_NOT_SET         -317231270
148 #define EM_ID23_ST1_H_FEED_FORWARD_ERR_PUT             -317231280
149 #define EM_ID23_ST1_V_FEED_FORWARD_ERR_PUT             -317231290
150 #define EM_ID23_ST2_H_FEED_FORWARD_ERR_PUT             -317231300
151 #define EM_ID23_ST2_V_FEED_FORWARD_ERR_PUT             -317231310
152 #define EM_ID23_ST3_H_FEED_FORWARD_ERR_PUT             -317231320
153 #define EM_ID23_ST3_V_FEED_FORWARD_ERR_PUT             -317231330
154 #define EM_ID23_ST4_H_FEED_FORWARD_ERR_PUT             -317231340
155 #define EM_ID23_ST4_V_FEED_FORWARD_ERR_PUT             -317231350
156 #define EM_ID23_STATUS_CONV_GET_ERR                    -317231360
157 #define EM_ID23_ST_ADC_CURR_GET_ERR                    -317231370
158 #define EM_ID23_ST_ADC_CURR_GET_ERR_ADR_NOT_SET        -317231380
159 #define EM_ID23_ST_CELLAR_NOT_PUT                      -317231390
160 #define EM_ID23_ST_CURR_CONV_GET_ERR                   -317231400
161 #define EM_ID23_ST_CURR_CONV_GET_ERR_CNV_NOT_SET       -317231410
162 #define EM_ID23_ST_CURR_ERR_ADR_NOT_SET                -317231420
163 #define EM_ID23_ST_CURR_ERR_CONV_NOT_SET               -317231430
164 #define EM_ID23_ST_CURR_ERR_DATA_NOT_SET               -317231440
165 #define EM_ID23_ST_CURR_PUT_ERR                        -317231450
166 #define EM_ID23_ST_CURR_SET_ERR                        -317231460
167 #define EM_ID23_ST_CURR_SET_ERR_15A_OVER               -317231470
168 #define EM_ID23_ST_CURR_SET_ERR_ADR_NOT_SET            -317231480
169 #define EM_ID23_ST_DAC_CURR_GET_ERR                    -317231490
170 #define EM_ID23_ST_DAC_CURR_GET_ERR_ADR_NOT_SET        -317231500
171 #define EM_ID23_ST_ERR_RESET_ERR                       -317231510
172 #define EM_ID23_ST_ERR_RESET_ERR_ADR_NOT_SET           -317231520
173 #define EM_ID23_ST_POWER_OFF_PUT_ERR                   -317231530
174 #define EM_ID23_ST_POWER_OFF_PUT_ERR_ADR_NOT_SET       -317231540
175 #define EM_ID23_ST_POWER_ON_PUT_ERR                    -317231550
176 #define EM_ID23_ST_POWER_ON_PUT_ERR_ADR_NOT_SET        -317231560
177 #define EM_ID23_ST_STATUS_GET_ERR                      -317231570
178 #define EM_ID23_ST_STATUS_GET_ERR_ADR_NOT_SET          -317231580
179 #define EM_ID23_TABLE_BX_GAP_ERROR                     -317231590
180 #define EM_ID23_TABLE_BX_PHASE_ERROR                   -317231600
181 #define EM_ID23_TABLE_BY_GAP_ERROR                     -317231610
182 #define EM_ID23_TABLE_BY_PHASE_ERROR                   -317231620
183 #define EM_ID23_TABLE_FILE_ERR_OPEN                    -317231630
184 #define EM_ID23_TABLE_S1X_GAP_ERROR                    -317231640
185 #define EM_ID23_TABLE_S1X_PHASE_ERROR                  -317231650
186 #define EM_ID23_TABLE_S1Y_GAP_ERROR                    -317231660
187 #define EM_ID23_TABLE_S1Y_PHASE_ERROR                  -317231670
188 #define EM_ID23_TABLE_S2X_GAP_ERROR                    -317231680
189 #define EM_ID23_TABLE_S2X_PHASE_ERROR                  -317231690
190 #define EM_ID23_TABLE_S2Y_GAP_ERROR                    -317231700
191 #define EM_ID23_TABLE_S2Y_PHASE_ERROR                  -317231710
192 #define EM_ID23_TABLE_S3X_GAP_ERROR                    -317231720
```

```
193 #define EM_ID23_TABLE_S3X_PHASE_ERROR                          -317231730
194 #define EM_ID23_TABLE_S3Y_GAP_ERROR                            -317231740
195 #define EM_ID23_TABLE_S3Y_PHASE_ERROR                          -317231750
196 #define EM_ID23_TABLE_S4X_GAP_ERROR                            -317231760
197 #define EM_ID23_TABLE_S4X_PHASE_ERROR                          -317231770
198 #define EM_ID23_TABLE_S4Y_GAP_ERROR                            -317231780
199 #define EM_ID23_TABLE_S4Y_PHASE_ERROR                          -317231790
200 #define EM_ID23_TTLDIO_INIT_ERR                                -317231800
201 #define EM_ID23_UTIL_POSITION_CELLAR_ERR                       -317231810
202 #define EM_ID23_UTIL_POSITION_CONV_ERR_UNSUPPORTED_MODE        -317231820
203 #define EM_ID23_VAC_GET_ERR                                    -317231830
204 #define EM_ID23_VAC_GET_ERR_ADR_NOT_SET                        -317231840
205 #define EM_ID23_VAC_STATUS_GET_ERR                             -317231850
206 #define EM_ID23_VAC_STATUS_GET_ERR_ADR_NOT_SET                 -317231860
207 #define EM_ID23_VAC_STATUS_GET_ERR_KIND                        -317231870
208
209 #endif
```

```
 1 /*(@header_begin)************************************************
 2 *  (C Header)                                                    *
 3 *      Name = em_ema_err_define                                  *
 4 *  (Purpose)                                                     *
 5 *      ID23 EMA library error definition                         *
 6 *  (Relation)                                                    *
 7 *                                                                *
 8 *  (History)                                                     *
 9 *      99-05-04         Hiramatsu  created                       *
10 ************************************************(@header_end)*/
11 #ifndef EM_ID23_EMA_ERR_DEFINE
12  #define EM_ID23_EMA_ERR_DEFINE
13
14 #define EM_ID23_EMA_LOOP_SEQUENCE_PHASE_R_LOAD_ERR   -317231001
15 #define EM_ID23_EMA_LOOP_SEQUENCE_PHASE_L_LOAD_ERR   -317231002
16 #define EM_ID23_EMA_LOOP_SEQUENCE_HOLD_R_LOAD_ERR    -317231003
17 #define EM_ID23_EMA_LOOP_SEQUENCE_HOLD_L_LOAD_ERR    -317231004
18 #define EM_ID23_EMA_LOOP_SEQUENCE_PHASE_R_GET_ERR    -317231005
19 #define EM_ID23_EMA_LOOP_SEQUENCE_PHASE_L_GET_ERR    -317231006
20 #define EM_ID23_EMA_LOOP_SEQUENCE_HOLD_R_GET_ERR     -317231007
21 #define EM_ID23_EMA_LOOP_SEQUENCE_HOLD_L_GET_ERR     -317231008
22 #define EM_ID23_EMA_LOOP_SEQUENCE_PHASE_R_ERR        -317231009
23 #define EM_ID23_EMA_LOOP_SEQUENCE_PHASE_L_ERR        -317231010
24 #define EM_ID23_EMA_LOOP_SEQUENCE_TRIGGER_R_ON_ERR   -317231011
25 #define EM_ID23_EMA_LOOP_SEQUENCE_TRIGGER_R_OFF_ERR  -317231012
26 #define EM_ID23_EMA_LOOP_SEQUENCE_TRIGGER_L_ON_ERR   -317231013
27 #define EM_ID23_EMA_LOOP_SEQUENCE_TRIGGER_L_OFF_ERR  -317231014
28 #define EM_ID23_EMA_LOOP_SEQUENCE_HOLD_LIMIT_ERR     -317231015
29 #define EM_ID23_EMA_LOOP_SEQUENCE_PHASE_LIMIT_ERR    -317231016
30
31 #endif
```

```
 1 #include <rpc/types.h>
 2
 3 #define EM_SVOCSIZE 256
 4
 5 struct Em_svoc {
 6         int error;
 7         char svoc[EM_SVOCSIZE];
 8 };
 9 typedef struct Em_svoc Em_svoc;
10 bool_t xdr_Em_svoc();
11
12 #define MEMPROG ((u_long)0x30000100)
13 #define MEMVERS ((u_long)1)
14 #define EM_CALL_INIT ((u_long)1)
15 extern int *em_call_init_1();
16 #define EM_CALL_SVOC ((u_long)2)
17 extern Em_svoc *em_call_svoc_1();
18 #define EM_CALL_TERMINATE ((u_long)3)
19 extern int *em_call_terminate_1();
20 #define EM_CALL_STATUS ((u_long)4)
21 extern int *em_call_status_1();
22 #define EM_CALL_FORCEDTERMINATE ((u_long)5)
23 extern int *em_call_forcedterminate_1();
```

Appendix I: Source Code for EMA and Equation of Movement

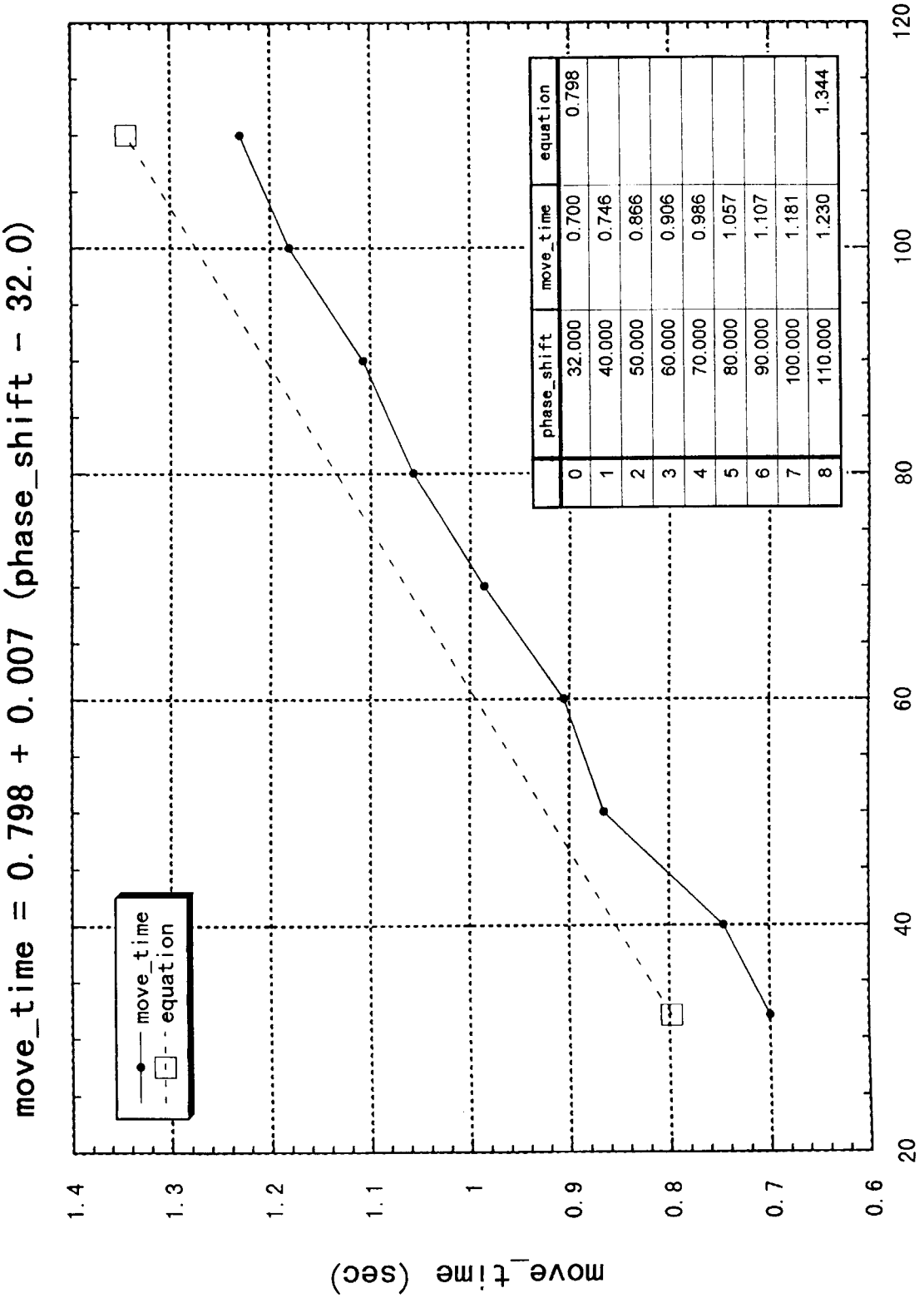
```
/*(@header_begin)*************************************************************
 *  (C Function)                                                             *
 *     Name = em_ema_loop_sequence                                           *
 *     Type = int                                                            *
 *  (Purpose)                                                                *
 *     EMA loop sequence                                                     *
 *  (Input)                                                                  *
 *                                                                           *
 *  (Output)                                                                 *
 *     none                                                                  *
 *  (Return)                                                                 *
 *     = 0 : ok                                                              *
 *     < 0 : fail                                                            *
 *  (Physical unit)                                                          *
 *     Pa, V, A, mm, sec                                                     *
 *  (Relation)                                                               *
 *     none                                                                  *
 *  (History)                                                                *
 *     99-05-04      Hiramatsu          created.                             *
 *     99-05-31      Hiramatsu          changed usleep() to susleep()        *
 *     99-08-28      Hiramatsu          included hardware-timer              *
 *************************************************************(@header_end)*/
#include <rpc/rpc.h>
#include <unistd.h>
#include <sys/timers.h>
#include <time.h>
#include "sr_em.h"
#include "sr_util.h"
#include "em_id23.h"

int em_ema_loop_sequence()
{
   int err;
   char num[30];
   static int count=0;                          /* for safety control        */
   static int hold_r=0, hold_l=0;               /* hold-time [msec]          */
   static double phase_r=0.0, phase_l=0.0;      /* phase-shift(D) [mm]       */
   int holdsec;                                 /* sleep-time [sec]          */
   int holdmicro;                               /* sleep-time [microsec]     */
   int waitmicro;                               /* hardware-timer [microsec] */
   double wtime_r, wtime_l;                     /* wait-time [sec]           */
   double start, stop_r, stop_l;                /* absolute-time [sec]       */
   double tmovemax, distance;                   /* move-time[sec] & dist[mm] */
   struct timespec tp;
   Em_data param;                               /* cellar                    */
   Em_svoc *ret, command;                       /* for recursive call        */

   if (count == 0) {                      /* verify count */
      err = em_ema_store(EM_VAR_LOAD, "phase_r", &param);  /* from cellar */
      if (err < 0) {
         return EM_ID23_EMA_LOOP_SEQUENCE_PHASE_R_LOAD_ERR;
      }
      err = em_data_get_double(param, 0, &phase_r);        /* allocate */
      if (err < 0) {
         return EM_ID23_EMA_LOOP_SEQUENCE_PHASE_R_GET_ERR;
      }
      err = em_ema_store(EM_VAR_LOAD, "phase_l", &param);  /* from cellar */
      if (err < 0) {
         return EM_ID23_EMA_LOOP_SEQUENCE_PHASE_L_LOAD_ERR;
      }
```

```
61        err = em_data_get_double(param, 0, &phase_l);          /* allocate */
62        if (err < 0) {
63           return EM_ID23_EMA_LOOP_SEQUENCE_PHASE_L_GET_ERR;
64        }
65        err = em_ema_store(EM_VAR_LOAD, "hold_r", &param);     /* from cellar */
66        if (err < 0) {
67           return EM_ID23_EMA_LOOP_SEQUENCE_HOLD_R_LOAD_ERR;
68        }
69        err = em_data_get_int(param, 0, &hold_r);              /* allocate */
70        if (err < 0) {
71           return EM_ID23_EMA_LOOP_SEQUENCE_HOLD_R_GET_ERR;
72        }
73        err = em_ema_store(EM_VAR_LOAD, "hold_l", &param);     /* from cellar */
74        if (err < 0) {
75           return EM_ID23_EMA_LOOP_SEQUENCE_HOLD_L_LOAD_ERR;
76        }
77        err = em_data_get_int(param, 0, &hold_l);              /* allocate */
78        if (err < 0) {
79           return EM_ID23_EMA_LOOP_SEQUENCE_HOLD_L_GET_ERR;
80        }
81
82        if (hold_r < 200 || hold_l < 200) {            /* avoid < 200msec */
83           return EM_ID23_EMA_LOOP_SEQUENCE_HOLD_LIMIT_ERR;
84        }
85
86        if (phase_r<-120. || phase_r> 120. || phase_l<-120. || phase_l> 120.) {
87           return EM_ID23_EMA_LOOP_SEQUENCE_PHASE_LIMIT_ERR;
88        }
89
90        distance = (phase_r - phase_l) / 2.0;          /* move-distance */
91        if (distance < 0.0) {
92           distance = - distance;
93        }
94        tmovemax = 0.798 + 0.007 * (distance - 32.0);  /* max move-time */
95     }
96
97     if (count > 5) {       /* stop the movement */
98        return 0;
99     }
100
101    strcpy(command.svoc,"s/put/bl_id23_phase_trigger_l/off");
102    ret = em_call_svoc_1(command);                     /* trigger-off */
103    if ((ret -> error) < 0) {
104       return EM_ID23_EMA_LOOP_SEQUENCE_TRIGGER_L_OFF_ERR;
105    }
106
107    getclock(TIMEOFDAY, &tp);                          /* start-time */
108    start = (double)tp.tv_sec + (double)tp.tv_nsec/1.0e9;
109
110    strcpy(command.svoc,"s/put/bl_id23_phase/");
111    util_ftoa_fixed(2, phase_r, num);
112    strcat(command.svoc, num);
113    strcat(command.svoc, "mm");
114    ret = em_call_svoc_1(command);                 /* move to right */
115    if ((ret -> error) < 0) {
116       return EM_ID23_EMA_LOOP_SEQUENCE_PHASE_R_ERR;
117    }
118
119    getclock(TIMEOFDAY, &tp);                      /* stop-time */
120    stop_r = (double)tp.tv_sec + (double)tp.tv_nsec/1.0e9;
```

```
121
122     wtime_r   = start + tmovemax - stop_r;
123     waitmicro = (int)(wtime_r * 1000000.0);
124     util_usleep(waitmicro);         /* hardware-timer */
125
126     strcpy(command.svoc,"s/put/bl_id23_phase_trigger_r/on");
127     ret = em_call_svoc_1(command);                  /* trigger-on */
128     if ((ret -> error) < 0) {
129        return EM_ID23_EMA_LOOP_SEQUENCE_TRIGGER_R_ON_ERR;
130     }
131
132     holdsec   = hold_r  / 1000;
133     holdmicro = (hold_r % 1000) * 1000;
134     susleep(holdsec, holdmicro);                    /* hold */
135
136     strcpy(command.svoc,"s/put/bl_id23_phase_trigger_r/off");
137     ret = em_call_svoc_1(command);                  /* trigger-off */
138     if ((ret -> error) < 0) {
139        return EM_ID23_EMA_LOOP_SEQUENCE_TRIGGER_R_OFF_ERR;
140     }
141
142     strcpy(command.svoc,"s/put/bl_id23_phase/");
143     util_ftoa_fixed(2, phase_l, num);
144     strcat(command.svoc, num);
145     strcat(command.svoc, "mm");
146     ret = em_call_svoc_1(command);                  /* move to left */
147     if ((ret -> error) < 0) {
148        return EM_ID23_EMA_LOOP_SEQUENCE_PHASE_L_ERR;
149     }
150
151     getclock(TIMEOFDAY, &tp);                       /* stop time */
152     stop_l = (double)tp.tv_sec + (double)tp.tv_nsec/1.0e9;
153
154     wtime_l   =  start + 2 * tmovemax + (double)hold_r/1000.0 - stop_l;
155     waitmicro = (int)(wtime_l * 1000000.0);
156     util_usleep(waitmicro);         /* hardware-timer */
157
158     strcpy(command.svoc,"s/put/bl_id23_phase_trigger_l/on");
159     ret = em_call_svoc_1(command);                  /* trigger-on */
160     if ((ret -> error) < 0) {
161        return EM_ID23_EMA_LOOP_SEQUENCE_TRIGGER_L_ON_ERR;
162     }
163
164     holdsec   = hold_l  / 1000;
165     holdmicro = (hold_l % 1000) * 1000;
166     susleep(holdsec, holdmicro);                    /* hold */
167
168     count = count + 1;                      /* increment count */
169
170     return 0;
171 }
```

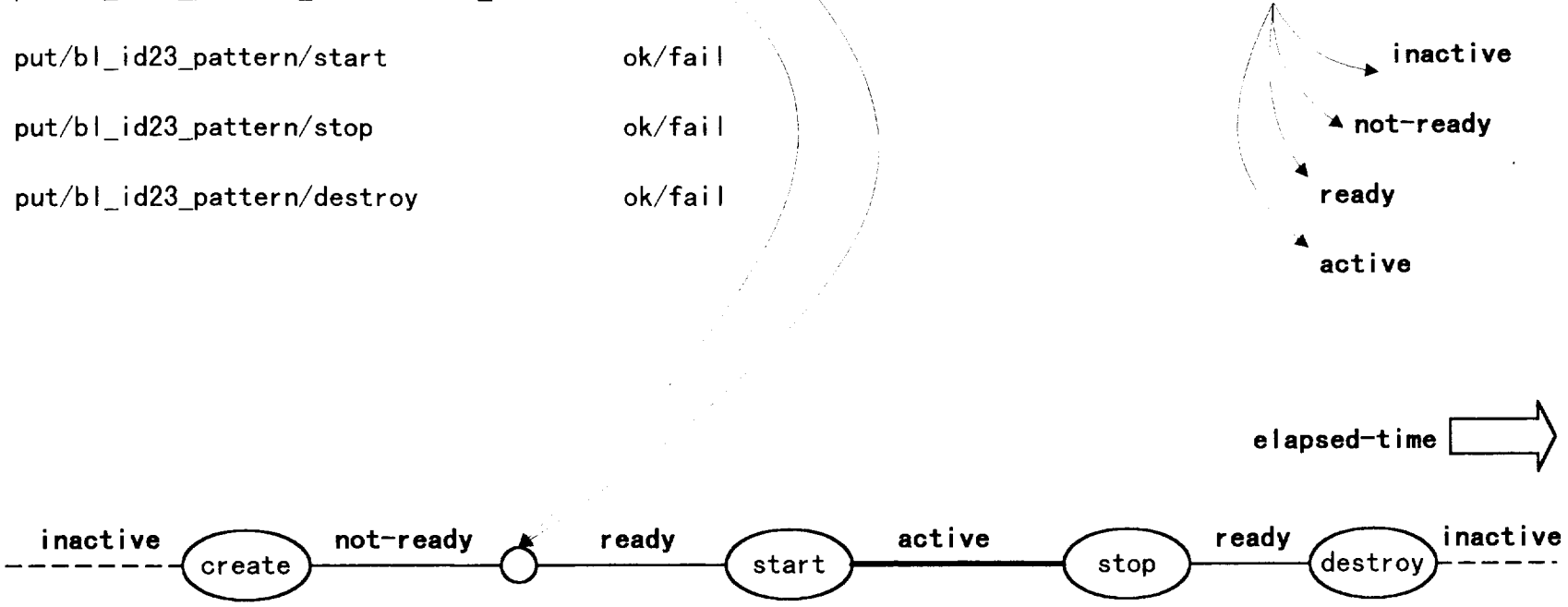


	phase_shift	move_time	equation
0	32.000	0.700	0.798
1	40.000	0.746	
2	50.000	0.866	
3	60.000	0.906	
4	70.000	0.986	
5	80.000	1.057	
6	90.000	1.107	
7	100.000	1.181	
8	110.000	1.230	1.344

Appendix J: Status Information for EMA Process

SVOC Commands for Periodic Movement of Phase (emitted at GUI)

SVOC command	reply	SVOC command	reply
put/bl_id23_pattern/create	ok/fail	get/bl_id23_pattern_phase/position	%fmm_%fmm
put/bl_id23_pattern_phase/%fmm_%fmm	ok/fail	get/bl_id23_pattern_hold/time	%dmsec_%dmsec
put/bl_id23_pattern_hold/%dmsec_%dmsec	ok/fail	**get/bl_id23_pattern/status**	
put/bl_id23_pattern/start	ok/fail		
put/bl_id23_pattern/stop	ok/fail		
put/bl_id23_pattern/destroy	ok/fail		



Appendix K: Source Code for EM

em_cntl_init.c
em_cntl_terminate.c
em_dev_init.c
em_dev_terminate.c
em_id23_gap_init_put.c
em_id23_gap_move_put.c
em_id23_gap_move_put_abs.c
em_id23_gap_preset_put.c
em_id23_gap_status_get.c
em_id23_hold.c
em_id23_lc_feed_forward.c
em_id23_lc_feed_forward_parm.c
em_id23_lc_table_set.c
em_id23_lc_table_thru.c
em_id23_lmt_emergency_status_get.c
em_id23_lmt_gap_status_get.c
em_id23_lmt_phase_status_get.c
em_id23_pattern_create.c
em_id23_pattern_destroy.c
em_id23_pattern_hold_conv_get.c
em_id23_pattern_hold_get.c
em_id23_pattern_hold_put.c
em_id23_pattern_phase_conv_get.c
em_id23_pattern_phase_get.c
em_id23_pattern_phase_put.c
em_id23_pattern_start.c
em_id23_pattern_status_conv_get.c
em_id23_pattern_status_get.c
em_id23_pattern_stop.c
em_id23_phase_brake_off_put.c
em_id23_phase_brake_on_put.c
em_id23_phase_conv_put.c
em_id23_phase_init_put.c
em_id23_phase_move_one_side_put.c
em_id23_phase_move_put.c
em_id23_phase_position_get.c
em_id23_phase_preset_put.c
em_id23_phase_reset.c
em_id23_phase_status_get.c
em_id23_phase_trigger_off_put.c
em_id23_phase_trigger_on_put.c
em_id23_phase_trigger_status_get.c
em_id23_position_conv_get.c
em_id23_position_gap_lin_get.c
em_id23_position_gap_rot_get.c
em_id23_position_phase_l_lin_get.c
em_id23_position_phase_l_rot_get.c
em_id23_position_phase_u_lin_get.c
em_id23_position_phase_u_rot_get.c
em_id23_rfbpm_conv_get.c
em_id23_rfbpm_convmm_get.c
em_id23_rfbpm_position_get.c
em_id23_rfbpm_status_get.c
em_id23_rio_init.c
em_id23_rio_terminate.c
em_id23_sip_status_get.c
em_id23_st_adc_curr_get.c
em_id23_st_curr_conv_get.c
em_id23_st_curr_exec.c
em_id23_st_curr_put.c
em_id23_st_curr_set.c
em_id23_st_dac_curr_get.c
em_id23_st_err_reset.c
em_id23_st_power_off_put.c
em_id23_st_power_on_put.c
em_id23_st_status_get.c
em_id23_status_conv_get.c
em_id23_ttldio_init.c
em_id23_util_delay_time.c
em_id23_util_position_cellar.c
em_id23_util_position_conv.c
em_id23_util_st_curr_cellar.c
em_id23_util_table_cellar.c
em_id23_util_table_file_get.c
em_id23_vac_conv_get.c
em_id23_vac_get.c
em_id23_vac_status_get.c

```
 1 /*(@header_begin)*************************************************************
 2 *  (C Function)                                                              *
 3 *      Name = em_cntl_init                                                   *
 4 *      Type = int                                                            *
 5 *  (Purpose)                                                                 *
 6 *      RIO hot start(long-steering magnet)                                   *
 7 *      zero Ampere start(AC-steering magnet)                                 *
 8 *  (Input)                                                                   *
 9 *      none                                                                  *
10 *  (Output)                                                                  *
11 *      none                                                                  *
12 *  (Return)                                                                  *
13 *      zero : success                                                        *
14 *      < 0  : fail                                                           *
15 *  (Physical unit)                                                           *
16 *      Pa, V, A, mm, sec                                                     *
17 *  (Relation)                                                                *
18 *                                                                            *
19 *  (History)                                                                 *
20 *      98-02-01     Hiramatsu   created                                      *
21 *      98-02-12     Hiramatsu   revised for RIO hot-start                    *
22 *      99-03-09     Hiramatsu   updated for RIO hot-start                    *
23 *      99-05-14     Hiramatsu   updated for 4 AC-steering magnets            *
24 *************************************************************(@header_end)*/
25
26 #include "em_id23.h"
27
28 int em_cntl_init()
29 {
30     int err, fd, mode;
31     double gap, phase, lc[10];
32     char svoc[256], unit[20], fname[EM_ID23_LC_TABLE_THRU_FILE_MAX];
33     char device_name[EM_MIX_DATA_CHAR_MAX];
34     Em_id23_table table;
35     Em_id23_position pos;
36     pos.set_flg = 0;
37
38 #ifdef EM_DEBUG_PRINTF
39     printf("Call func name is ¥"em_cntl_init¥"¥n");
40 #endif
41
42 /* RIO Hot start by loading the correction table */
43 #ifndef EM_DEBUG_SIM
44     #ifdef EM_DEBUG_TELLEM
45         sprintf(fname, "bl_id23_steer0.tbl");
46     #else
47         sprintf(fname, "/home/bl/blcntl/bl_id23_steer0.tbl");
48     #endif /* EM_DEBUG_TELLEM */
49 #else
50     sprintf(fname, "bl_id23_steer0.tbl");
51 #endif /* EM_DEBUG_SIM */
52
53     err = em_id23_util_table_file_get(fname, &table);
54     if (err < 0) {
55         return EM_ID23_CNTL_INIT_TABLE_GET_ERR;
56     }
57
58     err = em_id23_util_table_cellar(EM_ID23_PUT_TABLE, &table);
59     if (err < 0) {
60         return EM_ID23_CNTL_INIT_TABLE_STORE_ERR;
61     }
62
63 #ifdef EM_DEBUG_PRINTF
64     printf("fname is %s¥n", fname);
```

```
65      printf("set flag is %d¥n", table.set_flg);
66  #endif
67
68  #ifdef EM_DEBUG_SIM
69      pos.gapha = 300.0;
70      em_id23_util_position_cellar(EM_ID23_PUT_POSITION, 0, &pos);
71      if (pos.set_flg == 0) {
72         return EM_ID23_UTIL_POSITION_CELLAR_ERR;
73      }
74
75      pos.gapha = 0.0;
76      em_id23_util_position_cellar(EM_ID23_PUT_POSITION, 1, &pos);
77      if (pos.set_flg == 0) {
78         return EM_ID23_UTIL_POSITION_CELLAR_ERR;
79      }
80
81      pos.gapha = 0.0;
82      em_id23_util_position_cellar(EM_ID23_PUT_POSITION, 2, &pos);
83      if (pos.set_flg == 0) {
84         return EM_ID23_UTIL_POSITION_CELLAR_ERR;
85      }
86  #endif
87
88      err = em_id23_lc_feed_forward_parm(lc);
89      if (err < 0) {
90           return EM_ID23_CNTL_INIT_FEED_FORWARD_ERR;
91      }
92
93      sprintf(svoc, "s/set/bl_id23_lc_h/%fA", lc[0]);
94      err = em_idcom_call_svoc(svoc);
95      if (err < 0) {
96           return EM_ID23_CNTL_INIT_CURRENT_SET_ERR;
97      }
98
99      sprintf(svoc, "s/set/bl_id23_lc_v/%fA", lc[1]);
100     err = em_idcom_call_svoc(svoc);
101     if (err < 0) {
102          return EM_ID23_CNTL_INIT_CURRENT_SET_ERR;
103     }
104
105 /* Set zero Ampere to AC-steering magnet */
106     sprintf(svoc, "s/set/bl_id23_st_h_1/%fA", lc[2]);
107     err = em_idcom_call_svoc(svoc);
108     if (err < 0) {
109          return EM_ID23_CNTL_INIT_CURRENT_SET_ERR;
110     }
111
112     sprintf(svoc, "s/set/bl_id23_st_v_1/%fA", lc[3]);
113     err = em_idcom_call_svoc(svoc);
114     if (err < 0) {
115          return EM_ID23_CNTL_INIT_CURRENT_SET_ERR;
116     }
117
118     sprintf(svoc, "s/set/bl_id23_st_h_2/%fA", lc[4]);
119     err = em_idcom_call_svoc(svoc);
120     if (err < 0) {
121          return EM_ID23_CNTL_INIT_CURRENT_SET_ERR;
122     }
123
124     sprintf(svoc, "s/set/bl_id23_st_v_2/%fA", lc[5]);
125     err = em_idcom_call_svoc(svoc);
126     if (err < 0) {
127          return EM_ID23_CNTL_INIT_CURRENT_SET_ERR;
128     }
```

```
129
130     sprintf(svoc, "s/set/bl_id23_st_h_3/%fA", Ic[6]);
131     err = em_idcom_call_svoc(svoc);
132     if (err < 0) {
133          return EM_ID23_CNTL_INIT_CURRENT_SET_ERR;
134     }
135
136     sprintf(svoc, "s/set/bl_id23_st_v_3/%fA", Ic[7]);
137     err = em_idcom_call_svoc(svoc);
138     if (err < 0) {
139          return EM_ID23_CNTL_INIT_CURRENT_SET_ERR;
140     }
141
142     sprintf(svoc, "s/set/bl_id23_st_h_4/%fA", Ic[8]);
143     err = em_idcom_call_svoc(svoc);
144     if (err < 0) {
145          return EM_ID23_CNTL_INIT_CURRENT_SET_ERR;
146     }
147
148     sprintf(svoc, "s/set/bl_id23_st_v_4/%fA", Ic[9]);
149     err = em_idcom_call_svoc(svoc);
150     if (err < 0) {
151          return EM_ID23_CNTL_INIT_CURRENT_SET_ERR;
152     }
153
154 /* RIO communication start */
155 #ifndef EM_DEBUG_SIM
156     device_name[0] = '¥0' ;
157     mode = EM_VAR_GET_ALL_FIRST;
158     while (em_save_fd(device_name, mode, &fd) != EM_SAVE_FD_END) {
159         mode = EM_VAR_GET_ALL;
160         if (strstr(device_name, "dev/rio_") != NULL) {
161             err = dev_rio_start(fd , 1);
162             if (err != 0) {
163                 return EM_ID23_RIO_START_ERR ;
164             }
165         }
166     }
167 #endif
168
169 /* Put current to long-steering and AC-steering magnet */
170     sprintf(svoc, "s/put/bl_id23_st/exec");
171     err = em_idcom_call_svoc(svoc);
172     if (err < 0) {
173         return EM_ID23_CNTL_INIT_CURRENT_PUT_ERR;
174     }
175
176    return 0 ;
177 }
```

```
 1 /*(@header_begin)*************************************************************
 2 *  (C Function)                                                              *
 3 *      Name = em_cntl_terminate                                              *
 4 *      Type = int                                                            *
 5 *  (Purpose)                                                                 *
 6 *                                                                            *
 7 *  (Input)                                                                   *
 8 *      none                                                                  *
 9 *  (Output)                                                                  *
10 *      none                                                                  *
11 *  (Return)                                                                  *
12 *                                                                            *
13 *  (Physical unit)                                                           *
14 *                                                                            *
15 *  (Relation)                                                                *
16 *                                                                            *
17 *  (History)                                                                 *
18 *      Feb- 1-1998      Hiramatsu                  created.                  *
19 *                                                                            *
20 ****************************************************************(@header_end)*/
21
22 #include "em_id23.h"
23
24 int em_cntl_terminate()
25 {
26
27 #ifdef EM_DEBUG_PRINTF
28    printf("Call func name is ¥"em_cntl_terminate¥"¥n");
29 #endif
30
31    return 0 ;
32 }
```

```
 1 /*(@header_begin)*************************************************************
 2 *  (C Function)                                                              *
 3 *      Name = em_dev_init                                                    *
 4 *      Type = int                                                            *
 5 *  (Purpose)                                                                 *
 6 *      initialize each device                                                *
 7 *  (Input)                                                                   *
 8 *      None                                                                  *
 9 *  (Output)                                                                  *
10 *      None                                                                  *
11 *  (Return)                                                                  *
12 *      = 0 : ok                                                              *
13 *      < 0 : error code                                                      *
14 *  (Physical unit)                                                           *
15 *      Pa, V, A, mm, sec                                                     *
16 *  (Relation)                                                                *
17 *                                                                            *
18 *  (History)                                                                 *
19 *      Aug-14-1996       A.Taketani / SPring8      created.                  *
20 *      18-Nov-1996       T.Fukui/SPring-8 Modified for bl_id23.              *
21 *      98-01-11          Hiramatsu      adapted for ID23                     *
22 *      99-03-03          Hiramatsu      deleted init_counter                 *
23 *************************************************************(@header_end)*/
24
25 #include "em_id23.h"
26
27 int em_dev_init()
28 {
29    int err;
30
31 #ifdef EM_DEBUG_PRINTF
32    printf("Call func name is ¥"em_dev_init¥"¥n");
33 #endif
34
35 #ifndef EM_DEBUG_SIM
36    err = em_id23_ttldio_init();
37    if (err < 0) {
38       return EM_ID23_TTLDIO_INIT_ERR ;
39    }
40
41    err = em_id23_rio_init();
42    if (err < 0) {
43       return EM_ID23_RIO_INIT_ERR ;
44    }
45 #endif
46
47    return 0 ;
48 }
49
```

```
 1 /*(@header_begin)*************************************************************
 2 *  (C Function)                                                              *
 3 *       Name = em_dev_terminate                                              *
 4 *       Type = int                                                           *
 5 *  (Purpose)                                                                 *
 6 *       Terminate each device                                                *
 7 *  (Input)                                                                   *
 8 *       None                                                                 *
 9 *  (Output)                                                                  *
10 *       None                                                                 *
11 *  (Return)                                                                  *
12 *       = 0 : ok                                                             *
13 *       < 0 : error code                                                     *
14 *  (Physical unit)                                                           *
15 *       Pa, V, A, mm, sec                                                    *
16 *  (Relation)                                                                *
17 *                                                                            *
18 *  (History)                                                                 *
19 *      Aug-14-1996      A.Taketani / SPring8      created.                   *
20 *      18-Nov-1996      T.Fukui/SPring-8 Modified for bl_id23.               *
21 *      98-01-11         Hiramatsu      adapted for ID23                      *
22 *      99-03-03         Hiramatsu      deleted init_counter                  *
23 *************************************************************(@header_end)*/
24
25 #include "em_id23.h"
26
27 int em_dev_terminate()
28 {
29    int err;
30
31 #ifdef EM_DEBUG_PRINTF
32    printf("Call func name is ¥"em_dev_terminate¥"¥n");
33 #endif
34
35 #ifndef EM_DEBUG_SIM
36    err = em_id23_rio_terminate();
37    if (err < 0) {
38       return EM_ID23_RIO_TERMINATE_ERR;
39    }
40 #endif
41
42    return 0;
43 }
44
45
```

```
/*(@header_begin)*************************************************************
 *  (C Function)                                                             *
 *      Name = em_id23_gap_init_put                                          *
 *      Type = int                                                           *
 *  (Purpose)                                                                *
 *      obtain gap reference position (max 50 trials)                        *
 *  (Input)                                                                  *
 *      none                                                                 *
 *  (Output)                                                                 *
 *      none                                                                 *
 *  (Return)                                                                 *
 *      = 0   : ok                                                           *
 *      < 0   : fail                                                         *
 *  (Physical unit)                                                          *
 *      Pa, V, A, mm, sec                                                    *
 *  (Relation)                                                               *
 *                                                                           *
 *  (History)                                                                *
 *      99-06-15      Hiramatsu     created                                  *
 *************************************************************(@header_end)*/

#include "em_id23.h"

int em_id23_gap_init_put(Em_data in, Em_data adr, Em_data *out)
{
    int fd_0, fd_1, err, count = 0, diff, status;
    int cw_kind = 1;            /* direction to open gap  */
    int axis = 1;               /* axis is gap            */
    int repetition = 0;         /* number of repetition   */
    int time_num;               /* delay time [microsec]  */
    int dev_num;                /* port number for DI     */
    int di_data[4];             /* DI status data         */
    char svoc[256];             /* SVOC recursive call    */
    char unit[20];              /* unit [mm]              */
    double difference;          /* difference [mm]        */
    double present_gap;         /* absolute gap [mm]      */
    double gap[50];             /* gap reference [mm]     */

#ifdef EM_DEBUG_PRINTF
    printf("Call func name is ¥"em_id23_gap_init_put¥"¥n");
#endif

/* get file descriptor */
    fd_0 = em_get_fd_multi(adr, 0);     /* PTG0350 device */
    fd_1 = em_get_fd_multi(adr, 2);     /* DI device      */

/* read config.tbl */
    err = em_data_get_int(adr, EM_ID23_GAP_INIT_DELAY_TIME, &time_num);
    if (err < 0) {
        return EM_ID23_GAP_INIT_ERR_ADR_TIME_NOT_SET;
    }

    err = em_data_get_int(adr, EM_ID23_GAP_INIT_INPUT_PORT, &dev_num);
    if (err < 0) {
        return EM_ID23_GAP_INIT_ERR_ADR_PORT_NOT_SET;
    }

/* read first gap reference */
    err = em_idcom_func_svoc("s/get/bl_id23_gap/position",
                             &present_gap, unit);
    if (err < 0) {
        return EM_ID23_GAP_INIT_ERR_FIRST_REFERENCE;
    }
    gap[0] = present_gap;
```

```
65
66 printf("Gap[0] is %f¥n", gap[0]);
67    do {
68 /* move to a position far from gap reference(closing gap) */
69       sprintf(svoc,"s/put/bl_id23_gap_relat/%.3fmm", -2.0);
70       em_idcom_call_svoc(svoc);
71
72       sleep(3);
73
74 /* move to a position near to gap reference(opening gap) */
75       sprintf(svoc,"s/put/bl_id23_gap_relat/%.3fmm", 1.8);
76       em_idcom_call_svoc(svoc);
77
78       sleep(1);
79
80 #ifndef EM_DEBUG_SIM
81       do {
82 /* move 1 pulse by 1 pulse */
83            dev_ptg0350_move_one(fd_0, axis, cw_kind);
84            usleep(time_num);
85 /* read gap reference on/off */
86            err = dev_di_get_diall(fd_1, di_data);
87            if (err < 0) {
88                return EM_ID23_GAP_INIT_ERR_STATUS_GET;
89            }
90            status = (di_data[dev_num] & EM_ID23_GAP_LMT_HOME_BIT) >> 6;
91       } while (status == 0x0);
92
93       sleep(3);
94
95       do {
96            dev_ptg0350_move_one(fd_0, axis, cw_kind);
97            sleep(1);
98            dev_di_get_diall(fd_1, di_data);
99            status = (di_data[dev_num] & EM_ID23_GAP_LMT_HOME_BIT) >> 6;
100      } while (status == 0x0);
101 #endif
102
103      sleep(3);
104
105 /* read second gap reference */
106      err = em_idcom_func_svoc("s/get/bl_id23_gap/position",
107                               &present_gap, unit);
108      if (err < 0) {
109          return EM_ID23_GAP_INIT_ERR_SECOND_REFERENCE;
110      }
111
112      count += 1;
113
114      if (count > 50) {
115          return EM_ID23_GAP_INIT_PUT_ERR_COUNT_OVER;
116      }
117
118      gap[count] = present_gap;
119
120 /* calculate difference from previous gap-reference */
121      difference = gap[count] - gap[count - 1];
122      diff = (int)(difference * 1000);
123
124 /* repete movement until meet 3 consecutive rdgs */
125      if (abs(diff) < 5) {
126         repetition += 1;
127      } else {
128         repetition = 0;
```

```
129         }
130
131         sleep(1);
132
133     } while (repetition < 3);
134
135 /* call SVOC to preset linear-scale and rotary-encoder
136     sprintf(svoc, "s/put/bl_id23_gap/preset");
137     err = em_idcom_call_svoc(svoc);
138     if (err < 0) {
139         return EM_ID23_GAP_INIT_PUT_ERR;
140     }
141 */
142
143 /* prepare output */
144     em_data_clear(out);
145     em_data_put_int(out, 0, 0);
146
147     return 0;
148 }
```

```
1 /*(@header_begin)*************************************************************
2 *  (C Function)                                                              *
3 *      Name = em_id23_gap_move_put                                           *
4 *      Type = int                                                            *
5 *  (Purpose)                                                                 *
6 *      drive gap motor, put current to                                       *
7 *      long-steering magnet based on the correction table                    *
8 *  (Input)                                                                   *
9 *      parameter (config.tbl)                                                *
10 *     v_max     = target generation rate (pulse/sec)                       *
11 *     v_min     = generation rate at the start (pulse/sec)                 *
12 *     acc_cycle = time-step to change from v to v+1 (microsec)             *
13 *  (Output)                                                                  *
14 *      none                                                                 *
15 *  (Return)                                                                  *
16 *      0   : success                                                        *
17 *      < 0 : fail                                                           *
18 *  (Physical unit)                                                           *
19 *      Pa, V, A, mm, sec                                                    *
20 *  (Relation)                                                                *
21 *                                                                           *
22 *  (History)                                                                 *
23 *      98- 1-19      Hiramatsu       created                                *
24 *      98-10-07      Hiramatsu       modified for pulse_num=zero            *
25 *      99-01-08      Hiramatsu       updated for wtime(from config.tbl)     *
26 *************************************************************(@header_end)*/
27
28 #include "em_id23.h"
29
30 int em_id23_gap_move_put(Em_data in, Em_data adr, Em_data *out)
31 {
32    int err, fd, counter, pulse_num, wtime, data_temp;
33    char pulse_out_status;
34    double x_data, y_data;
35    unsigned short v_max , v_min, acc_cycle;
36
37 #ifdef EM_DEBUG_PRINTF
38    printf("Call func name is ¥"em_id23_gap_move_put¥"¥n");
39 #endif
40
41 /* get file descriptor */
42    fd = em_get_fd(adr);
43
44 /* get target gap distance */
45    err = em_data_get_double(in, EM_ID23_GAP_CONV_MOVE_DATA, &x_data);
46    if (err < 0) {
47       return EM_ID23_GAP_MOVE_PUT_ERR;
48    }
49    y_data= EM_ID23_GAP_MOVE_CONV_COEFF * x_data;
50    pulse_num = (int)y_data;
51
52 /* read config.tbl */
53    err = em_data_get_int(adr, EM_ID23_GAP_MOVE_V_MAX, &data_temp);
54    if (err < 0) {
55       return EM_ID23_GAP_MOVE_ERR_ADR_NOT_SET;
56    }
57    v_max= (unsigned short)data_temp;
58
59    err = em_data_get_int(adr, EM_ID23_GAP_MOVE_V_MIN, &data_temp);
60    if (err < 0) {
61       return EM_ID23_GAP_MOVE_ERR_ADR_NOT_SET;
62    }
63    v_min= (unsigned short)data_temp;
64
```

```
65    err = em_data_get_int(adr, EM_ID23_GAP_MOVE_ACC_CYCLE, &data_temp);
66    if (err < 0) {
67       return EM_ID23_GAP_MOVE_ERR_ADR_NOT_SET;
68    }
69    acc_cycle= (unsigned short)data_temp;
70
71    err = em_data_get_int(adr, EM_ID23_GAP_MOVE_WTIME, &wtime);
72    if (err < 0) {
73       return EM_ID23_GAP_MOVE_ERR_ADR_NOT_SET ;
74    }
75
76 /* drive pulse motor for gap movement */
77 #ifndef EM_DEBUG_SIM
78    dev_ptg0350_move(fd, pulse_num , 0, acc_cycle,
79                            v_max, v_min, 0, EM_ID23_TIME_OUT_NUM);
80 #endif
81
82 /* put current to long-steering magnet based on correction table */
83    do {
84
85 #ifdef EM_DEBUG_PRINTF
86       counter = counter + 1;
87 #endif /* EM_DEBUG_PRINTF */
88
89       usleep(wtime * 10000);
90
91       err = em_id23_lc_feed_forward();
92       if (err < 0) {
93           return EM_ID23_GAP_MOVE_FEED_FORWARD_ERR;
94       }
95
96 #ifndef EM_DEBUG_SIM
97       dev_ptg0350_out_check(fd, &pulse_out_status);
98 #else
99       pulse_out_status = 0;
100 #endif
101
102    } while (pulse_out_status != 0);     /* no more pulse when pulse_out_status 0
103 */
104
105 #ifdef EM_DEBUG_PRINTF
106    printf("COUNTER is = %d ", counter );
107 #endif /* EM_DEBUG_PRINTF */
108
109      err = em_id23_lc_feed_forward();
110      if (err < 0) {
111          return EM_ID23_GAP_MOVE_FEED_FORWARD_ERR;
112      }
113
114 /* prepare output */
115    em_data_clear(out);
116    em_data_put_int(out, 0, 0);
117
118    return 0;
119 }
```

```
/*(@header_begin)************************************************************
 *  (C Function)                                                             *
 *       Name = em_id23_gap_move_put_abs                                     *
 *       Type = int                                                          *
 *  (Purpose)                                                                *
 *       read present gap, calculate relative distance to be moved,          *
 *       and move                                                            *
 *  (Input)                                                                  *
 *       target gap                                                          *
 *  (Output)                                                                 *
 *       none                                                                *
 *  (Return)                                                                 *
 *       0    : success                                                      *
 *       < 0 : fail                                                          *
 *  (Physical unit)                                                          *
 *      Pa, V, A, mm, sec                                                    *
 *  (Relation)                                                               *
 *                                                                           *
 *  (History)                                                                *
 *      Jan- 6-1998      Hiramatsu        created                            *
 *      98-10-14         Hiramatsu        updated                            *
 ************************************************************(@header_end)*/

#include "em_id23.h"

int em_id23_gap_move_put_abs(Em_data in, Em_data adr, Em_data *out)
{
    int err;
    char unit[20], comm[100];
    double presentgap, value, distance;
    Em_id23_position pos;
    pos.set_flg = 0;

#ifdef EM_DEBUG_PRINTF
    printf("Call func name is ¥"em_id23_gap_move_put_abs¥"¥n");
#endif

/* get target gap */
    err = em_data_get_double(in, 0, &value);
    if (err < 0) {
        return EM_ID23_GAP_MOVE_PUT_ABS_ERR;
    }

#ifndef EM_DEBUG_SIM
/* read present gap from linear scale */
    err = em_idcom_func_svoc("s/get/bl_id23_gap/position", &presentgap, unit);
    if (err < 0) {
        return EM_ID23_GAP_MOVE_PUT_ABS_ERR;
    }
#else
/* PUT position to cellar */
    pos.gapha = value;
    em_id23_util_position_cellar(EM_ID23_PUT_POSITION, 0, &pos);
    if (pos.set_flg == 0) {
       return EM_ID23_UTIL_POSITION_CELLAR_ERR;
    }
/* GET position from cellar */
    em_id23_util_position_cellar(EM_ID23_GET_POSITION, 0, &pos);
    if (pos.set_flg == 0) {
       return EM_ID23_UTIL_POSITION_CELLAR_ERR;
    }
    presentgap =  pos.gapha;
#endif

```

```
65 /* gap distance to be moved */
66    distance = value - presentgap;
67
68 /* drive pulse motor until reach target gap */
69    sprintf(comm, "s/put/bl_id23_gap_relat/%.3fmm", distance);
70    err = em_idcom_call_svoc(comm);
71    if (err < 0) {
72        return EM_ID23_GAP_MOVE_PUT_ABS_ERR;
73    }
74
75 /* prepare output */
76    em_data_clear(out);
77    em_data_put_int(out,0,0);
78
79    return 0;
80 }
```

```
/*(@header_begin)************************************************************
 *  (C Function)                                                              *
 *      Name = em_id23_gap_preset_put                                         *
 *      Type = int                                                            *
 *  (Purpose)                                                                 *
 *      preset rotary encoder and linear scale                                *
 *  (Input)                                                                   *
 *      parameter (config.tbl)                                                *
 *  (Output)                                                                  *
 *      none                                                                  *
 *  (Return)                                                                  *
 *      0   : ok                                                              *
 *      < 0 : fail                                                            *
 *  (Physical unit)                                                           *
 *      Pa, V, A, mm, sec                                                     *
 *  (Relation)                                                                *
 *                                                                            *
 *  (History)                                                                 *
 *      Jul-06-1997      F.Funayama /iST  created.                            *
 *      98- 1-11         Hiramatsu    revised                                 *
 *      98-10-14         Hiramatsu    updated                                 *
 ************************************************************(@header_end)*/

#include "em_id23.h"

int em_id23_gap_preset_put(Em_data in, Em_data adr, Em_data *out)
{
    int fd, err, dev_num_1, dev_num_2;
    int do_data;                /* preset-on(=1) , preset-off(=0) */
    int time_num;               /* delay time [microsec]          */

#ifdef  EM_DEBUG_PRINTF
    printf("Call func name is ¥"em_id23_gap_preset_put¥"¥n");
#endif

/* get file descriptor */
    fd = em_get_fd(adr);

/* read config.tbl */
    err = em_data_get_int(adr, EM_ID23_GAP_PRESET_LIN, &dev_num_1);
    if (err < 0) {
        return EM_ID23_GAP_PRESET_ERR_ADR_NOT_SET;
    }

    err = em_data_get_int(adr, EM_ID23_GAP_PRESET_ROT, &dev_num_2);
    if (err < 0) {
        return EM_ID23_GAP_PRESET_ERR_ADR_NOT_SET;
    }

    err = em_data_get_int(adr, EM_ID23_GAP_PRESET_DELAY_TIME, &time_num);
    if (err < 0) {
        return EM_ID23_GAP_PRESET_ERR_ADR_NOT_SET;
    }

/* preset-on */
     do_data = 1;

#ifndef EM_DEBUG_SIM
    err = dev_do_put_dobit(fd, dev_num_1, &do_data);
    if (err < 0) {
        return EM_ID23_GAP_PRESET_ERR;
    }

    err = dev_do_put_dobit(fd, dev_num_2, &do_data);
```

```
65      if (err < 0) {
66          return EM_ID23_GAP_PRESET_ERR;
67      }
68  #endif
69
70  /* sleep */
71      usleep(time_num);
72
73  /* preset-off(release) */
74      do_data = 0;
75
76  #ifndef EM_DEBUG_SIM
77      err = dev_do_put_dobit(fd, dev_num_1, &do_data);
78      if (err < 0) {
79          return EM_ID23_GAP_PRESET_ERR;
80      }
81
82      err = dev_do_put_dobit(fd, dev_num_2, &do_data);
83      if (err < 0) {
84          return EM_ID23_GAP_PRESET_ERR;
85      }
86  #endif
87
88  /* prepare output */
89      em_data_clear(out);
90      em_data_put_int(out, 0, 0);
91
92      return 0;
93  }
```

```
 1 /*(@header_begin)************************************************************
 2 *  (C Function)                                                             *
 3 *      Name = em_id23_gap_status_get                                        *
 4 *      Type = int                                                           *
 5 *  (Purpose)                                                                *
 6 *      read device status                                                   *
 7 *  (Input)                                                                  *
 8 *      parameter (config.tbl)                                               *
 9 *  (Output)                                                                 *
10 *      out->data[0].eint : di_data                                          *
11 *  (Return)                                                                 *
12 *      0   : ok                                                             *
13 *      < 0 : fail                                                           *
14 *  (Physical unit)                                                          *
15 *      Pa, V, A, mm, sec                                                    *
16 *  (Relation)                                                               *
17 *                                                                           *
18 *  (History)                                                                *
19 *      Jul-10-1997      F.Funayama /iST  created.                           *
20 *      98- 1-11          Hiramatsu      revised                            *
21 *      98-10-14          Hiramatsu      updated                            *
22 ************************************************************(@header_end)*/
23
24 #include "em_id23.h"
25
26 int em_id23_gap_status_get(Em_data in, Em_data adr, Em_data *out)
27 {
28     int fd, dev_num, di_data, err;
29
30 /* get file descriptor */
31     fd = em_get_fd(adr);
32
33 /* read config.tbl */
34     err = em_data_get_int(adr, EM_ID23_GAP_STATUS_INPUT_BIT, &dev_num);
35     if (err < 0) {
36         return EM_ID23_GAP_STATUS_ERR_ADR_NOT_SET;
37     }
38
39 /* get over-heat signal */
40 #ifndef EM_DEBUG_SIM
41     err = dev_di_get_dibit(fd, dev_num, &di_data);
42     if (err < 0) {
43         return EM_ID23_GAP_STATUS_GET_ERR ;
44     }
45 #else
46     di_data = 0x0;
47 #endif
48
49 /* prepare output */
50     em_data_clear(out);
51     em_data_put_int(out, 0, di_data);
52
53     return 0;
54 }
```

```
 1 /*(@header_begin)*************************************************************
 2 *  (C Function)                                                              *
 3 *      Name = em_id23_hold                                                   *
 4 *      Type = int                                                            *
 5 *  (Purpose)                                                                 *
 6 *      sleep hold-time[msec] for phase-pattern movement                      *
 7 *  (Input)                                                                   *
 8 *      none                                                                  *
 9 *  (Output)                                                                  *
10 *      none                                                                  *
11 *  (Return)                                                                  *
12 *                                                                            *
13 *  (Physical unit)                                                           *
14 *      Pa, V, A, mm, sec                                                     *
15 *  (Relation)                                                                *
16 *                                                                            *
17 *  (History)                                                                 *
18 *      Jan-24-1999     Hiramatsu   created.                                  *
19 *      99-06-09        Hiramatsu   adapted for susleep()                     *
20 *************************************************************(@header_end)*/
21
22 #include "em_id23.h"
23
24 int em_id23_hold(Em_data in, Em_data adr, Em_data *out)
25 {
26    int err;
27    int hold;               /* sleep-time [msec]     */
28    int waitsec;            /* sleep-time [sec]      */
29    int waitmicro;          /* sleep-time [microsec] */
30
31 #ifdef EM_DEBUG_PRINTF
32    printf("Call func name is ¥"em_id23_hold¥"¥n");
33 #endif
34
35 /*  key-in the hold-time */
36    err = em_data_get_double(in, 0, &hold);
37    if (err < 0) {
38        return EM_ID23_HOLD_TIME_GET_ERR ;
39    }
40
41 /* convert and then sleep */
42    waitsec = hold / 1000;
43    waitmicro = (hold % 1000) * 1000;
44    susleep(waitsec, waitmicro);
45
46 /* prepare output */
47    em_data_clear(out);
48    em_data_put_int(out, 0, 0);
49
50    return 0;
51 }
```

```
 1 /*(@header_begin)*************************************************************
 2 *  (C Function)                                                              *
 3 *      Name = em_id23_lc_feed_forward                                        *
 4 *      Type = int                                                            *
 5 *  (Purpose)                                                                 *
 6 *      calculate current for Bx and By long-steering magnet                  *
 7 *      put current to Bx and By power-supply                                 *
 8 *  (Input)                                                                   *
 9 *                                                                            *
10 *  (Output)                                                                  *
11 *                                                                            *
12 *  (Return)                                                                  *
13 *      0   : ok                                                              *
14 *      < 0 : fail                                                            *
15 *  (Physical unit)                                                           *
16 *      Pa, V, A, mm, sec                                                     *
17 *  (Relation)                                                                *
18 *                                                                            *
19 *  (History)                                                                 *
20 *      98- 1-20      Hiramatsu             created.                          *
21 *      98-10-20      Hiramatsu             revised for SVOC call             *
22 *      99-05-08      Hiramatsu             added 4 AC-steerings              *
23 *************************************************************(@header_end)*/
24 #include "em_id23.h"
25
26 int em_id23_lc_feed_forward()
27 {
28    int err;
29    char svoc[100], unit[30];
30    double lc[10];   /* lc[0] for Bx  magnet, lc[1] for By  magnet */
31                     /* lc[2] for S1x magnet, lc[3] for S1y magnet */
32                     /* lc[4] for S2x magnet, lc[5] for S2y magnet */
33                     /* lc[6] for S3x magnet, lc[7] for S3y magnet */
34                     /* lc[8] for S4x magnet, lc[9] for S4y magnet */
35
36 #ifdef EM_DEBUG_PRINTF
37    printf("Call func name is ¥"em_id23_lc_feed_forward¥"¥n");
38 #endif
39
40 /* calculate current to long-steering magnets */
41    err = em_id23_lc_feed_forward_parm(lc);
42    if (err < 0) {
43        return EM_ID23_FEED_FORWARD_CALCULATION_ERR;
44    }
45
46 /* set current to long-steering magnets */
47    sprintf(svoc, "s/set/bl_id23_lc_h/%fA", lc[0]);
48    err = em_idcom_call_svoc(svoc);
49    if (err < 0) {
50        return EM_ID23_LC_H_FEED_FORWARD_ERR_PUT;
51    }
52    sprintf(svoc, "s/set/bl_id23_lc_v/%fA", lc[1]);
53    err = em_idcom_call_svoc(svoc);
54    if (err < 0) {
55        return EM_ID23_LC_V_FEED_FORWARD_ERR_PUT;
56    }
57
58 /* set current to AC-steering magnets */
59    sprintf(svoc, "s/set/bl_id23_st_h_1/%fA", lc[2]);
60    err = em_idcom_call_svoc(svoc);
61    if (err < 0) {
62        return EM_ID23_ST1_H_FEED_FORWARD_ERR_PUT;
63    }
64    sprintf(svoc, "s/set/bl_id23_st_v_1/%fA", lc[3]);
```

```
65      err = em_idcom_call_svoc(svoc);
66      if (err < 0) {
67          return EM_ID23_ST1_V_FEED_FORWARD_ERR_PUT;
68      }
69      sprintf(svoc, "s/set/bl_id23_st_h_2/%fA", Ic[4]);
70      err = em_idcom_call_svoc(svoc);
71      if (err < 0) {
72          return EM_ID23_ST2_H_FEED_FORWARD_ERR_PUT;
73      }
74      sprintf(svoc, "s/set/bl_id23_st_v_2/%fA", Ic[5]);
75      err = em_idcom_call_svoc(svoc);
76      if (err < 0) {
77          return EM_ID23_ST2_V_FEED_FORWARD_ERR_PUT;
78      }
79      sprintf(svoc, "s/set/bl_id23_st_h_3/%fA", Ic[6]);
80      err = em_idcom_call_svoc(svoc);
81      if (err < 0) {
82          return EM_ID23_ST3_H_FEED_FORWARD_ERR_PUT;
83      }
84      sprintf(svoc, "s/set/bl_id23_st_v_3/%fA", Ic[7]);
85      err = em_idcom_call_svoc(svoc);
86      if (err < 0) {
87          return EM_ID23_ST3_V_FEED_FORWARD_ERR_PUT;
88      }
89      sprintf(svoc, "s/set/bl_id23_st_h_4/%fA", Ic[8]);
90      err = em_idcom_call_svoc(svoc);
91      if (err < 0) {
92          return EM_ID23_ST4_H_FEED_FORWARD_ERR_PUT;
93      }
94      sprintf(svoc, "s/set/bl_id23_st_v_4/%fA", Ic[9]);
95      err = em_idcom_call_svoc(svoc);
96      if (err < 0) {
97          return EM_ID23_ST4_V_FEED_FORWARD_ERR_PUT;
98      }
99
100 /* exec long-steering and AC-steering magnet */
101     sprintf(svoc, "s/put/bl_id23_st/exec");
102     err = em_idcom_call_svoc(svoc);
103     if (err < 0) {
104         return EM_ID23_ST_CURR_PUT_ERR;
105     }
106
107     return 0;
108 }
```

```
/*(@header_begin)************************************************************
 *  (C Function)                                                             *
 *      Name = em_id23_lc_feed_forward_parm                                  *
 *      Type = int                                                           *
 *  (Purpose)                                                                *
 *      read gap and phase(D)                                                *
 *      get correction table from cellar                                     *
 *      calculate current for long steering magnet                           *
 *  (Input)                                                                  *
 *      correction table (from cellar)                                       *
 *  (Output)                                                                 *
 *      none                                                                 *
 *  (Return)                                                                 *
 *      0    : ok                                                            *
 *      < 0 : fail                                                           *
 *  (Physical unit)                                                          *
 *      Pa, V, A, mm, sec                                                    *
 *  (Relation)                                                               *
 *                                                                           *
 *  (History)                                                                *
 *      98-01-22    Hiramatsu    created                                     *
 *      99-05-09    Hiramatsu    added 4 AC-steering magnets                 *
 ************************************************************(@header_end)*/

#include "em_id23.h"

int em_id23_lc_feed_forward_parm(double *lc)

{
    int err, i, j;
    char unit[30];
    double gap ;                    /* absolute gap    [mm]          */
    double phase_up ;               /* upper phase     [mm]          */
    double phase_lo ;               /* lower phase     [mm]          */
    double phase ;                  /* phase-shift(D)  [mm]          */
    double factorgx , factorgy ;    /* proportion factor for gap     */
    double factorpx , factorpy ;    /* proportion factor for phase   */
    double curr1x , curr1y, curr2x , curr2y ;
    Em_id23_table table;
    Em_id23_position pos;

#ifdef EM_DEBUG_PRINTF
    printf("Call func is ¥"em_id23_lc_feed_forward_parm¥"¥n");
#endif

#ifndef EM_DEBUG_SIM
/* read present gap */
    err = em_idcom_func_svoc("s/get/bl_id23_gap/position", &gap, unit);
    if (err < 0) {
        return EM_ID23_GAP_PRESENT_GET_ERR ;
    }

/* read present phase-shift(D) */
    err = em_idcom_func_svoc("s/get/bl_id23_phase/position", &phase, unit);
    if (err < 0) {
        return EM_ID23_PHASE_PRESENT_GET_ERR ;
    }
#else
    em_id23_util_position_cellar(EM_ID23_GET_POSITION, 0, &pos);
    if (pos.set_flg == 0) {
        return EM_ID23_UTIL_POSITION_CELLAR_ERR;
    }
    gap   = pos.gapha;

```

```
65     em_id23_util_position_cellar(EM_ID23_GET_POSITION, 1, &pos);
66     if (pos.set_flg == 0) {
67         return EM_ID23_UTIL_POSITION_CELLAR_ERR;
68     }
69     phase_up = pos.gapha;
70
71     em_id23_util_position_cellar(EM_ID23_GET_POSITION, 2, &pos);
72     if (pos.set_flg == 0) {
73         return EM_ID23_UTIL_POSITION_CELLAR_ERR;
74     }
75     phase_lo = pos.gapha;
76     phase = phase_up - phase_lo;
77 #endif
78
79 /* get correction table from cellar */
80     err = em_id23_util_table_cellar(EM_ID23_GET_TABLE, &table) ;
81     if (table.set_flg == 0) {
82         return EM_ID23_LC_CURR_ERR_DATA_NOT_SET ;
83     }
84
85 /* linear interpolation for bx_table(Bx coil) */
86     i = 0;
87     if (phase < table.bx_phase[i]) {
88         return EM_ID23_TABLE_BX_PHASE_ERROR;
89     } else {
90         for (i = 1 ; i < 50 ; ++i) {
91             if (phase < table.bx_phase[i]) {
92                 break;
93             }
94         }
95     }
96
97     j = 0;
98     if (gap < table.bx_gap[j]) {
99         return EM_ID23_TABLE_BX_GAP_ERROR ;
100    } else {
101        for (j = 1 ; j < 50 ; ++j) {
102            if (gap < table.bx_gap[j]) {
103                break;
104            }
105        }
106    }
107
108    factorpx = (phase - table.bx_phase[i-1]) /
109               (table.bx_phase[i] - table.bx_phase[i-1]);
110    curr1x   = table.bx_table[j-1][i-1] + factorpx *
111               (table.bx_table[j-1][i] - table.bx_table[j-1][i-1]);
112    curr2x   = table.bx_table[j][i-1]   + factorpx *
113               (table.bx_table[j][i]   - table.bx_table[j][i-1]);
114    factorgx = (gap - table.bx_gap[j-1]) /
115               (table.bx_gap[j] - table.bx_gap[j-1]);
116    lc[0]    = curr1x + factorgx * (curr2x - curr1x);
117
118 #ifdef EM_DEBUG_PRINTF
119    printf("lc[0] is %f¥n", lc[0]);
120 #endif
121
122 /* linear interpolation for by_table(By coil) */
123    i = 0;
124    if (phase < table.by_phase[i]) {
125        return EM_ID23_TABLE_BY_PHASE_ERROR ;
126    } else {
127        for (i = 1 ; i < 50 ; ++i) {
128            if (phase < table.by_phase[i]) {
```

```
129                 break ;
130             }
131         }
132     }
133
134     j = 0;
135     if (gap < table.by_gap[j]) {
136         return EM_ID23_TABLE_BY_GAP_ERROR;
137     } else {
138         for (j = 1 ; j < 50 ; ++j) {
139             if (gap < table.by_gap[j]) {
140                 break ;
141             }
142         }
143     }
144
145     factorpy = (phase - table.by_phase[i-1]) /
146                (table.by_phase[i] - table.by_phase[i-1]);
147     curr1y   = table.by_table[j-1][i-1] + factorpy *
148                (table.by_table[j-1][i] - table.by_table[j-1][i-1]);
149     curr2y   = table.by_table[j][i-1]   + factorpy *
150                (table.by_table[j][i] - table.by_table[j][i-1]);
151     factorgy = (gap - table.by_gap[j-1]) /
152                (table.by_gap[j] - table.by_gap[j-1]);
153     lc[1]    = curr1y + factorgy * (curr2y - curr1y);
154
155 #ifdef EM_DEBUG_PRINTF
156     printf("lc[1] is %f¥n", lc[1]);
157 #endif
158
159 /* linear interpolation for s1x_table(S1x coil) */
160     i = 0;
161     if (phase < table.s1x_phase[i]) {
162         return EM_ID23_TABLE_S1X_PHASE_ERROR;
163     } else {
164         for (i = 1 ; i < 50 ; ++i) {
165             if (phase < table.s1x_phase[i]) {
166                 break;
167             }
168         }
169     }
170
171     j = 0;
172     if (gap < table.s1x_gap[j]) {
173         return EM_ID23_TABLE_S1X_GAP_ERROR;
174     } else {
175         for (j = 1 ; j < 50 ; ++j) {
176             if (gap < table.s1x_gap[j]) {
177                 break;
178             }
179         }
180     }
181
182     factorpx = (phase - table.s1x_phase[i-1]) /
183                (table.s1x_phase[i] - table.s1x_phase[i-1]);
184     curr1x   = table.s1x_table[j-1][i-1] + factorpx *
185                (table.s1x_table[j-1][i] - table.s1x_table[j-1][i-1]);
186     curr2x   = table.s1x_table[j][i-1]   + factorpx *
187                (table.s1x_table[j][i]   - table.s1x_table[j][i-1]);
188     factorgx = (gap - table.s1x_gap[j-1]) /
189                (table.s1x_gap[j] - table.s1x_gap[j-1]);
190     lc[2]    = curr1x + factorgx * (curr2x - curr1x);
191
192 #ifdef EM_DEBUG_PRINTF
```

```
193     printf("lc[2] is %f¥n", lc[2]);
194 #endif
195
196 /* linear interpolation for s1y_table(S1y coil) */
197     i = 0;
198     if (phase < table.s1y_phase[i]) {
199         return EM_ID23_TABLE_S1Y_PHASE_ERROR;
200     } else {
201         for (i = 1 ; i < 50 ; ++i) {
202             if (phase < table.s1y_phase[i]) {
203                 break ;
204             }
205         }
206     }
207
208     j = 0;
209     if (gap < table.s1y_gap[j]) {
210         return EM_ID23_TABLE_S1Y_GAP_ERROR;
211     } else {
212         for (j = 1 ; j < 50 ; ++j) {
213             if (gap < table.s1y_gap[j]) {
214                 break ;
215             }
216         }
217     }
218
219     factorpy = (phase - table.s1y_phase[i-1]) /
220                (table.s1y_phase[i] - table.s1y_phase[i-1]);
221     curr1y   = table.s1y_table[j-1][i-1] + factorpy *
222                (table.s1y_table[j-1][i] - table.s1y_table[j-1][i-1]);
223     curr2y   = table.s1y_table[j][i-1]   + factorpy *
224                (table.s1y_table[j][i] - table.s1y_table[j][i-1]);
225     factorgy = (gap - table.s1y_gap[j-1]) /
226                (table.s1y_gap[j] - table.s1y_gap[j-1]);
227     lc[3]    = curr1y + factorgy * (curr2y - curr1y);
228
229 #ifdef EM_DEBUG_PRINTF
230     printf("lc[3] is %f¥n", lc[3]);
231 #endif
232
233 /* linear interpolation for s2x_table(S2x coil) */
234     i = 0;
235     if (phase < table.s2x_phase[i]) {
236         return EM_ID23_TABLE_S2X_PHASE_ERROR;
237     } else {
238         for (i = 1 ; i < 50 ; ++i) {
239             if (phase < table.s2x_phase[i]) {
240                 break;
241             }
242         }
243     }
244
245     j = 0 ;
246     if (gap < table.s2x_gap[j]) {
247         return EM_ID23_TABLE_S2X_GAP_ERROR;
248     } else {
249         for (j = 1 ; j < 50 ; ++j) {
250             if (gap < table.s2x_gap[j]) {
251                 break;
252             }
253         }
254     }
255
256     factorpx = (phase - table.s2x_phase[i-1]) /
```

```
257                 (table.s2x_phase[i] - table.s2x_phase[i-1]);
258     curr1x    = table.s2x_table[j-1][i-1] + factorpx *
259                 (table.s2x_table[j-1][i] - table.s2x_table[j-1][i-1]);
260     curr2x    = table.s2x_table[j][i-1]   + factorpx *
261                 (table.s2x_table[j][i]   - table.s2x_table[j][i-1]);
262     factorgx  = (gap - table.s2x_gap[j-1]) /
263                 (table.s2x_gap[j] - table.s2x_gap[j-1]);
264     lc[4]     = curr1x + factorgx * (curr2x - curr1x);
265
266 #ifdef EM_DEBUG_PRINTF
267     printf("lc[4] is %f¥n", lc[4]);
268 #endif
269
270 /* linear interpolation for s2y_table(S2y coil) */
271     i = 0;
272     if (phase < table.s2y_phase[i]) {
273         return EM_ID23_TABLE_S2Y_PHASE_ERROR;
274     } else {
275         for (i = 1 ; i < 50 ; ++i) {
276             if (phase < table.s2y_phase[i]) {
277                 break ;
278             }
279         }
280     }
281
282     j = 0;
283     if (gap < table.s2y_gap[j]) {
284         return EM_ID23_TABLE_S2Y_GAP_ERROR ;
285     } else {
286         for (j = 1 ; j < 50 ; ++j) {
287             if (gap < table.s2y_gap[j]) {
288                 break ;
289             }
290         }
291     }
292
293     factorpy = (phase - table.s2y_phase[i-1]) /
294                 (table.s2y_phase[i] - table.s2y_phase[i-1]);
295     curr1y    = table.s2y_table[j-1][i-1] + factorpy *
296                 (table.s2y_table[j-1][i] - table.s2y_table[j-1][i-1]);
297     curr2y    = table.s2y_table[j][i-1]   + factorpy *
298                 (table.s2y_table[j][i] - table.s2y_table[j][i-1]);
299     factorgy = (gap - table.s2y_gap[j-1]) /
300                 (table.s2y_gap[j] - table.s2y_gap[j-1]);
301     lc[5]     = curr1y + factorgy * (curr2y - curr1y);
302
303 #ifdef EM_DEBUG_PRINTF
304     printf("lc[5] is %f¥n", lc[5]);
305 #endif
306
307 /* linear interpolation for s3x_table(S3x coil) */
308     i = 0;
309     if (phase < table.s3x_phase[i]) {
310         return EM_ID23_TABLE_S3X_PHASE_ERROR ;
311     } else {
312         for (i = 1 ; i < 50 ; ++i) {
313             if (phase < table.s3x_phase[i]) {
314                 break;
315             }
316         }
317     }
318
319     j = 0;
320     if (gap < table.s3x_gap[j]) {
```

```
321         return EM_ID23_TABLE_S3X_GAP_ERROR ;
322     } else {
323         for (j = 1 ; j < 50 ; ++j) {
324             if (gap < table.s3x_gap[j]) {
325                 break;
326             }
327         }
328     }
329
330     factorpx = (phase - table.s3x_phase[i-1]) /
331                (table.s3x_phase[i] - table.s3x_phase[i-1]);
332     curr1x   = table.s3x_table[j-1][i-1] + factorpx *
333                (table.s3x_table[j-1][i] - table.s3x_table[j-1][i-1]);
334     curr2x   = table.s3x_table[j][i-1]   + factorpx *
335                (table.s3x_table[j][i]   - table.s3x_table[j][i-1]);
336     factorgx = (gap - table.s3x_gap[j-1]) /
337                (table.s3x_gap[j] - table.s3x_gap[j-1]);
338     lc[6]    = curr1x + factorgx * (curr2x - curr1x);
339
340 #ifdef EM_DEBUG_PRINTF
341     printf("lc[6] is %f¥n", lc[6]);
342 #endif
343
344 /* linear interpolation for s3y_table(S3y coil) */
345     i = 0;
346     if (phase < table.s3y_phase[i]) {
347         return EM_ID23_TABLE_S3Y_PHASE_ERROR ;
348     } else {
349         for (i = 1 ; i < 50 ; ++i) {
350             if (phase < table.s3y_phase[i]) {
351                 break ;
352             }
353         }
354     }
355
356     j = 0;
357     if (gap < table.s3y_gap[j]) {
358         return EM_ID23_TABLE_S3Y_GAP_ERROR ;
359     } else {
360         for (j = 1 ; j < 50 ; ++j) {
361             if (gap < table.s3y_gap[j]) {
362                 break ;
363             }
364         }
365     }
366
367     factorpy = (phase - table.s3y_phase[i-1]) /
368                (table.s3y_phase[i] - table.s3y_phase[i-1]);
369     curr1y   = table.s3y_table[j-1][i-1] + factorpy *
370                (table.s3y_table[j-1][i] - table.s3y_table[j-1][i-1]);
371     curr2y   = table.s3y_table[j][i-1]   + factorpy *
372                (table.s3y_table[j][i] - table.s3y_table[j][i-1]);
373     factorgy = (gap - table.s3y_gap[j-1]) /
374                (table.s3y_gap[j] - table.s3y_gap[j-1]);
375     lc[7]    = curr1y + factorgy * (curr2y - curr1y);
376
377 #ifdef EM_DEBUG_PRINTF
378     printf("lc[7] is %f¥n", lc[7]);
379 #endif
380
381 /* linear interpolation for s4x_table(S4x coil) */
382     i = 0;
383     if (phase < table.s4x_phase[i]) {
384         return EM_ID23_TABLE_S4X_PHASE_ERROR ;
```

```
385     } else {
386         for (i = 1 ; i < 50 ; ++i) {
387             if (phase < table.s4x_phase[i]) {
388                 break;
389             }
390         }
391     }
392
393     j = 0;
394     if (gap < table.s4x_gap[j]) {
395         return EM_ID23_TABLE_S4X_GAP_ERROR ;
396     } else {
397         for (j = 1 ; j < 50 ; ++j) {
398             if (gap < table.s4x_gap[j]) {
399                 break;
400             }
401         }
402     }
403
404     factorpx = (phase - table.s4x_phase[i-1]) /
405                (table.s4x_phase[i] - table.s4x_phase[i-1]);
406     curr1x   = table.s4x_table[j-1][i-1] + factorpx *
407                (table.s4x_table[j-1][i] - table.s4x_table[j-1][i-1]);
408     curr2x   = table.s4x_table[j][i-1]   + factorpx *
409                (table.s4x_table[j][i]   - table.s4x_table[j][i-1]);
410     factorgx = (gap - table.s4x_gap[j-1]) /
411                (table.s4x_gap[j] - table.s4x_gap[j-1]);
412     lc[8]    = curr1x + factorgx * (curr2x - curr1x);
413
414 #ifdef EM_DEBUG_PRINTF
415     printf("lc[8] is %f¥n", lc[8]);
416 #endif
417
418 /* linear interpolation for s4y_table(S4y coil) */
419     i = 0;
420     if (phase < table.s4y_phase[i]) {
421         return EM_ID23_TABLE_S4Y_PHASE_ERROR ;
422     } else {
423         for (i = 1 ; i < 50 ; ++i) {
424             if (phase < table.s4y_phase[i]) {
425                 break ;
426             }
427         }
428     }
429
430     j = 0;
431     if (gap < table.s4y_gap[j]) {
432         return EM_ID23_TABLE_S4Y_GAP_ERROR ;
433     } else {
434         for (j = 1 ; j < 50 ; ++j) {
435             if (gap < table.s4y_gap[j]) {
436                 break ;
437             }
438         }
439     }
440
441     factorpy = (phase - table.s4y_phase[i-1]) /
442                (table.s4y_phase[i] - table.s4y_phase[i-1]);
443     curr1y   = table.s4y_table[j-1][i-1] + factorpy *
444                (table.s4y_table[j-1][i] - table.s4y_table[j-1][i-1]);
445     curr2y   = table.s4y_table[j][i-1]   + factorpy *
446                (table.s4y_table[j][i] - table.s4y_table[j][i-1]);
447     factorgy = (gap - table.s4y_gap[j-1]) /
448                (table.s4y_gap[j] - table.s4y_gap[j-1]);
```

```
449     lc[9]    = curr1y + factorgy * (curr2y - curr1y);
450
451 #ifdef EM_DEBUG_PRINTF
452     printf("lc[9] is %f¥n", lc[9]);
453 #endif
454
455     return 0 ;
456 }
```

```
 1 /*(@header_begin)************************************************************
 2 *   (C Function)                                                             *
 3 *       Name = em_id23_lc_table_set                                          *
 4 *       Type = int                                                           *
 5 *   (Purpose)                                                                *
 6 *       store name for correction table                                      *
 7 *   (Input)                                                                  *
 8 *       name of correction table                                             *
 9 *   (Output)                                                                 *
10 *       name of correction table                                             *
11 *   (Return)                                                                 *
12 *       0    : ok                                                            *
13 *       < 0 : fail                                                           *
14 *   (Physical unit)                                                          *
15 *       Pa, V, A, mm, sec                                                    *
16 *   (Relation)                                                               *
17 *                                                                            *
18 *   (History)                                                                *
19 *       Jul-12-1997       F.Funayama /iST  created.                          *
20 *       97-12-17          Hiramatsu     revised                              *
21 *       98-10-14          Hiramatsu     updated                              *
22 ************************************************************(@header_end)*/
23
24 #include "em_id23.h"
25
26 int em_id23_lc_table_set(Em_data in, Em_data conv, Em_data *out)
27 {
28     int err;
29     char fname[EM_ID23_LC_TABLE_SET_FILE_MAX];
30
31 #ifdef  EM_DEBUG_PRINTF
32     printf("Call func name is ¥"em_id23_lc_table_set¥"¥n");
33 #endif
34
35 /* key-in the file name of correction table */
36     em_data_get_char(in, EM_ID23_LC_TABLE_SET_FILE_NAME, fname);
37
38 /* prepare output */
39     em_data_clear(out);
40     em_data_put_char(out, EM_ID23_LC_TABLE_SET_FILE_NAME, fname);
41
42     return 0;
43 }
```

```
/*(@header_begin)*************************************************************
 *  (C Function)                                                             *
 *      Name = em_id23_lc_table_thru                                         *
 *      Type = int                                                           *
 *  (Purpose)                                                                *
 *      read name of correction table                                        *
 *      construct correction table                                           *
 *      put correction table to cellar                                       *
 *  (Input)                                                                  *
 *      name of correction table                                             *
 *  (Output)                                                                 *
 *      none                                                                 *
 *  (Return)                                                                 *
 *      0   : ok                                                             *
 *      < 0 : fail                                                           *
 *  (Physical unit)                                                          *
 *      Pa, V, A, mm, sec                                                    *
 *  (Relation)                                                               *
 *                                                                           *
 *  (History)                                                                *
 *      Jul-12-1997      F.Funayama /iST  created.                           *
 *      98- 1-26         Hiramatsu  revised                                  *
 *      98-10-14         Hiramatsu  updated                                  *
 *************************************************************(@header_end)*/

#include "em_id23.h"

int em_id23_lc_table_thru(Em_data in, Em_data adr, Em_data *out)
{
        int err;
        char fname[EM_ID23_LC_TABLE_THRU_FILE_MAX];
        char f_tmp[EM_ID23_LC_TABLE_THRU_FILE_MAX];
        char p_tmp[EM_ID23_LC_TABLE_THRU_FILE_MAX];
        Em_id23_table table;
        table.set_flg = 0;

#ifdef  EM_DEBUG_PRINTF
        printf("Call func name is ¥"em_id23_lc_table_thru¥"¥n");
#endif

/* get file name of correction table */
        em_data_get_char(in, EM_ID23_LC_TABLE_THRU_FILE_NAME, f_tmp);

/* read config.tbl */
        err = em_data_get_char(adr, EM_ID23_LC_TABLE_THRU_FILE_NAME, p_tmp);
        if (err < 0) {
            return EM_ID23_LC_TABLE_THRU_ERR_ADR_NOT_SET;
        }

/* add file name to directory name */
        sprintf(fname, "%s%s", p_tmp, f_tmp);

#ifdef  EM_DEBUG_PRINTF
        printf("Table file name is ¥"%s¥"¥n", fname);
#endif

/* read file (ASCII-text) and construct correction table */
        err = em_id23_util_table_file_get(fname, &table ) ;
        if (err < 0) {
            return EM_ID23_LC_TABLE_FILE_GET_ERR;
        }

/* PUT table to cellar */
        err = em_id23_util_table_cellar(EM_ID23_PUT_TABLE, &table) ;
```

```
65          if (table.set_flg == 0) {
66              return EM_ID23_LC_CURR_ERR_DATA_NOT_SET ;
67          }
68
69  /* prepare output */
70          em_data_clear(out);
71          em_data_put_int(out, 0, 0);
72
73          return 0;
74
75  }
```

```
 1 /*(@header_begin)*************************************************************
 2 *  (C Function)                                                              *
 3 *      Name = em_id23_lmt_emergency_status_get                               *
 4 *      Type = int                                                            *
 5 *  (Purpose)                                                                 *
 6 *      read device status                                                    *
 7 *  (Input)                                                                   *
 8 *      parameter (config.tbl)                                                *
 9 *  (Output)                                                                  *
10 *      out->data[0].eint : status                                            *
11 *  (Return)                                                                  *
12 *      0   : ok                                                              *
13 *      < 0 : fail                                                            *
14 *  (Physical unit)                                                           *
15 *      Pa, V, A, mm, sec                                                     *
16 *  (Relation)                                                                *
17 *                                                                            *
18 *  (History)                                                                 *
19 *      Jul-16-1997      F.Funayama /iST  created.                            *
20 *      98- 1-11         Hiramatsu    revised                                 *
21 *      98-10-14         Hiramatsu    updated                                 *
22 *************************************************************(@header_end)*/
23
24 #include "em_id23.h"
25
26 int em_id23_lmt_emergency_status_get(Em_data in, Em_data adr, Em_data *out)
27 {
28     int fd, dev_num_1, dev_num_2, di_data[4], err, status = 0 ;
29
30 #ifdef EM_DEBUG_PRINTF
31     printf("Call func name is ¥"em_id23_lmt_emergency_status_get¥"¥n");
32 #endif
33
34 /* get file descriptor */
35     fd = em_get_fd(adr);
36
37 /* read config.tbl */
38     err = em_data_get_int(adr, EM_ID23_EMERGENCY_INPUT_PORT_1, &dev_num_1);
39     if (err < 0) {
40         return EM_ID23_EMERGENCY_ERR_ADR_NOT_SET;
41     }
42
43     err = em_data_get_int(adr, EM_ID23_EMERGENCY_INPUT_PORT_2, &dev_num_2);
44     if (err < 0) {
45         return EM_ID23_EMERGENCY_ERR_ADR_NOT_SET;
46     }
47
48 /* get emergency status */
49 #ifndef EM_DEBUG_SIM
50     err = dev_di_get_diall(fd, di_data);
51     if (err < 0) {
52         return EM_ID23_EMERGENCY_STATUS_ERR;
53     }
54 #else
55     status = 0x1f;
56 #endif
57
58 /* Contact Sensor upper */
59     if (di_data[dev_num_1] & EM_ID23_EMERGENCY_CNT_1_BIT)
60         status |= EM_ID23_EMERGENCY_CNT_1_SET_BIT;
61
62 /* Contact Sensor lower */
63     if (di_data[dev_num_1] & EM_ID23_EMERGENCY_CNT_2_BIT)
64         status |= EM_ID23_EMERGENCY_CNT_2_SET_BIT;
```

```
65
66 /* Emergency on Control Panel */
67    if (di_data[dev_num_2] & EM_ID23_EMERGENCY_PB_1_BIT)
68        status |= EM_ID23_EMERGENCY_PB_1_SET_BIT;
69
70 /* Emergency on Machine */
71    if (di_data[dev_num_2] & EM_ID23_EMERGENCY_PB_2_BIT)
72        status |= EM_ID23_EMERGENCY_PB_2_SET_BIT;
73
74 /* Emergency on Experiment Hall */
75    if (di_data[dev_num_2] & EM_ID23_EMERGENCY_PB_3_BIT)
76        status |= EM_ID23_EMERGENCY_PB_3_SET_BIT;
77
78 /* Local */
79    if (di_data[dev_num_2] & EM_ID23_EMERGENCY_LOCAL_BIT)
80        status |= EM_ID23_EMERGENCY_LOCAL_SET_BIT;
81
82 /* prepare output */
83    em_data_clear(out) ;
84    em_data_put_int(out, 0, status) ;
85
86    return 0;
87 }
```

```
 1 /*(@header_begin)*************************************************************
 2 *  (C Function)                                                              *
 3 *      Name = em_id23_lmt_gap_status_get                                     *
 4 *      Type = int                                                            *
 5 *  (Purpose)                                                                 *
 6 *      read device status                                                    *
 7 *  (Input)                                                                   *
 8 *      parameter (config.tbl)                                                *
 9 *  (Output)                                                                  *
10 *      out->data[0].eint : status                                            *
11 *  (Return)                                                                  *
12 *      0   : ok                                                              *
13 *      < 0 : fail                                                            *
14 *  (Physical unit)                                                           *
15 *      Pa, V, A, mm, sec                                                     *
16 *  (Relation)                                                                *
17 *                                                                            *
18 *  (History)                                                                 *
19 *      Jul-16-1997       F.Funayama /iST  created.                           *
20 *      98- 1-11          Hiramatsu    revised                                *
21 *      98-10-14          Hiramatsu    updated                                *
22 *************************************************************(@header_end)*/
23
24 #include "em_id23.h"
25
26 int em_id23_lmt_gap_status_get(Em_data in, Em_data adr, Em_data *out)
27 {
28         int fd, dev_num_1, dev_num_2, di_data[4], err, status = 0 ;
29
30 #ifdef EM_DEBUG_PRINTF
31         printf("Call func name is ¥"em_id23_lmt_gap_status_get¥"¥n");
32 #endif
33
34 /* get file descriptor */
35         fd = em_get_fd(adr);
36
37 /* read config.tbl */
38         err = em_data_get_int(adr, EM_ID23_GAP_LMT_INPUT_PORT_1, &dev_num_1);
39         if (err < 0) {
40             return EM_ID23_GAP_LMT_ERR_ADR_NOT_SET;
41         }
42
43         err = em_data_get_int(adr, EM_ID23_GAP_LMT_INPUT_PORT_2, &dev_num_2);
44         if (err < 0) {
45             return EM_ID23_GAP_LMT_ERR_ADR_NOT_SET;
46         }
47
48 /* get limit switch data */
49 #ifndef EM_DEBUG_SIM
50         err = dev_di_get_diall(fd, di_data);
51         if (err < 0) {
52             return EM_ID23_GAP_LMT_STATUS_ERR;
53         }
54 #else
55         status = 0xff;
56 #endif
57
58 /* upper upstream open */
59         if (di_data[dev_num_1] & EM_ID23_GAP_LMT_UP_UP_OP_BIT) {
60                 status |= EM_ID23_GAP_LMT_UP_UP_OP_SET_BIT;
61         }
62 /* upper upstream close */
63         if (di_data[dev_num_1] & EM_ID23_GAP_LMT_UP_UP_CL_BIT) {
64                 status |= EM_ID23_GAP_LMT_UP_UP_CL_SET_BIT;
```

```
65          }
66  /* upper downstream open */
67          if (di_data[dev_num_1] & EM_ID23_GAP_LMT_UP_DW_OP_BIT) {
68                  status |= EM_ID23_GAP_LMT_UP_DW_OP_SET_BIT;
69          }
70  /* upper downstream close */
71          if (di_data[dev_num_1] & EM_ID23_GAP_LMT_UP_DW_CL_BIT) {
72                  status |= EM_ID23_GAP_LMT_UP_DW_CL_SET_BIT;
73          }
74  /* lower upstream open  */
75          if (di_data[dev_num_1] & EM_ID23_GAP_LMT_LW_UP_OP_BIT) {
76                  status |= EM_ID23_GAP_LMT_LW_UP_OP_SET_BIT;
77          }
78  /* lower upstream close */
79          if (di_data[dev_num_1] & EM_ID23_GAP_LMT_LW_UP_CL_BIT) {
80                  status |= EM_ID23_GAP_LMT_LW_UP_CL_SET_BIT;
81          }
82  /* lower downstream open */
83          if (di_data[dev_num_1] & EM_ID23_GAP_LMT_LW_DW_OP_BIT) {
84                  status |= EM_ID23_GAP_LMT_LW_DW_OP_SET_BIT;
85          }
86  /* lower downstream close */
87          if (di_data[dev_num_1] & EM_ID23_GAP_LMT_LW_DW_CL_BIT) {
88                  status |= EM_ID23_GAP_LMT_LW_DW_CL_SET_BIT;
89          }
90  /* gap reference */
91          if (di_data[dev_num_2] & EM_ID23_GAP_LMT_HOME_BIT) {
92                  status |= EM_ID23_GAP_LMT_HOME_SET_BIT;
93          }
94
95  /* prepare output */
96          em_data_clear(out);
97          em_data_put_int(out, 0, status);
98
99          return 0;
100 }
```

```
1 /*(@header_begin)*************************************************************
2 *  (C Function)                                                              *
3 *      Name = em_id23_lmt_phase_status_get                                   *
4 *      Type = int                                                            *
5 *  (Purpose)                                                                 *
6 *      read device status                                                    *
7 *  (Input)                                                                   *
8 *      parameter (config.tbl)                                                *
9 *  (Output)                                                                  *
10 *     out->data[0].eint : status                                            *
11 *  (Return)                                                                  *
12 *     0   : ok                                                              *
13 *     < 0 : fail                                                            *
14 *  (Physical unit)                                                           *
15 *     Pa, V, A, mm, sec                                                     *
16 *  (Relation)                                                                *
17 *                                                                           *
18 *  (History)                                                                 *
19 *     Jul-16-1997      F.Funayama /iST  created.                            *
20 *     98- 1-11         Hiramatsu    revised                                 *
21 *     98-10-14         Hiramatsu    updated                                 *
22 *************************************************************(@header_end)*/
23
24 #include "em_id23.h"
25
26 int em_id23_lmt_phase_status_get(Em_data in, Em_data adr, Em_data *out)
27 {
28         int fd, dev_num_1, dev_num_2, di_data[4], err, status = 0 ;
29
30 #ifdef EM_DEBUG_PRINTF
31         printf("Call func name is ¥"em_id23_lmt_phase_status_get¥"¥n");
32 #endif
33
34 /* get file descriptor */
35         fd = em_get_fd(adr);
36
37 /* read config.tbl */
38         err = em_data_get_int(adr, EM_ID23_PHASE_LMT_INPUT_PORT_1, &dev_num_1);
39         if (err < 0) {
40            return EM_ID23_PHASE_LMT_ERR_ADR_NOT_SET;
41         }
42
43         err = em_data_get_int(adr, EM_ID23_PHASE_LMT_INPUT_PORT_2, &dev_num_2);
44         if (err < 0) {
45             return EM_ID23_PHASE_LMT_ERR_ADR_NOT_SET;
46         }
47
48 /* get limit switch data */
49 #ifndef EM_DEBUG_SIM
50         err = dev_di_get_diall(fd, di_data);
51         if (err < 0) {
52             return EM_ID23_PHASE_LMT_STATUS_ERR;
53         }
54 #else
55         status = 0xff;
56 #endif
57
58 /* row1 downstream */
59         if (di_data[dev_num_1] & EM_ID23_PHASE_LMT_UP_1_BIT) {
60                 status |= EM_ID23_PHASE_LMT_UP_1_SET_BIT;
61         }
62 /* row1 upstream */
63         if (di_data[dev_num_1] & EM_ID23_PHASE_LMT_DW_1_BIT) {
64                 status |=  EM_ID23_PHASE_LMT_DW_1_SET_BIT;
```

```
65          }
66  /* row2 downstream */
67          if (di_data[dev_num_1] & EM_ID23_PHASE_LMT_UP_2_BIT) {
68                  status |= EM_ID23_PHASE_LMT_UP_2_SET_BIT;
69          }
70  /* row2 upstream */
71          if (di_data[dev_num_1] & EM_ID23_PHASE_LMT_DW_2_BIT) {
72                  status |= EM_ID23_PHASE_LMT_DW_2_SET_BIT;
73          }
74  /* row3 downstream */
75          if (di_data[dev_num_1] & EM_ID23_PHASE_LMT_UP_3_BIT) {
76                  status |= EM_ID23_PHASE_LMT_UP_3_SET_BIT;
77          }
78  /* row3 upstream */
79          if (di_data[dev_num_1] & EM_ID23_PHASE_LMT_DW_3_BIT) {
80                  status |= EM_ID23_PHASE_LMT_DW_3_SET_BIT;
81          }
82  /* row4 downstream */
83          if (di_data[dev_num_1] & EM_ID23_PHASE_LMT_UP_4_BIT) {
84                  status |= EM_ID23_PHASE_LMT_UP_4_SET_BIT;
85          }
86  /* row4 upstream */
87          if (di_data[dev_num_1] & EM_ID23_PHASE_LMT_DW_4_BIT) {
88                  status |= EM_ID23_PHASE_LMT_DW_4_SET_BIT;
89          }
90  /* upper reference */
91          if (di_data[dev_num_2] & EM_ID23_PHASE_LMT_UP_HOME_BIT) {
92                  status |= EM_ID23_PHASE_LMT_UP_HOME_SET_BIT;
93          }
94  /* lower reference */
95          if (di_data[dev_num_2] & EM_ID23_PHASE_LMT_LW_HOME_BIT) {
96                  status |= EM_ID23_PHASE_LMT_LW_HOME_SET_BIT;
97          }
98
99  /* prepare output */
100         em_data_clear(out);
101         em_data_put_int(out, 0, status);
102
103         return 0;
104 }
```

```
/*(@header_begin)*************************************************************
 *  (C Function)                                                             *
 *      Name = em_id23_pattern_create                                        *
 *      Type = int                                                           *
 *  (Purpose)                                                                *
 *      create EMA process                                                   *
 *  (Input)                                                                  *
 *      none                                                                 *
 *  (Output)                                                                 *
 *      none                                                                 *
 *  (Return)                                                                 *
 *      0   : ok                                                             *
 *      < 0 : fail                                                           *
 *  (Physical unit)                                                          *
 *      Pa, V, A, mm, sec                                                    *
 *  (Relation)                                                               *
 *                                                                           *
 *  (History)                                                                *
 *      Jun-17-1999      yoichi                        created.              *
 *************************************************************(@header_end)*/

#include "em_id23.h"

int em_id23_pattern_create(Em_data in, Em_data adr, Em_data *out)
{
    int err;
    char svoc[256];

#ifdef EM_DEBUG_PRINTF
    printf("Call func name is ¥"em_id23_pattern_create¥"¥n");
#endif

    sprintf(svoc, "s/put/bl_id23_pattern_ema/create");
    err = em_idcom_call_svoc(svoc);
    if (err < 0) {
        return EM_ID23_PATTERN_CREATE_ERR;
    }

/* prepare output */
    em_data_clear(out);
    em_data_put_int(out, 0, 0);

    return 0;
}
```

```
 1 /*(@header_begin)************************************************************
 2 *  (C Function)                                                             *
 3 *      Name = em_id23_pattern_destroy                                       *
 4 *      Type = int                                                           *
 5 *  (Purpose)                                                                *
 6 *      destroy EMA process                                                  *
 7 *  (Input)                                                                  *
 8 *      none                                                                 *
 9 *  (Output)                                                                 *
10 *      none                                                                 *
11 *  (Return)                                                                 *
12 *      0   : ok                                                             *
13 *      < 0 : fail                                                           *
14 *  (Physical unit)                                                          *
15 *      Pa, V, A, mm, sec                                                    *
16 *  (Relation)                                                               *
17 *                                                                           *
18 *  (History)                                                                *
19 *      Jun-17-1999      yoichi                        created.              *
20 ************************************************************(@header_end)*/
21
22 #include "em_id23.h"
23
24 int em_id23_pattern_destroy(Em_data in, Em_data adr, Em_data *out)
25 {
26    int err;
27    char svoc[256];
28
29 #ifdef EM_DEBUG_PRINTF
30    printf("Call func name is ¥"em_id23_pattern_destroy¥"¥n");
31 #endif
32
33    sprintf(svoc, "s/put/bl_id23_pattern_ema/destroy");
34    err = em_idcom_call_svoc(svoc);
35    if (err < 0) {
36        return EM_ID23_PATTERN_DESTROY_ERR;
37    }
38
39 /* prepare output */
40    em_data_clear(out);
41    em_data_put_int(out, 0, 0);
42
43    return 0;
44 }
```

```
 1 /*(@header_begin)*************************************************************
 2 *  (C Function)                                                              *
 3 *      Name = em_id23_pattern_hold_conv_get                                  *
 4 *      Type = int                                                            *
 5 *  (Purpose)                                                                 *
 6 *      convert values of hold-time                                           *
 7 *  (Input)                                                                   *
 8 *      none                                                                  *
 9 *  (Output)                                                                  *
10 *      none                                                                  *
11 *  (Return)                                                                  *
12 *                                                                            *
13 *  (Physical unit)                                                           *
14 *      Pa, V, A, mm, sec                                                     *
15 *  (Relation)                                                                *
16 *                                                                            *
17 *  (History)                                                                 *
18 *      Jun-17-1999      yoichi                        created.               *
19 *************************************************************(@header_end)*/
20
21 #include "em_id23.h"
22
23 int em_id23_pattern_hold_conv_get(Em_data in, Em_data conv, Em_svoc *out)
24 {
25     int err;
26     int hold_r, hold_l;
27
28 #ifdef EM_DEBUG_PRINTF
29     printf("Call func name is ¥"em_id23_pattern_hold_get¥"¥n");
30 #endif
31
32     err = em_data_get_int(in, 0, &hold_r);
33     if (err < 0) {
34         return EM_ID23_PATTERN_HOLD_CONV_GET_ERR;
35     }
36
37     err = em_data_get_int(in, 1, &hold_l);
38     if (err < 0) {
39         return EM_ID23_PATTERN_HOLD_CONV_GET_ERR;
40     }
41
42 /* prepare output */
43     em_data_clear(out);
44     sprintf(out->svoc, "%dmsec_%dmsec", hold_r, hold_l);
45
46     return 0;
47 }
```

```
 1 /*(@header_begin)************************************************************
 2 *  (C Function)                                                             *
 3 *      Name = em_id23_pattern_hold_get                                      *
 4 *      Type = int                                                           *
 5 *  (Purpose)                                                                *
 6 *      read values of hold-time                                             *
 7 *  (Input)                                                                  *
 8 *      none                                                                 *
 9 *  (Output)                                                                 *
10 *      none                                                                 *
11 *  (Return)                                                                 *
12 *                                                                           *
13 *  (Physical unit)                                                          *
14 *      Pa, V, A, mm, sec                                                    *
15 *  (Relation)                                                               *
16 *                                                                           *
17 *  (History)                                                                *
18 *      Jun-17-1999      yoichi                        created.              *
19 ************************************************************(@header_end)*/
20
21 #include "em_id23.h"
22
23 int em_id23_pattern_hold_get(Em_data in, Em_data adr, Em_data *out)
24 {
25    int err;
26    int hold_r, hold_l;
27    double holdtime;
28    char unit[20];
29
30 #ifdef EM_DEBUG_PRINTF
31    printf("Call func name is ¥"em_id23_pattern_hold_get¥"¥n");
32 #endif
33
34    err = em_idcom_func_svoc("s/get/bl_id23_pattern_ema/hold_r",
35                             &holdtime, unit);
36    if (err < 0) {
37        return EM_ID23_PATTERN_HOLD_GET_ERR;
38    }
39    hold_r = (int)holdtime;
40
41    err = em_idcom_func_svoc("s/get/bl_id23_pattern_ema/hold_l",
42                             &holdtime, unit);
43    if (err < 0) {
44        return EM_ID23_PATTERN_HOLD_GET_ERR;
45    }
46    hold_l = (int)holdtime;
47
48 /* prepare output */
49    em_data_clear(out);
50    em_data_put_int(out, 0, hold_r);
51    em_data_put_int(out, 1, hold_l);
52
53    return 0;
54 }
```

```
1 /*(@header_begin)*************************************************************
2 *  (C Function)                                                              *
3 *      Name = em_id23_pattern_hold_put                                       *
4 *      Type = int                                                            *
5 *  (Purpose)                                                                 *
6 *      set values of hold-time for right and left polarization               *
7 *  (Input)                                                                   *
8 *      hold_r  and  hold_l                                                   *
9 *  (Output)                                                                  *
10 *      none                                                                 *
11 *  (Return)                                                                 *
12 *                                                                           *
13 *  (Physical unit)                                                          *
14 *      Pa, V, A, mm, sec                                                    *
15 *  (Relation)                                                               *
16 *                                                                           *
17 *  (History)                                                                *
18 *      Jun-17-1999      yoichi                        created.              *
19 *************************************************************(@header_end)*/
20
21 #include "em_id23.h"
22
23 int em_id23_pattern_hold_put(Em_data in, Em_data adr, Em_data *out)
24 {
25    int err;
26    int hold_r, hold_l;
27    char svoc[256];              /* SVOC recursive call */
28
29 #ifdef EM_DEBUG_PRINTF
30    printf("Call func name is ¥"em_id23_pattern_hold_put¥"¥n");
31 #endif
32
33    err = em_data_get_int(in, 0, &hold_r);
34    if (err < 0) {
35       return EM_ID23_PATTERN_HOLD_PUT_ERR_HOLD_R;
36    }
37
38    err = em_data_get_int(in, 1, &hold_l);
39    if (err < 0) {
40       return EM_ID23_PATTERN_HOLD_PUT_ERR_HOLD_L;
41    }
42
43    sprintf(svoc,"s/put/bl_id23_pattern_ema/hold_r_%dmsec", hold_r);
44    err = em_idcom_call_svoc(svoc);
45    if (err < 0) {
46       return EM_ID23_PATTERN_HOLD_PUT_ERR_HOLD_R_PUT;
47    }
48
49    sprintf(svoc,"s/put/bl_id23_pattern_ema/hold_l_%dmsec", hold_l);
50    err = em_idcom_call_svoc(svoc);
51    if (err < 0) {
52       return EM_ID23_PATTERN_HOLD_PUT_ERR_HOLD_L_PUT;
53    }
54
55 /* prepare output */
56    em_data_clear(out);
57    em_data_put_int(out, 0, 0);
58
59    return 0;
60 }
```

```
 1 /*(@header_begin)*************************************************************
 2 *  (C Function)                                                              *
 3 *      Name = em_id23_pattern_phase_conv_get                                 *
 4 *      Type = int                                                            *
 5 *  (Purpose)                                                                 *
 6 *      convert values of phase-shift                                         *
 7 *  (Input)                                                                   *
 8 *      none                                                                  *
 9 *  (Output)                                                                  *
10 *      none                                                                  *
11 *  (Return)                                                                  *
12 *                                                                            *
13 *  (Physical unit)                                                           *
14 *      Pa, V, A, mm, sec                                                     *
15 *  (Relation)                                                                *
16 *                                                                            *
17 *  (History)                                                                 *
18 *      Jun-17-1999      yoichi                      created.                 *
19 *************************************************************(@header_end)*/
20
21 #include "em_id23.h"
22
23 int em_id23_pattern_phase_conv_get(Em_data in, Em_data conv, Em_svoc *out)
24 {
25    int err;
26    double phase_r, phase_I;
27
28 #ifdef EM_DEBUG_PRINTF
29    printf("Call func name is ¥"em_id23_pattern_phase_conv_get¥"¥n");
30 #endif
31
32    err = em_data_get_double(in, 0, &phase_r);
33    if (err < 0) {
34        return EM_ID23_PATTERN_PHASE_CONV_GET_ERR;
35    }
36
37
38    err = em_data_get_double(in, 1, &phase_I);
39    if (err < 0) {
40        return EM_ID23_PATTERN_PHASE_CONV_GET_ERR;
41    }
42
43 /* prepare output */
44    em_data_clear(out);
45    sprintf(out->svoc, "%.3fmm_%.3fmm", phase_r, phase_I);
46
47    return 0;
48 }
```

```
 1 /*(@header_begin)*************************************************************
 2 *  (C Function)                                                              *
 3 *      Name = em_id23_pattern_phase_get                                      *
 4 *      Type = int                                                            *
 5 *  (Purpose)                                                                 *
 6 *      read values of phase-shift                                            *
 7 *  (Input)                                                                   *
 8 *      none                                                                  *
 9 *  (Output)                                                                  *
10 *      none                                                                  *
11 *  (Return)                                                                  *
12 *                                                                            *
13 *  (Physical unit)                                                           *
14 *      Pa, V, A, mm, sec                                                     *
15 *  (Relation)                                                                *
16 *                                                                            *
17 *  (History)                                                                 *
18 *      Jun-17-1999      yoichi                        created.               *
19 *************************************************************(@header_end)*/
20
21 #include "em_id23.h"
22
23 int em_id23_pattern_phase_get(Em_data in, Em_data adr, Em_data *out)
24 {
25    int err;
26    double phase_r, phase_i;
27    char unit[20];
28
29 #ifdef EM_DEBUG_PRINTF
30    printf("Call func name is ¥"em_id23_pattern_phase_get¥"¥n");
31 #endif
32
33    err = em_idcom_func_svoc("s/get/bl_id23_pattern_ema/phase_r",
34                             &phase_r, unit);
35    if (err < 0) {
36        return EM_ID23_PATTERN_PHASE_GET_ERR;
37    }
38
39    err = em_idcom_func_svoc("s/get/bl_id23_pattern_ema/phase_i",
40                             &phase_i, unit);
41    if (err < 0) {
42        return EM_ID23_PATTERN_PHASE_GET_ERR;
43    }
44
45
46 /* prepare output */
47    em_data_clear(out);
48    em_data_put_double(out, 0, phase_r);
49    em_data_put_double(out, 1, phase_i);
50
51    return 0;
52 }
```

```
/*(@header_begin)*************************************************************
 *  (C Function)                                                             *
 *      Name = em_id23_pattern_phase_put                                     *
 *      Type = int                                                           *
 *  (Purpose)                                                                *
 *      set values of phase_shift for right and left polarization            *
 *  (Input)                                                                  *
 *      phase_r  and  phase_l                                                *
 *  (Output)                                                                 *
 *      none                                                                 *
 *  (Return)                                                                 *
 *                                                                           *
 *  (Physical unit)                                                          *
 *      Pa, V, A, mm, sec                                                    *
 *  (Relation)                                                               *
 *                                                                           *
 *  (History)                                                                *
 *      Jun-17-1999      yoichi                        created.              *
 *****************************************************************(@header_end)*/

#include "em_id23.h"

int em_id23_pattern_phase_put(Em_data in, Em_data adr, Em_data *out)
{
   int err;
   double phase_r, phase_l;
   char svoc[256];              /* SVOC recursive call */

#ifdef EM_DEBUG_PRINTF
   printf("Call func name is ¥"em_id23_pattern_phase_put¥"¥n");
#endif

   err = em_data_get_double(in, 0, &phase_r);
   if (err < 0) {
      return EM_ID23_PATTERN_PHASE_PUT_ERR_PHASE_R;
   }

   err = em_data_get_double(in, 1, &phase_l);
   if (err < 0) {
      return EM_ID23_PATTERN_PHASE_PUT_ERR_PHASE_L;
   }

   sprintf(svoc,"s/put/bl_id23_pattern_ema/phase_r_%fmm", phase_r);
   err = em_idcom_call_svoc(svoc);
   if (err < 0) {
      return EM_ID23_PATTERN_PHASE_PUT_ERR_PHASE_R_PUT;
   }

   sprintf(svoc,"s/put/bl_id23_pattern_ema/phase_l_%fmm", phase_l);
   err = em_idcom_call_svoc(svoc);
   if (err < 0) {
      return EM_ID23_PATTERN_PHASE_PUT_ERR_PHASE_L_PUT;
   }

/* prepare output */
   em_data_clear(out);
   em_data_put_int(out, 0, 0);

   return 0;
}
```

```
 1 /*(@header_begin)*************************************************************
 2 *  (C Function)                                                              *
 3 *      Name = em_id23_pattern_start                                          *
 4 *      Type = int                                                            *
 5 *  (Purpose)                                                                 *
 6 *      start EMA process                                                     *
 7 *  (Input)                                                                   *
 8 *      none                                                                  *
 9 *  (Output)                                                                  *
10 *      none                                                                  *
11 *  (Return)                                                                  *
12 *      0   : ok                                                              *
13 *      < 0 : fail                                                            *
14 *  (Physical unit)                                                           *
15 *      Pa, V, A, mm, sec                                                     *
16 *  (Relation)                                                                *
17 *                                                                            *
18 *  (History)                                                                 *
19 *      Jun-17-1999      yoichi                      created.                 *
20 *************************************************************(@header_end)*/
21
22 #include "em_id23.h"
23
24 int em_id23_pattern_start(Em_data in, Em_data adr, Em_data *out)
25 {
26    int err;
27    char svoc[256];
28
29 #ifdef EM_DEBUG_PRINTF
30    printf("Call func name is ¥"em_id23_pattern_start¥"¥n");
31 #endif
32
33    sprintf(svoc, "s/put/bl_id23_pattern_ema/start");
34    err = em_idcom_call_svoc(svoc);
35    if (err < 0) {
36       return EM_ID23_PATTERN_START_ERR;
37    }
38
39 /* prepare output */
40    em_data_clear(out);
41    em_data_put_int(out, 0, 0);
42
43    return 0;
44 }
```

```
 1 /*(@header_begin)************************************************************
 2 *  (C Function)                                                             *
 3 *      Name = em_id23_pattern_status_conv_get                               *
 4 *      Type = int                                                           *
 5 *  (Purpose)                                                                *
 6 *      convert EMA status(run/pause)                                        *
 7 *  (Input)                                                                  *
 8 *      none                                                                 *
 9 *  (Output)                                                                 *
10 *      none                                                                 *
11 *  (Return)                                                                 *
12 *                                                                           *
13 *  (Physical unit)                                                          *
14 *      Pa, V, A, mm, sec                                                    *
15 *  (Relation)                                                               *
16 *                                                                           *
17 *  (History)                                                                *
18 *      Jun-17-1999      yoichi                        created.              *
19 ************************************************************(@header_end)*/
20
21 #include "em_id23.h"
22
23 int em_id23_pattern_status_conv_get(Em_data in, Em_data conv, Em_svoc *out)
24 {
25     int err;
26     char status[20];
27
28 #ifdef EM_DEBUG_PRINTF
29     printf("Call func name is ¥"em_id23_pattern_status_conv_get¥"¥n");
30 #endif
31
32     err = em_data_get_char(in, 0, status);
33     if (err < 0) {
34         return EM_ID23_PATTERN_STATUS_CONV_GET_ERR;
35     }
36
37 /* prepare output */
38     em_data_clear(out);
39     sprintf(out->svoc, "%s", status);
40
41     return 0;
42 }
```

```
/*(@header_begin)*************************************************************
 *  (C Function)                                                               *
 *      Name = em_id23_pattern_status_get                                      *
 *      Type = int                                                             *
 *  (Purpose)                                                                  *
 *      read EMA status (run / pause)                                          *
 *  (Input)                                                                    *
 *      none                                                                   *
 *  (Output)                                                                   *
 *      none                                                                   *
 *  (Return)                                                                   *
 *                                                                             *
 *  (Physical unit)                                                            *
 *      Pa, V, A, mm, sec                                                      *
 *  (Relation)                                                                 *
 *                                                                             *
 *  (History)                                                                  *
 *      Jun-17-1999      Hiramatsu      created                                *
 *************************************************************(@header_end)*/

#include "em_id23.h"

int em_id23_pattern_status_get(Em_data in, Em_data adr, Em_data *out)
{
    int err, decision;
    char status[20], svoc[256], unit[20];
    char s[256], v[256], o[256], c[256];
    double dummy;
    Em_svoc *ret, command;

#ifdef EM_DEBUG_PRINTF
    printf("Call func name is ¥"em_id23_pattern_status_get¥"¥n");
#endif

/* initialize */
    em_data_clear(out);

    strcpy(command.svoc,"s/get/bl_id23_pattern_ema/status");
    ret = em_call_svoc_1(command);
    if (ret->error == -302070057) {
        strcpy(status,"inactive");
        em_data_put_char(out, 0, status);
        return 0;
    }

    err = em_idcom_util_divide_svoc(ret->svoc, s, v, o, c);
    if (err < 0) {
        return err;
    }

    err= sscanf(c, "%s", status);
    if (err < 0) {
        return EM_ID23_PATTERN_STATUS_GET_ERR_NO_VALUE_IN_C;
    }

    decision = strcmp("pause", status);

    if (decision < 0) {
       strcpy(status, "active");
    } else {
       err = em_idcom_func_svoc("s/get/bl_id23_pattern_ema/phase_r", &dummy, unit
);
       if (err < 0) {
           strcpy(status, "not_ready");
```

```
65        }
66
67        err = em_idcom_func_svoc("s/get/bl_id23_pattern_ema/phase_l", &dummy, unit
68 );
69        if (err < 0) {
70            strcpy(status, "not_ready");
71        }
72
73        err = em_idcom_func_svoc("s/get/bl_id23_pattern_ema/hold_r", &dummy, unit)
74 ;
75        if (err < 0) {
76            strcpy(status, "not_ready");
77        }
78
79        err = em_idcom_func_svoc("s/get/bl_id23_pattern_ema/hold_l", &dummy, unit)
80 ;
81        if (err < 0) {
82            strcpy(status, "not_ready");
83        } else {
84            strcpy(status, "ready");
85        }
86    }
87
88 /* prepare output */
89    em_data_clear(out);
90    em_data_put_char(out, 0, status);
91
92    return 0;
93 }
```

```
 1 /*(@header_begin)*************************************************************
 2 *  (C Function)                                                              *
 3 *      Name = em_id23_pattern_stop                                           *
 4 *      Type = int                                                            *
 5 *  (Purpose)                                                                 *
 6 *      stop EMA process                                                      *
 7 *  (Input)                                                                   *
 8 *      none                                                                  *
 9 *  (Output)                                                                  *
10 *      none                                                                  *
11 *  (Return)                                                                  *
12 *      0   : ok                                                              *
13 *      < 0 : fail                                                            *
14 *  (Physical unit)                                                           *
15 *      Pa, V, A, mm, sec                                                     *
16 *  (Relation)                                                                *
17 *                                                                            *
18 *  (History)                                                                 *
19 *      Jun-17-1999      yoichi                        created.               *
20 *************************************************************(@header_end)*/
21
22 #include "em_id23.h"
23
24 int em_id23_pattern_stop(Em_data in, Em_data adr, Em_data *out)
25 {
26    int err;
27    char svoc[256];
28
29 #ifdef EM_DEBUG_PRINTF
30    printf("Call func name is ¥"em_id23_pattern_stop¥"¥n");
31 #endif
32
33    sprintf(svoc,"s/put/bl_id23_pattern_ema/stop");
34    err = em_idcom_call_svoc(svoc);
35    if (err < 0) {
36        return EM_ID23_PATTERN_STOP_ERR;
37    }
38
39 /* prepare output */
40    em_data_clear(out);
41    em_data_put_int(out, 0, 0);
42
43    return 0;
44 }
```

```
 1 /*(@header_begin)*************************************************************
 2 *  (C Function)                                                              *
 3 *      Name = em_id23_phase_brake_off_put                                    *
 4 *      Type = int                                                            *
 5 *  (Purpose)                                                                 *
 6 *      put brake-off                                                         *
 7 *  (Input)                                                                   *
 8 *      parameter (config.tbl)                                                *
 9 *  (Output)                                                                  *
10 *      none                                                                  *
11 *  (Return)                                                                  *
12 *      0   : ok                                                              *
13 *      < 0 : fail                                                            *
14 *  (Physical unit)                                                           *
15 *      Pa, V, A, mm, sec                                                     *
16 *  (Relation)                                                                *
17 *                                                                            *
18 *  (History)                                                                 *
19 *      Jul-11-1997      F.Funayama /iST  created.                            *
20 *      98- 1-11         Hiramatsu        revised                             *
21 *      98-10-14         Hiramatsu        updated                             *
22 *      99-06-07         Hiramatsu        corrected variable name             *
23 *************************************************************(@header_end)*/
24
25 #include "em_id23.h"
26
27 int em_id23_phase_brake_off_put(Em_data in, Em_data adr, Em_data *out)
28 {
29     int fd, err;
30     int dev_num_0, dev_num_1;    /* bit number for brake   */
31     int do_data;                 /* brake-off(=0)          */
32
33 #ifdef  EM_DEBUG_PRINTF
34     printf("Call func name is ¥"em_id23_phase_brake_off_put¥"¥n");
35 #endif
36
37 /* get file-descriptor */
38     fd = em_get_fd(adr);
39
40 /* read from config.tbl */
41     err = em_data_get_int(adr, EM_ID23_PHASE_BRAKE_U_ON_BIT, &dev_num_0);
42     if (err < 0) {
43         return EM_ID23_PHASE_BRAKE_U_ON_BIT_ERR;
44     }
45     err = em_data_get_int(adr, EM_ID23_PHASE_BRAKE_L_ON_BIT, &dev_num_1);
46     if (err < 0) {
47        return EM_ID23_PHASE_BRAKE_L_ON_BIT_ERR;
48     }
49
50 #ifndef EM_DEBUG_SIM
51     do_data = 0;          /* put brake-off */
52
53     err = dev_do_put_dobit(fd, dev_num_0, &do_data);
54     if (err < 0) {
55         return EM_ID23_PHASE_BRAKE_OFF_PUT_ERR;
56     }
57     err = dev_do_put_dobit(fd, dev_num_1, &do_data);
58     if (err < 0) {
59         return EM_ID23_PHASE_BRAKE_OFF_PUT_ERR;
60     }
61 #endif
62
63 /* prepare output */
64     em_data_clear(out);
65     em_data_put_int(out, 0, 0);
66
67     return 0;
68 }
```

```
/*(@header_begin)************************************************************
 *  (C Function)                                                              *
 *      Name = em_id23_phase_brake_on_put                                     *
 *      Type = int                                                            *
 *  (Purpose)                                                                 *
 *      put brake-on                                                          *
 *  (Input)                                                                   *
 *      parameter (config.tbl)                                                *
 *  (Output)                                                                  *
 *      none                                                                  *
 *  (Return)                                                                  *
 *      0   : ok                                                              *
 *      < 0 : fail                                                            *
 *  (Physical unit)                                                           *
 *      Pa, V, A, mm, sec                                                     *
 *  (Relation)                                                                *
 *                                                                            *
 *  (History)                                                                 *
 *      Jul-11-1997      F.Funayama /iST  created.                            *
 *      98- 1-11         Hiramatsu        revised                             *
 *      98-10-14         Hiramatsu        updated                             *
 *      99-06-07         Hiramatsu        corrected variable name             *
 ************************************************************(@header_end)*/

#include "em_id23.h"

int em_id23_phase_brake_on_put(Em_data in, Em_data adr, Em_data *out)
{
    int fd, err;
    int dev_num_0, dev_num_1;   /* bit number for brake   */
    int do_data;                /* brake-on(=1)           */

#ifdef  EM_DEBUG_PRINTF
    printf("Call func name is ¥"em_id23_phase_brake_on_put¥"¥n");
#endif

/* get file descriptor */
    fd = em_get_fd(adr);

/* read from config.tbl */
    err = em_data_get_int(adr, EM_ID23_PHASE_BRAKE_U_ON_BIT, &dev_num_0);
    if (err < 0) {
        return EM_ID23_PHASE_BRAKE_U_ON_BIT_ERR;
    }
    err = em_data_get_int(adr, EM_ID23_PHASE_BRAKE_L_ON_BIT, &dev_num_1);
    if (err < 0) {
        return EM_ID23_PHASE_BRAKE_L_ON_BIT_ERR;
    }


#ifndef EM_DEBUG_SIM
    do_data = 1;                /* put brake-on  */
    err = dev_do_put_dobit(fd, dev_num_0, &do_data);
    if (err < 0) {
       return EM_ID23_PHASE_BRAKE_ON_PUT_ERR;
    }
    err = dev_do_put_dobit(fd, dev_num_1, &do_data);
    if (err < 0) {
       return EM_ID23_PHASE_BRAKE_ON_PUT_ERR ;
    }
#endif

/* prepare output */
    em_data_clear(out);
    em_data_put_int(out, 0, 0);

    return 0;
}
```

```
/*(@header_begin)************************************************************
 *  (C Function)                                                              *
 *      Name = em_id23_phase_conv_put                                         *
 *      Type = int                                                            *
 *  (Purpose)                                                                 *
 *      convert phase-shift(mm) to number of pulses                           *
 *  (Input)                                                                   *
 *      parameter (config.tbl)                                                *
 *  (Output)                                                                  *
 *      pulse number                                                          *
 *  (Return)                                                                  *
 *      0   : ok                                                              *
 *      < 0 : fail                                                            *
 *  (Physical unit)                                                           *
 *      Pa, V, A, mm, sec                                                     *
 *  (Relation)                                                                *
 *                                                                            *
 *  (History)                                                                 *
 *      Jul-10-1997      F.Funayama /iST  created.                            *
 *      98- 1-19         Hiramatsu    revised                                 *
 *      99-01-09         Hiramatsu    updated                                 *
 ************************************************************(@header_end)*/

#include "em_id23.h"

int em_id23_phase_conv_put(Em_data in, Em_data conv, Em_data *out)
{
    int err, move_kind, pulse_num_u, pulse_num_l;
    double phase_diff, phase_target, phase_move_num;
    double phase_up, phase_lo, y_data;
    double phase_move_num_upp, phase_move_num_low;
    char unit[30];
    Em_id23_position pos;
    pos.set_flg = 0;

#ifdef EM_DEBUG_PRINTF
    printf("Call func name is ¥"em_id23_phase_conv_put¥"¥n");
#endif

/* get phase-shift(D) */
    err = em_data_get_double(in, EM_ID23_PHASE_CONV_MOVE_DATA, &phase_diff);
    if (err < 0) {
        return EM_ID23_PHASE_CONV_MOVE_DATA_NOT_SET;
    }

/* read config.tbl */
    err = em_data_get_int(conv, EM_ID23_PHASE_CONV_MOVE_KIND, &move_kind);
    if (err < 0) {
        return EM_ID23_PHASE_CONV_ERR_CNV_NOT_SET;
    }

#ifndef EM_DEBUG_SIM
/* read present upper-phase from linear scale */
    err = em_idcom_func_svoc("s/get/bl_id23_phase_upper/position",
                             &phase_up, unit);
    if (err < 0) {
        return EM_ID23_PHASE_U_PUT_ERR;
    }
/* read present lower-phase from linear scale */
    err = em_idcom_func_svoc("s/get/bl_id23_phase_lower/position",
                             &phase_lo, unit);
    if (err < 0) {
        return EM_ID23_PHASE_L_PUT_ERR;
    }
```

```
#else
/* PUT position to cellar */
    pos.gapha = phase_diff / 2;
    em_id23_util_position_cellar(EM_ID23_PUT_POSITION, 1, &pos);
    if (pos.set_flg == 0) {
        return EM_ID23_UTIL_POSITION_CELLAR_ERR;
    }

    pos.gapha = (-1) * phase_diff / 2;
    em_id23_util_position_cellar(EM_ID23_PUT_POSITION, 2, &pos);
    if (pos.set_flg == 0) {
        return EM_ID23_UTIL_POSITION_CELLAR_ERR;
    }

/* GET position from cellar */
    em_id23_util_position_cellar(EM_ID23_GET_POSITION, 1, &pos);
    if (pos.set_flg == 0) {
       return EM_ID23_UTIL_POSITION_CELLAR_ERR;
    }
    phase_up = pos.gapha;

    em_id23_util_position_cellar(EM_ID23_GET_POSITION, 2, &pos);
    if (pos.set_flg == 0) {
       return EM_ID23_UTIL_POSITION_CELLAR_ERR;
    }
    phase_lo = pos.gapha;
#endif

    switch(move_kind) {
       case EM_ID23_PHASE_CONV_MOVE_BOTH:
          phase_target = phase_diff / 2;
          phase_move_num_upp = phase_target - phase_up;
          y_data= EM_ID23_PHASE_MOVE_CONV_COEFF * phase_move_num_upp;
          pulse_num_u = (int)y_data;
          phase_move_num_low = - phase_target - phase_lo;
          y_data= EM_ID23_PHASE_MOVE_CONV_COEFF * phase_move_num_low;
          pulse_num_l = (int)y_data;
          break;
       case EM_ID23_PHASE_CONV_MOVE_UPPER:
          phase_target = phase_diff;
          phase_move_num = phase_target - phase_up;
          y_data= EM_ID23_PHASE_MOVE_CONV_COEFF * phase_move_num;
          pulse_num_u = (int)y_data;
          break;
       case EM_ID23_PHASE_CONV_MOVE_LOWER:
          phase_target = phase_diff;
          phase_move_num = phase_target - phase_lo;
          y_data= EM_ID23_PHASE_MOVE_CONV_COEFF * phase_move_num;
          pulse_num_l = (int)y_data;
          break;
       default:
          phase_move_num = 0;
          return EM_ID23_PHASE_CONV_PUT_ERR_UNSUPPORTED_MODE;
    }

/* prepare output */
    em_data_clear(out);

    err = em_data_put_int(out, EM_ID23_PHASE_MOVE_U_PULS_NUM, pulse_num_u);
    if (err < 0) {
        return EM_ID23_PHASE_MOVE_U_PULS_NUM_ERR;
    }

    err = em_data_put_int(out, EM_ID23_PHASE_MOVE_L_PULS_NUM, pulse_num_l);
```

```
129     if (err < 0) {
130         return EM_ID23_PHASE_MOVE_L_PULS_NUM_ERR;
131     }
132
133     return 0;
134 }
```

```
/*(@header_begin)************************************************************
 *  (C Function)                                                             *
 *      Name = em_id23_phase_init_put                                        *
 *      Type = int                                                           *
 *  (Purpose)                                                                *
 *      obtain phase reference position (max 10 trials)                      *
 *  (Input)                                                                  *
 *      none                                                                 *
 *  (Output)                                                                 *
 *      none                                                                 *
 *  (Return)                                                                 *
 *      0   : ok                                                             *
 *      < 0 : fail                                                           *
 *  (Physical unit)                                                          *
 *      Pa, V, A, mm, sec                                                    *
 *  (Relation)                                                               *
 *                                                                           *
 *  (History)                                                                *
 *      99-06-16   Hiramatsu   created                                       *
 *      99-09-22   Hiramatsu   revised for phase_upper and phase_lower       *
 ************************************************************(@header_end)*/

#include "em_id23.h"

int em_id23_phase_init_put(Em_data in, Em_data adr, Em_data *out)
{
    int fd_0, fd_1;
    int err, count = 0, diff, status;
    int cw_kind = -1;               /* move direction (-1:beam upstream) */
    int phase_which;                /* axis of upper(1) or lower(2)      */
    int repetition = 0;             /* number of repetition              */
    int phase_kind;                 /* upper-phase(0) or lower-phase(1)  */
    int time_num;                   /* delay time [microsec]             */
    int dev_num;                    /* DI device                         */
    int di_data[4];                 /* DI status data                    */
    int pulse_num;                  /* number of pulse necessary to move */
    unsigned short v_max = 100;     /* maximum velocity [pps]            */
    unsigned short v_min = 1;       /* minimum velocity [pps]            */
    unsigned short acc_cycle = 100; /* acceleration [microsec]           */
    char svoc[256];                 /* SVOC recursive call               */
    char unit[20];                  /* unit [mm]                         */
    double difference;              /* difference [mm]                   */
    double present_phase;           /* absolute phase [mm]               */
    double phase[50];               /* phase reference [mm]              */

#ifdef EM_DEBUG_PRINTF
    printf("Call func name is ¥"em_id23_phase_init_put¥"¥n");
#endif

/* get file descriptor */
    fd_0 = em_get_fd_multi(adr, 0);       /* PTG0350 device  */
    fd_1 = em_get_fd_multi(adr, 3);       /* DI device       */

/* read config.tbl */
    err = em_data_get_int(adr, EM_ID23_PHASE_INIT_KIND, &phase_kind);
    if (err < 0) {
       return EM_ID23_PHASE_INIT_ERR_ADR_NOT_SET;
    }

    err = em_data_get_int(adr, EM_ID23_PHASE_INIT_DELAY_TIME, &time_num);
    if (err < 0) {
        return EM_ID23_PHASE_INIT_ERR_ADR_TIME_NOT_SET;
    }

```

```
65      err = em_data_get_int(adr, EM_ID23_PHASE_INIT_INPUT_PORT, &dev_num);
66      if (err < 0) {
67          return EM_ID23_PHASE_INIT_ERR_ADR_PORT_NOT_SET;
68      }
69
70  /* read first phase-reference */
71      if (phase_kind == EM_ID23_PHASE_UPPER) {
72         err = em_idcom_func_svoc("s/get/bl_id23_phase_upper/position",
73                                  &present_phase, unit);
74         if (err < 0) {
75             return EM_ID23_PHASE_INIT_ERR_FIRST_REFERENCE_UP;
76         }
77      } else {
78         err = em_idcom_func_svoc("s/get/bl_id23_phase_lower/position",
79                                  &present_phase, unit);
80         if (err < 0) {
81             return EM_ID23_PHASE_INIT_ERR_FIRST_REFERENCE_LOW;
82         }
83      }
84
85      phase[0] = present_phase;
86  printf("phase[0] is %f ¥n", phase[0]);
87
88      do {            /* move to beam downstream */
89          pulse_num = (int)(EM_ID23_PHASE_MOVE_CONV_COEFF * 5.0);
90          if (phase_kind == EM_ID23_PHASE_UPPER) {
91              dev_ptg0350_move(fd_0, pulse_num, 0, acc_cycle, v_max,
92                               v_min, 0, EM_ID23_TIME_OUT_NUM);
93          } else {
94              dev_ptg0350_move(fd_0, 0, pulse_num, acc_cycle, v_max,
95                               v_min, 0, EM_ID23_TIME_OUT_NUM);
96          }
97
98          sleep(3);
99
100 /* move to beam upstream */
101         pulse_num = (int)(EM_ID23_PHASE_MOVE_CONV_COEFF * (-1) * 4.8);
102         if (phase_kind == EM_ID23_PHASE_UPPER) {
103             dev_ptg0350_move(fd_0, pulse_num, 0, acc_cycle, v_max,
104                              v_min, 0, EM_ID23_TIME_OUT_NUM);
105         } else {
106             dev_ptg0350_move(fd_0, 0, pulse_num, acc_cycle, v_max,
107                              v_min, 0, EM_ID23_TIME_OUT_NUM);
108         }
109
110         sleep(1);
111
112 #ifndef EM_DEBUG_SIM
113         do {        /* move 1 pulse by 1 pulse */
114             dev_ptg0350_move_one(fd_0, phase_which, cw_kind);
115             usleep(time_num);
116 /* read phase-reference */
117             err = dev_di_get_diall(fd_1, di_data);
118             if (err < 0) {
119                 return EM_ID23_PHASE_INIT_ERR_STATUS_GET;
120             }
121             status = (di_data[dev_num] &
122                              EM_ID23_PHASE_LMT_UP_HOME_BIT) >> 7;
123         } while (status == 0x0);
124 #endif
125
126         sleep(3);
127
128 /* read second phase-reference */
```

```
129         if (phase_kind == EM_ID23_PHASE_UPPER) {
130            err = em_idcom_func_svoc("s/get/bl_id23_phase_upper/position",
131                                     &present_phase, unit);
132            if (err < 0) {
133                return EM_ID23_PHASE_INIT_ERR_SECOND_REFERENCE_UP;
134            }
135         } else {
136            err = em_idcom_func_svoc("s/get/bl_id23_phase_lower/position",
137                                     &present_phase, unit);
138            if (err < 0) {
139                return EM_ID23_PHASE_INIT_ERR_SECOND_REFERENCE_LOW;
140            }
141         }
142
143         count += 1;
144
145         if (count > 50) {
146             return EM_ID23_PHASE_INIT_PUT_ERR_COUNT_OVER;
147         }
148
149         phase[count] = present_phase;
150
151 printf("phase[count] is %d  %f ¥n", count, phase[count]);
152 /* calculate difference from previous phase-reference */
153         difference = phase[count] - phase[count - 1];
154         diff = (int)(difference * 1000);
155
156 /* repete movement until meet 3 consecutive rdgs */
157         if (abs(diff) < 50) {
158             repetition += 1;
159         } else {
160             repetition = 0;
161         }
162         sleep(3);
163     } while (repetition < 3);
164
165 /* preset linear-scale and rotary-encoder
166     if (phase_kind == EM_ID23_PHASE_UPPER) {
167         sprintf(svoc, "s/put/bl_id23_phase_upper/preset");
168         err = em_idcom_call_svoc(svoc);
169         if (err < 0) {
170             return EM_ID23_PHASE_INIT_PUT_ERR;
171         }
172     } else {
173         sprintf(svoc, "s/put/bl_id23_phase_lower/preset");
174         err = em_idcom_call_svoc(svoc);
175         if (err < 0) {
176             return EM_ID23_PHASE_INIT_PUT_ERR;
177         }
178     }
179 */
180
181 /* prepare output */
182     em_data_clear(out);
183     em_data_put_int(out, 0, 0);
184
185     return 0;
186 }
```

```
/*(@header_begin)*************************************************************
 *  (C Function)                                                             *
 *      Name = em_id23_phase_move_one_side_put                               *
 *      Type = int                                                           *
 *  (Purpose)                                                                *
 *      drive servo-motor                                                    *
 *  (Input)                                                                  *
 *      parameter (config.tbl)                                               *
 *  (Output)                                                                 *
 *      none                                                                 *
 *  (Return)                                                                 *
 *      0   : ok                                                             *
 *      < 0 : fail                                                           *
 *  (Physical unit)                                                          *
 *      Pa, V, A, mm, sec                                                    *
 *  (Relation)                                                               *
 *                                                                           *
 *  (History)                                                                *
 *      Jul-22-1997      H.Funayama /iST  created.                           *
 *      98- 1-11         Hiramatsu    revised                                *
 *      98-10-14         Hiramatsu    updated                                *
 *      99-01-10         Hiramatsu    updated                                *
 *************************************************************(@header_end)*/

#include <sys/timers.h>
#include "em_id23.h"

int em_id23_phase_move_one_side_put(Em_data in, Em_data adr, Em_data *out)
{
    int fd, err, data_temp;
    int phase_kind;                 /* upper-phase or lower-phase */
    int wtime;                      /* wait-time       [microsec] */
    int pulse_num;                  /* number of pulses  to move  */
    unsigned short v_max, v_min, acc_cycle;   /* motor parameter  */
    double moment1, moment2;        /* absolute-time [sec]        */
    char pulse_out_status;          /* no pulse at status 0       */
    struct timespec tp;

#ifdef  EM_DEBUG_PRINTF
    printf("Call func name is ¥"em_id23_phase_move_one_side_put¥"¥n");
#endif

/* get file descriptor */
    fd = em_get_fd(adr);

/* read config.tbl */
    err = em_data_get_int(adr, EM_ID23_PHASE_MOVE_ONE_SIDE_V_MAX, &data_temp);
    if (err < 0) {
        return EM_ID23_PHASE_MOVE_ONE_SIDE_ERR_ADR_NOT_SET;
    }
    v_max = (unsigned short)data_temp;

    err = em_data_get_int(adr, EM_ID23_PHASE_MOVE_ONE_SIDE_V_MIN, &data_temp);
    if (err < 0) {
        return EM_ID23_PHASE_MOVE_ONE_SIDE_ERR_ADR_NOT_SET;
    }
    v_min = (unsigned short)data_temp;

    err = em_data_get_int(adr, EM_ID23_PHASE_MOVE_ONE_SIDE_ACC_CYCLE,
                          &data_temp);
    if (err < 0) {
        return EM_ID23_PHASE_MOVE_ONE_SIDE_ERR_ADR_NOT_SET;
    }
    acc_cycle = (unsigned short)data_temp;
```

```
65
66      err = em_data_get_int(adr, EM_ID23_PHASE_MOVE_ONE_SIDE_PHASE_SIDE,
67                            &phase_kind);
68      if (err < 0) {
69          return EM_ID23_PHASE_MOVE_ONE_SIDE_ERR_ADR_NOT_SET;
70      }
71
72      err = em_data_get_int(adr, EM_ID23_PHASE_MOVE_ONE_SIDE_WTIME, &wtime);
73      if (err < 0) {
74          return EM_ID23_PHASE_MOVE_ONE_SIDE_ERR_ADR_NOT_SET ;
75      }
76
77  /* get number of pulses */
78      if (phase_kind == EM_ID23_PHASE_MOVE_ONE_SIDE_PHASE_UPPER) {
79         em_data_get_int(in, EM_ID23_PHASE_MOVE_U_PULS_NUM, &pulse_num);
80      } else {
81         em_data_get_int(in, EM_ID23_PHASE_MOVE_L_PULS_NUM, &pulse_num);
82      }
83
84  /* drive servo-motor for phase movement */
85      if (phase_kind == EM_ID23_PHASE_MOVE_ONE_SIDE_PHASE_UPPER) {
86  #ifndef EM_DEBUG_SIM
87          dev_ptg0350_move(fd, pulse_num, 0, acc_cycle,
88                                  v_max, v_min, 1, EM_ID23_TIME_OUT_NUM);
89  #endif
90
91          getclock(TIMEOFDAY, &tp);
92          moment1 = (double)tp.tv_sec;
93
94  /* put current to steering magnets based on correction table */
95          do {
96              usleep(wtime * 10000);
97              err = em_id23_lc_feed_forward();
98              if (err < 0) {
99                  return EM_ID23_PHASE_FEED_FORWARD_ERR;
100             }
101
102 #ifndef EM_DEBUG_SIM
103             err = dev_ptg0350_out_check(fd, &pulse_out_status);
104             if (err < 0) {
105                 return EM_ID23_PHASE_MOVE_CHECK_ERR;
106             }
107 #else
108             pulse_out_status = 0;     /* no pulse at status 0 */
109 #endif
110
111             getclock(TIMEOFDAY, &tp);
112             moment2 = (double)tp.tv_sec;
113
114         } while (pulse_out_status != 0 || (moment2 - moment1) < 10.0);
115
116     } else {
117
118 #ifndef EM_DEBUG_SIM
119         dev_ptg0350_move(fd, 0, pulse_num, acc_cycle,
120                                 v_max, v_min, 1, EM_ID23_TIME_OUT_NUM);
121 #endif
122
123         getclock(TIMEOFDAY, &tp);
124         moment1 = (double)tp.tv_sec;
125
126         do {
127             usleep(wtime * 10000);
128             err = em_id23_lc_feed_forward();
```

```
129             if (err < 0) {
130                 return EM_ID23_PHASE_FEED_FORWARD_ERR;
131             }
132
133 #ifndef EM_DEBUG_SIM
134             err = dev_ptg0350_out_check(fd, &pulse_out_status);
135             if (err < 0) {
136                 return EM_ID23_PHASE_MOVE_CHECK_ERR;
137             }
138 #else
139             pulse_out_status = 0;     /* no pulse at status 0 */
140 #endif
141
142             getclock(TIMEOFDAY, &tp);
143             moment2 = (double)tp.tv_sec;
144
145         } while (pulse_out_status != 0 || (moment2 - moment1) < 10.0);
146
147     }
148
149 /* prepare output */
150     em_data_clear(out);
151     em_data_put_int(out, 0, err);
152
153     return 0;
154 }
```

```
/*(@header_begin)*************************************************************
 *  (C Function)                                                             *
 *      Name = em_id23_phase_move_put                                        *
 *      Type = int                                                           *
 *  (Purpose)                                                                *
 *      drive servo-motor, and excitate long-steering magnets                *
 *  (Input)                                                                  *
 *      parameter (config.tbl)                                               *
 *      v_max     = target generation rate (pulse/sec)                       *
 *      v_min     = generation rate at the start (pulse/sec)                 *
 *      acc_cycle = time-step to change from v to (v+1) (microsec)           *
 *  (Output)                                                                 *
 *      none                                                                 *
 *  (Return)                                                                 *
 *      0   : success                                                        *
 *      < 0 : fail                                                           *
 *  (Physical unit)                                                          *
 *      Pa, V, A, mm, sec                                                    *
 *  (Relation)                                                               *
 *                                                                           *
 *  (History)                                                                *
 *      98- 1-19       Hiramatsu     created                                 *
 *      98-10-14       Hiramatsu     updated                                 *
 *      99-01-08       Hiramatsu     updated for wtime(from config.tbl)      *
 *************************************************************(@header_end)*/

#include "em_id23.h"

int em_id23_phase_move_put(Em_data in, Em_data adr, Em_data *out)
{
    int err, fd, status, pulse_num_u, pulse_num_l, data_temp;
    int wtime;                  /* wait-time [microsec] */
    char pulse_out_status;
    unsigned short v_max , v_min, acc_cycle;
    Em_id23_position pos;

#ifdef EM_DEBUG_PRINTF
    printf("Call func name is ¥"em_id23_phase_move_put¥"¥n");
#endif

/* get file descriptor */
    fd = em_get_fd(adr);

/* get number of pulses */
    err = em_data_get_int(in, EM_ID23_PHASE_MOVE_U_PULS_NUM, &pulse_num_u);
    if (err < 0) {
        return EM_ID23_PHASE_MOVE_U_PULS_ERR;
    }

    err = em_data_get_int(in, EM_ID23_PHASE_MOVE_L_PULS_NUM, &pulse_num_l);
    if (err < 0) {
        return EM_ID23_PHASE_MOVE_L_PULS_ERR;
    }

/* read config.tbl */
    err = em_data_get_int(adr, EM_ID23_PHASE_MOVE_V_MAX, &data_temp);
    if (err < 0) {
        return EM_ID23_PHASE_MOVE_ERR_ADR_NOT_SET;
    }
    v_max = (unsigned short)data_temp;

    err = em_data_get_int(adr, EM_ID23_PHASE_MOVE_V_MIN, &data_temp);
    if (err < 0) {
        return EM_ID23_PHASE_MOVE_ERR_ADR_NOT_SET;
```

```
65    }
66    v_min = (unsigned short)data_temp;
67
68    err = em_data_get_int(adr, EM_ID23_PHASE_MOVE_ACC_CYCLE, &data_temp);
69    if (err < 0) {
70        return EM_ID23_PHASE_MOVE_ERR_ADR_NOT_SET;
71    }
72    acc_cycle = (unsigned short)data_temp;
73
74    err = em_data_get_int(adr, EM_ID23_PHASE_MOVE_WTIME, &wtime);
75    if (err < 0) {
76        return EM_ID23_PHASE_MOVE_ERR_ADR_NOT_SET;
77    }
78
79 /* drive servo-motor for phase movement */
80 #ifndef EM_DEBUG_SIM
81    dev_ptg0350_move(fd, pulse_num_u, pulse_num_l , acc_cycle,
82                          v_max, v_min, 0, EM_ID23_TIME_OUT_NUM);
83 #endif
84
85 /* put current to long-steering magnet based on correction table */
86    do {
87       usleep(wtime * 10000);
88       err= em_id23_lc_feed_forward();
89       if (err < 0) {
90           return EM_ID23_PHASE_FEED_FORWARD_ERR ;
91       }
92
93 #ifndef EM_DEBUG_SIM
94       err = dev_ptg0350_out_check(fd, &pulse_out_status);
95       if (err < 0) {
96           return EM_ID23_PHASE_MOVE_CHECK_ERR ;
97       }
98 #else
99       pulse_out_status = 0 ;
100 #endif
101
102   } while (pulse_out_status != 0) ;  /* no pulse at status 0 */
103
104   err = em_id23_lc_feed_forward();
105   if (err < 0) {
106       return EM_ID23_PHASE_FEED_FORWARD_ERR ;
107   }
108
109 /* prepare output */
110   em_data_clear(out);
111   em_data_put_int(out, 0, err);
112
113   return 0;
114 }
```

```
 1 /*(@header_begin)*************************************************************
 2 *  (C Function)                                                              *
 3 *      Name = em_id23_phase_position_get                                     *
 4 *      Type = int                                                            *
 5 *  (Purpose)                                                                 *
 6 *      calculate phase-shift(D) from upper&lower phase(z)                    *
 7 *  (Input)                                                                   *
 8 *      upper&lower phase(z)                                                  *
 9 *  (Output)                                                                  *
10 *      phase-shift(D)                                                        *
11 *  (Return)                                                                  *
12 *      0   : ok                                                              *
13 *      < 0 : fail                                                            *
14 *  (Physical unit)                                                           *
15 *      Pa, V, A, mm, sec                                                     *
16 *  (Relation)                                                                *
17 *                                                                            *
18 *  (History)                                                                 *
19 *      97-Dec-17   Hiramatsu  created                                        *
20 *      98-10-14    Hiramatsu  updated                                        *
21 *************************************************************(@header_end)*/
22
23 #include "em_id23.h"
24
25 int em_id23_phase_position_get(Em_data in, Em_data adr, Em_data *out)
26 {
27    int err;
28    char svoc[256], unit[20];
29    double phase, phase_up, phase_lo;
30    Em_id23_position pos;
31
32 #ifdef EM_DEBUG_PRINTF
33    printf("Call func name is ¥"em_id23_phase_position_get¥"¥n");
34 #endif
35
36 #ifndef EM_DEBUG_SIM
37 /* get upper linear scale */
38    err = em_idcom_func_svoc("s/get/bl_id23_phase_upper/position",
39                             &phase_up, unit);
40    if (err < 0) {
41        return EM_ID23_PHASE_U_POSITION_GET_ERR;
42    }
43
44 /* get lower linear scale */
45    err = em_idcom_func_svoc("s/get/bl_id23_phase_lower/position",
46                             &phase_lo, unit);
47    if (err < 0) {
48        return EM_ID23_PHASE_L_POSITION_GET_ERR;
49    }
50 #else
51    em_id23_util_position_cellar(EM_ID23_GET_POSITION, 1, &pos);
52    if (pos.set_flg == 0) {
53       return EM_ID23_UTIL_POSITION_CELLAR_ERR;
54    }
55    phase_up = pos.gapha;
56
57    em_id23_util_position_cellar(EM_ID23_GET_POSITION, 2, &pos);
58    if (pos.set_flg == 0) {
59       return EM_ID23_UTIL_POSITION_CELLAR_ERR;
60    }
61    phase_lo = pos.gapha;
62 #endif
63
64 /* calculate phase-shift(D) from upper & lower phase */
```

```
65     phase = phase_up - phase_lo;
66
67  /* prepare output */
68     em_data_clear(out);
69     em_data_put_double(out, 0, phase);
70
71     return 0;
72  }
```

```
/*(@header_begin)*************************************************************
 *  (C Function)                                                               *
 *      Name = em_id23_phase_preset_put                                        *
 *      Type = int                                                             *
 *  (Purpose)                                                                  *
 *      preset rotary-encoder and linear-scale                                 *
 *  (Input)                                                                    *
 *      parameter (config.tbl)                                                 *
 *  (Output)                                                                   *
 *      none                                                                   *
 *  (Return)                                                                   *
 *      0   : ok                                                               *
 *      < 0 : fail                                                             *
 *  (Physical unit)                                                            *
 *      Pa, V, A, mm, sec                                                      *
 *  (Relation)                                                                 *
 *                                                                             *
 *  (History)                                                                  *
 *      Jul-10-1997       F.Funayama /iST  created.                            *
 *      98-1-11           Hiramatsu     revised                                *
 *      98-10-14          Hiramatsu     updated                                *
 *****************************************************************(@header_end)*/
#include "em_id23.h"

int em_id23_phase_preset_put(Em_data in, Em_data adr, Em_data *out)
{
    int fd, err, dev_num_1, dev_num_2;
    int do_data;      /* =1(preset-on), =0(preset-off) */
    int time_num;     /* delay time [microsec]         */

#ifdef  EM_DEBUG_PRINTF
    printf("Call func name is ¥"em_id23_phase_preset_put¥"¥n");
#endif

/* get file descriptor */
    fd = em_get_fd(adr);

/* read config.tbl */
    err = em_data_get_int(adr, EM_ID23_PHASE_PRESET_LIN, &dev_num_1);
    if (err < 0) {
       return EM_ID23_PHASE_PRESET_ERR_ADR_NOT_SET;
    }

    err = em_data_get_int(adr, EM_ID23_PHASE_PRESET_ROT, &dev_num_2);
    if (err < 0) {
        return EM_ID23_PHASE_PRESET_ERR_ADR_NOT_SET;
    }

    err = em_data_get_int(adr, EM_ID23_PHASE_PRESET_DELAY_TIME, &time_num);
    if (err < 0) {
        return EM_ID23_PHASE_PRESET_ERR_ADR_NOT_SET;
    }

/* preset-on */
     do_data = 1;

#ifndef EM_DEBUG_SIM
    err = dev_do_put_dobit(fd, dev_num_1, &do_data);
    if (err < 0) {
        return EM_ID23_PHASE_PRESET_ERR;
    }

    err = dev_do_put_dobit(fd, dev_num_2, &do_data);
    if (err < 0) {
```

```
65          return EM_ID23_PHASE_PRESET_ERR;
66      }
67  #endif
68
69  /* sleep */
70      usleep(time_num);
71
72  /* preset-off(release) */
73      do_data = 0;
74
75  #ifndef EM_DEBUG_SIM
76      err = dev_do_put_dobit(fd, dev_num_1, &do_data);
77      if (err < 0) {
78          return EM_ID23_PHASE_PRESET_ERR;
79      }
80
81      err = dev_do_put_dobit(fd, dev_num_2, &do_data);
82      if (err < 0) {
83          return  EM_ID23_PHASE_PRESET_ERR;
84      }
85  #endif
86
87  /* prepare output */
88      em_data_clear(out);
89      em_data_put_int(out, 0, 0);
90
91      return 0 ;
92  }
```

```
 1 /*(@header_begin)*************************************************************
 2 *  (C Function)                                                              *
 3 *      Name = em_id23_phase_reset                                            *
 4 *      Type = int                                                            *
 5 *  (Purpose)                                                                 *
 6 *      reset interlock of drive amplifier                                    *
 7 *  (Input)                                                                   *
 8 *      parameter (config.tbl)                                                *
 9 *  (Output)                                                                  *
10 *      none                                                                  *
11 *  (Return)                                                                  *
12 *      0   : ok                                                              *
13 *      < 0 : fail                                                            *
14 *  (Physical unit)                                                           *
15 *      Pa, V, A, mm, sec                                                     *
16 *  (Relation)                                                                *
17 *                                                                            *
18 *  (History)                                                                 *
19 *      Jul-10-1997      F.Funayama /iST  created.                            *
20 *      98- 1-11         Hiramatsu     revised                                *
21 *      98-10-14         Hiramatsu     updated                                *
22 *************************************************************(@header_end)*/
23
24 #include "em_id23.h"
25
26 int em_id23_phase_reset(Em_data in, Em_data adr, Em_data *out)
27 {
28     int fd, err;
29     int time_num;               /* delay time [microsec]   */
30     int dev_num_1, dev_num_2;
31     int do_data;
32
33 #ifdef  EM_DEBUG_PRINTF
34     printf("Call func name is ¥"em_id23_phase_reset¥"¥n");
35 #endif
36
37 /* get file descriptor */
38     fd = em_get_fd(adr);
39
40 /* read config.tbl */
41     err = em_data_get_int(adr, EM_ID23_PHASE_RESET_OUT_BIT_1, &dev_num_1);
42     if (err < 0) {
43         return EM_ID23_PHASE_RESET_ERR_ADR_NOT_SET;
44     }
45
46     err = em_data_get_int(adr, EM_ID23_PHASE_RESET_OUT_BIT_2, &dev_num_2);
47     if (err < 0) {
48         return EM_ID23_PHASE_RESET_ERR_ADR_NOT_SET;
49     }
50
51     err = em_data_get_int(adr, EM_ID23_PHASE_RESET_DELAY_TIME, &time_num);
52     if (err < 0) {
53         return EM_ID23_PHASE_RESET_ERR_ADR_NOT_SET;
54     }
55
56 /* set data */
57     do_data = 1;
58
59 /* put reset */
60 #ifndef EM_DEBUG_SIM
61     err = dev_do_put_dobit(fd, dev_num_1, &do_data);
62     if (err < 0) {
63         return EM_ID23_PHASE_RESET_ERR;
64     }
```

```
65
66      err = dev_do_put_dobit(fd, dev_num_2, &do_data);
67      if (err < 0) {
68          return EM_ID23_PHASE_RESET_ERR;
69      }
70  #endif
71
72  /* sleep */
73      usleep(time_num);
74
75  /* set data */
76      do_data = 0;
77
78  /* put reset */
79  #ifndef EM_DEBUG_SIM
80      err = dev_do_put_dobit(fd, dev_num_1, &do_data);
81      if (err < 0) {
82         return EM_ID23_PHASE_RESET_ERR;
83      }
84
85      err = dev_do_put_dobit(fd, dev_num_2, &do_data);
86      if (err < 0) {
87          return EM_ID23_PHASE_RESET_ERR;
88      }
89  #endif
90
91  /* prepare output */
92      em_data_clear(out);
93      em_data_put_int(out, 0, 0);
94
95      return 0;
96  }
```

```
 1 /*(@header_begin)*************************************************************
 2 *  (C Function)                                                              *
 3 *      Name = em_id23_phase_status_get                                       *
 4 *      Type = int                                                            *
 5 *  (Purpose)                                                                 *
 6 *      read device status                                                    *
 7 *  (Input)                                                                   *
 8 *      parameter (config.tbl)                                                *
 9 *  (Output)                                                                  *
10 *      out->data[0].eint : di_data                                           *
11 *  (Return)                                                                  *
12 *      0   : ok                                                              *
13 *      < 0 : fail                                                            *
14 *  (Physical unit)                                                           *
15 *      Pa, V, A, mm, sec                                                     *
16 *  (Relation)                                                                *
17 *                                                                            *
18 *  (History)                                                                 *
19 *      Jul-11-1997      F.Funayama /iST  created.                            *
20 *      98- 1-11         Hiramatsu      revised                               *
21 *      98-10-14         Hiramatsu      updated                               *
22 *************************************************************(@header_end)*/
23
24 #include "em_id23.h"
25
26 int em_id23_phase_status_get(Em_data in, Em_data adr, Em_data *out)
27 {
28    int fd, dev_num, err, di_data;
29
30 #ifdef EM_DEBUG_PRINTF
31    printf("Call func name is ¥"em_id23_phase_status_get¥"¥n");
32 #endif
33
34 /* get file descriptor */
35    fd = em_get_fd(adr);
36
37 /* read config.tbl */
38    err = em_data_get_int(adr, EM_ID23_PHASE_STATUS_INPUT_PORT, &dev_num);
39    if (err < 0) {
40        return EM_ID23_PHASE_STATUS_ERR_ADR_NOT_SET;
41    }
42
43 #ifndef EM_DEBUG_SIM
44 /* get status data */
45    err = dev_di_get_diport(fd, dev_num, &di_data);
46    if (err < 0) {
47        return EM_ID23_PHASE_STATUS_GET_ERR;
48    }
49
50 /* shift bit position */
51    di_data = (di_data >> 5) & 0x3ff;
52 #else
53    di_data = 0xc6 ;
54 #endif
55
56 /* prepare output */
57    em_data_clear(out);
58    em_data_put_int(out, 0, di_data);
59
60    return 0 ;
61 }
```

```
1 /*(@header_begin)************************************************************
2 *  (C Function)                                                             *
3 *       Name = em_id23_phase_trigger_off_put                                *
4 *       Type = int                                                          *
5 *  (Purpose)                                                                *
6 *      put trigger-off signal to device                                     *
7 *  (Input)                                                                  *
8 *      parameter (config.tbl)                                               *
9 *  (Output)                                                                 *
10 *     none                                                                 *
11 *  (Return)                                                                *
12 *     0   : ok                                                             *
13 *     < 0 : fail                                                           *
14 *  (Physical unit)                                                         *
15 *     Pa, V, A, mm, sec                                                    *
16 *  (Relation)                                                              *
17 *                                                                          *
18 *  (History)                                                               *
19 *     Jul-11-1997      F.Funayama /iST  created.                           *
20 *     98- 1-11         Hiramatsu       revised                             *
21 *     98-10-14         Hiramatsu       updated                             *
22 ************************************************************(@header_end)*/
23
24 #include "em_id23.h"
25
26 int em_id23_phase_trigger_off_put(Em_data in, Em_data adr, Em_data *out)
27 {
28         int fd, err, dev_num, do_data;
29
30 #ifdef  EM_DEBUG_PRINTF
31         printf("Call func name is ¥"em_id23_phase_trigger_off_put¥"¥n");
32 #endif
33
34 /* get file descriptor */
35         fd = em_get_fd(adr);
36
37 /* read config.tbl */
38         err = em_data_get_int(adr, EM_ID23_PHASE_TRIGGER_BIT_NUM, &dev_num);
39         if (err < 0) {
40             return EM_ID23_PHASE_TRIGGER_OFF_ERR_ADR_NOT_SET;
41         }
42
43 /* trigger-off */
44         do_data = 0 ;
45
46 #ifndef EM_DEBUG_SIM
47         err = dev_ttldio_put_dobit(fd, dev_num, &do_data);
48         if (err < 0) {
49             return EM_ID23_PHASE_TRIGGER_STATUS_ERR ;
50         }
51 #endif
52
53 /* prepare output */
54         em_data_clear(out);
55         em_data_put_int(out, 0, 0);
56
57         return 0 ;
58 }
```

```
/*(@header_begin)*************************************************************
*  (C Function)                                                               *
*       Name = em_id23_phase_trigger_on_put                                   *
*       Type = int                                                            *
*  (Purpose)                                                                  *
*      put trigger-on signal to device                                        *
*  (Input)                                                                    *
*      parameter (config.tbl)                                                 *
*  (Output)                                                                   *
*      none                                                                   *
*  (Return)                                                                   *
*      0    : ok                                                              *
*      < 0 : fail                                                             *
*  (Physical unit)                                                            *
*      Pa, V, A, mm, sec                                                      *
*  (Relation)                                                                 *
*                                                                             *
*  (History)                                                                  *
*      Jul-11-1997       F.Funayama /iST  created.                            *
*      98- 1-11          Hiramatsu       revised                              *
*      98-10-14          Hiramatsu       updated                              *
*****************************************************************(@header_end)*/

#include "em_id23.h"

int em_id23_phase_trigger_on_put(Em_data in, Em_data adr, Em_data *out)
{
        int fd, err, dev_num, do_data;

#ifdef  EM_DEBUG_PRINTF
        printf("Call func name is ¥"em_id23_phase_trigger_on_put¥"¥n");
#endif

/* get file descriptor */
        fd = em_get_fd(adr);

/* read config.tbl */
        err = em_data_get_int(adr, EM_ID23_PHASE_TRIGGER_BIT_NUM, &dev_num);
        if (err < 0) {
            return EM_ID23_PHASE_TRIGGER_ON_ERR_ADR_NOT_SET;
        }

/* trigger-on */
        do_data = 1 ;

#ifndef EM_DEBUG_SIM
        err = dev_ttldio_put_dobit(fd, dev_num, &do_data);
        if (err < 0) {
            return EM_ID23_PHASE_TRIGGER_STATUS_ERR;
        }
#endif

/* prepare output */
        em_data_clear(out);
        em_data_put_int(out, 0, 0);

        return 0 ;
}
```

```
/*(@header_begin)*************************************************************
 *  (C Function)                                                             *
 *      Name = em_id23_phase_trigger_status_get                              *
 *      Type = int                                                           *
 *  (Purpose)                                                                *
 *      read device status                                                   *
 *  (Input)                                                                  *
 *      parameter (config.tbl)                                               *
 *  (Output)                                                                 *
 *      out->data[0].eint = dio_data                                         *
 *  (Return)                                                                 *
 *      0   : ok                                                             *
 *      < 0 : fail                                                           *
 *  (Physical unit)                                                          *
 *      Pa, V, A, mm, sec                                                    *
 *  (Relation)                                                               *
 *                                                                           *
 *  (History)                                                                *
 *      99-01-15          Hiramatsu       created                            *
 *************************************************************(@header_end)*/
#include "em_id23.h"

int em_id23_phase_trigger_status_get(Em_data in, Em_data adr, Em_data *out)
{
    int fd, err, dio_num, dio_data;

#ifdef  EM_DEBUG_PRINTF
    printf("Call func name is ¥"em_id23_phase_trigger_status_get¥"¥n");
#endif

/* get file descriptor */
    fd = em_get_fd(adr);

/* read config.tbl */
    err = em_data_get_int(adr, EM_ID23_PHASE_TRIGGER_BIT_NUM, &dio_num);
    if (err < 0) {
        return EM_ID23_PHASE_TRIGGER_ON_ERR_ADR_NOT_SET;
    }

/* get trigger status(0:trigger-off, 1:trigger-on) */
#ifndef EM_DEBUG_SIM
    err = dev_ttldio_get_dobit(fd, dio_num, &dio_data);
    if (err < 0) {
        return EM_ID23_PHASE_TRIGGER_STATUS_ERR ;
    }
#endif

/* prepare output */
    em_data_clear(out);
    em_data_put_int(out, 0, dio_data);

    return 0 ;
}
```

```
 1 /*(@header_begin)*************************************************************
 2 *  (C Function)                                                              *
 3 *      Name = em_id23_position_conv_get                                      *
 4 *      Type = int                                                            *
 5 *  (Purpose)                                                                 *
 6 *      convert BCD data to real value(gap and phase-shift)                   *
 7 *  (Input)                                                                   *
 8 *      parameter (config.tbl) and BCD data                                   *
 9 *  (Output)                                                                  *
10 *      real value of position                                                *
11 *  (Return)                                                                  *
12 *      0   : ok                                                              *
13 *      < 0 : fail                                                            *
14 *  (Physical unit)                                                           *
15 *      Pa, V, A, mm, sec                                                     *
16 *  (Relation)                                                                *
17 *                                                                            *
18 *  (History)                                                                 *
19 *      Jul-08-1997       F.Funayama /iST  created.                           *
20 *      97-12-20          Hiramatsu  checked for unit(mm)                     *
21 *      98-10-14          Hiramatsu  updated                                  *
22 *      99-01-10          Hiramatsu  updated                                  *
23 *************************************************************(@header_end)*/
24
25 #include "em_id23.h"
26
27 int em_id23_position_conv_get(Em_data in, Em_data conv, Em_svoc *out)
28 {
29    int data_kind;       /* rotary-encoder or linear-scale */
30    int err, bcd_data[20];
31    double position;     /* position [mm] */
32    short cnt, count;
33
34 #ifdef EM_DEBUG_PRINTF
35    printf("Call func name is ¥"em_id23_position_conv_get¥"¥n");
36 #endif
37
38 /* prepare output */
39    em_data_clear(out);
40
41 /* read config.tbl */
42    err = em_data_get_int(conv, EM_ID23_POSITION_CONV_KIND, &data_kind);
43    if (err < 0) {
44        return EM_ID23_POSITION_CONV_ERR_CNV_NOT_SET;
45    }
46
47    switch(data_kind) {
48    case EM_ID23_GAP_ROT:
49    case EM_ID23_PHASE_U_ROT:
50    case EM_ID23_PHASE_L_ROT:
51         count = 4;
52         break;
53    case EM_ID23_GAP_LIN:
54    case EM_ID23_PHASE_U_LIN:
55    case EM_ID23_PHASE_L_LIN:
56         count = 12;
57         break;
58    default:
59         return EM_ID23_POSITION_CONV_ERR_DATA_KIND_DIFFER ;
60    }
61
62    for (cnt = 0 ; cnt < count ; cnt++) {
63       em_data_get_int(in, cnt, &(bcd_data[cnt]));
64    }
```

```
65
66      err = em_id23_util_position_conv(data_kind, bcd_data, &position);
67      if (err < 0) {
68          return EM_ID23_POSITION_CONV_ERR ;
69      }
70
71  /* store actual data (position) */
72      if (err == 0) {
73          switch(data_kind) {
74          case EM_ID23_GAP_ROT:
75              sprintf(out->svoc, "%3.3lfmm", position);
76              break;
77          case EM_ID23_PHASE_U_ROT:
78          case EM_ID23_PHASE_L_ROT:
79              sprintf(out->svoc, "%2.2lfmm", position);
80              break;
81
82          case EM_ID23_GAP_LIN:
83          case EM_ID23_PHASE_U_LIN:
84          case EM_ID23_PHASE_L_LIN:
85              sprintf(out->svoc, "%4.3lfmm", position);
86              break;
87
88          default:
89              return EM_ID23_POSITION_CONV_ERR_DATA_KIND_DIFFER ;
90          }
91      } else {
92          sprintf(out->svoc, "fail");
93      }
94
95      return 0 ;
96  }
```

```
 1 /*(@header_begin)*************************************************************
 2 *  (C Function)                                                              *
 3 *      Name = em_id23_position_gap_lin_get                                   *
 4 *      Type = int                                                            *
 5 *  (Purpose)                                                                 *
 6 *      get gap position from linear-scale, then latch-off                    *
 7 *  (Input)                                                                   *
 8 *      parameter (config.tbl)                                                *
 9 *  (Output)                                                                  *
10 *      di_data                                                               *
11 *  (Return)                                                                  *
12 *      0   : ok                                                              *
13 *      < 0 : fail                                                            *
14 *  (Physical unit)                                                           *
15 *      Pa, V, A, mm, sec                                                     *
16 *  (Relation)                                                                *
17 *                                                                            *
18 *  (History)                                                                 *
19 *      98- 1-7       Hiramatsu   created                                     *
20 *      98-10-14      Hiramatsu   updated                                     *
21 *      99-03-17      Hiramatsu   modify latchon,encoder_get,latchoff         *
22 *************************************************************(@header_end)*/
23
24 #include "em_id23.h"
25
26 int em_id23_position_gap_lin_get(Em_data in, Em_data adr, Em_data *out)
27 {
28    int err, i, ct ;
29    int fd_0 ;          /* file descriptor for latchon and off  */
30    int fd_1 ;          /* file descriptor for position reading */
31    int dio_num ;       /* bit number to read                   */
32    int time_num ;      /* delay time [microsec]                */
33    int dio_data ;      /* =0(latchon) , =1(latchoff)           */
34    int di_data[12];    /* read position data                   */
35
36 #ifdef EM_DEBUG_PRINTF
37    printf("Call func name is ¥"em_id23_position_gap_lin_get¥"¥n");
38 #endif
39
40 #ifndef EM_DEBUG_SIM
41 /* initialize */
42    for (i = 0 ; i < 12 ; i++) {
43       di_data[i] = 0x0;
44    }
45 #else
46    di_data[0]  = 0x45 ;  /* dummy data for simulation */
47    di_data[1]  = 0x28 ;
48    di_data[2]  = 0x00 ;
49    di_data[3]  = 0x23 ;
50    di_data[4]  = 0x18 ;
51    di_data[5]  = 0x26 ;
52    di_data[6]  = 0x00 ;
53    di_data[7]  = 0x00 ;
54    di_data[8]  = 0x18 ;
55    di_data[9]  = 0x26 ;
56    di_data[10] = 0x10 ;
57    di_data[11] = 0x00 ;
58 #endif
59
60 /* get file descriptor */
61    fd_0 = em_get_fd_multi(adr, 0) ;    /* TTLDIO device (latch) */
62    fd_1 = em_get_fd_multi(adr, 3) ;    /* DI or TTLDI device    */
63
64 /* read config.tbl */
```

```
65    err = em_data_get_int(adr, 1, &dio_num);
66    if (err < 0) {
67        return EM_ID23_POSITION_GET_ERR_ADR_NOT_SET;
68    }
69
70    err = em_data_get_int(adr, 2, &time_num);
71    if (err < 0) {
72        return EM_ID23_POSITION_GET_ERR_ADR_NOT_SET;
73    }
74
75 /* verify latch on/off */
76 #ifndef EM_DEBUG_SIM
77    err = dev_ttldio_get_dobit(fd_0, dio_num, &dio_data);
78    if (err < 0) {
79        return EM_ID23_POSITION_GET_ERR;
80    }
81    if (dio_data == 0) {       /* latchon */
82        ct = 0 ;
83        do {        /* wait for the latchoff */
84            err = dev_ttldio_get_dobit(fd_0, dio_num, &dio_data);
85            if (err < 0) {
86                return EM_ID23_POSITION_GET_ERR;
87            }
88
89            ++ct;
90
91            if (ct > 100000) {
92                dio_data = 1 ;    /* forced latchoff */
93            }
94
95        } while(dio_data == 0);
96    }                         /* latchoff */
97
98 /* latchon yourself */
99    dio_data = 0 ;
100   err = dev_ttldio_put_dobit(fd_0, dio_num, &dio_data);
101   if (err < 0) {
102       return EM_ID23_POSITION_GET_ERR ;
103   }
104 #endif
105
106 /* sleep */
107   em_id23_util_delay_time(time_num);
108
109 #ifndef EM_DEBUG_SIM
110   err = dev_ttldi_get_diall(fd_1, di_data);
111   if (err < 0) {
112       return EM_ID23_POSITION_GET_ERR ;
113   }
114
115   dio_data = 1 ;     /* latchoff yourself */
116   err = dev_ttldio_put_dobit(fd_0, dio_num, &dio_data);
117   if (err < 0) {
118       return EM_ID23_POSITION_GET_ERR ;
119   }
120 #endif
121
122 /* prepare output */
123   em_data_clear(out);
124   for (i = 0 ; i < 12 ; i++) {
125   em_data_put_int(out , i, di_data[i]);
126   }
127
128   return 0;
```

```
/*(@header_begin)*************************************************************
 *  (C Function)                                                             *
 *      Name = em_id23_position_gap_rot_get                                  *
 *      Type = int                                                           *
 *  (Purpose)                                                                *
 *      get gap position from rotary-encoder, then latch-off                 *
 *  (Input)                                                                  *
 *      parameter (config.tbl)                                               *
 *  (Output)                                                                 *
 *      di_data                                                              *
 *  (Return)                                                                 *
 *      0   : ok                                                             *
 *      < 0 :fail                                                            *
 *  (Physical unit)                                                          *
 *      Pa, V, A, mm, sec                                                    *
 *  (Relation)                                                               *
 *                                                                           *
 *  (History)                                                                *
 *      98- 1-7       Hiramatsu   created                                    *
 *      98-10-14      Hiramatsu   updated                                    *
 *      99-03-17      Hiramatsu   modify latchon,encoder_get,latchoff        *
 *************************************************************(@header_end)*/

#include "em_id23.h"

int em_id23_position_gap_rot_get(Em_data in, Em_data adr, Em_data *out)
{
   int err, i, ct ;
   int fd_0 ;          /* file descriptor for latchon and off  */
   int fd_1 ;          /* file descriptor for position reading */
   int dio_num ;       /* bit number to read                   */
   int time_num ;      /* delay time [microsec]                */
   int dio_data ;      /* =0(latchon) , =1(latchoff)           */
   int di_data[12];    /* read position data                   */

#ifdef EM_DEBUG_PRINTF
   printf("Call func name is ¥"em_id23_position_gap_rot_get¥"¥n");
#endif

#ifndef EM_DEBUG_SIM
/* initialize */
   for (i = 0 ; i < 12 ; i++) {
      di_data[i] = 0x0;
   }
#else
   di_data[0]  = 0x45 ;  /* dummy data for simulation */
   di_data[1]  = 0x28 ;
   di_data[2]  = 0x00 ;
   di_data[3]  = 0x23 ;
   di_data[4]  = 0x18 ;
   di_data[5]  = 0x26 ;
   di_data[6]  = 0x00 ;
   di_data[7]  = 0x00 ;
   di_data[8]  = 0x18 ;
   di_data[9]  = 0x26 ;
   di_data[10] = 0x10 ;
   di_data[11] = 0x00 ;
#endif

/* get file descriptor */
   fd_0 = em_get_fd_multi(adr, 0) ;    /* TTLDIO device (latch) */
   fd_1 = em_get_fd_multi(adr, 3) ;    /* DI or TTLDI device    */

/* read config.tbl */
```

```
65     err = em_data_get_int(adr, 1, &dio_num);
66     if (err < 0) {
67         return EM_ID23_POSITION_GET_ERR_ADR_NOT_SET;
68     }
69
70     /* read delay-time from config.tbl */
71     err = em_data_get_int(adr, 2, &time_num);
72     if (err < 0) {
73        return EM_ID23_POSITION_GET_ERR_ADR_NOT_SET;
74     }
75
76 /* verify latch on/off */
77 #ifndef EM_DEBUG_SIM
78     err = dev_ttldio_get_dobit(fd_0, dio_num, &dio_data);
79     if (err < 0) {
80         return EM_ID23_POSITION_GET_ERR;
81     }
82
83     if (dio_data == 0) {        /* latchon */
84         ct = 0 ;
85         do {        /* wait for the latchoff */
86             err = dev_ttldio_get_dobit(fd_0, dio_num, &dio_data);
87             if (err < 0) {
88                 return EM_ID23_POSITION_GET_ERR;
89             }
90
91             ++ct ;
92
93             if (ct > 100000) {
94                 dio_data = 1 ;    /* forced latchoff */
95             }
96
97         } while (dio_data == 0);
98     }                          /* latchoff */
99
100 /* latchon yourself */
101     dio_data = 0 ;
102     err = dev_ttldio_put_dobit(fd_0, dio_num, &dio_data);
103     if (err < 0) {
104         return EM_ID23_POSITION_GET_ERR ;
105     }
106 #endif
107
108 /* sleep */
109     em_id23_util_delay_time(time_num);
110
111 #ifndef EM_DEBUG_SIM
112     err = dev_di_get_diall(fd_1, di_data);
113     if (err < 0) {
114         return EM_ID23_POSITION_GET_ERR;
115     }
116
117     dio_data = 1 ;      /* latchoff yourself */
118     err = dev_ttldio_put_dobit(fd_0, dio_num, &dio_data);
119     if (err < 0) {
120         return EM_ID23_POSITION_GET_ERR;
121     }
122 #endif
123
124 /* prepare output */
125     em_data_clear(out);
126     for (i = 0 ; i < 12 ; i++) {
127         em_data_put_int(out, i ,di_data[i]);
128     }
129
130     return 0;
131 }
```

```
 1 /*(@header_begin)*************************************************************
 2 *  (C Function)                                                              *
 3 *      Name = em_id23_position_phase_l_lin_get                               *
 4 *      Type = int                                                            *
 5 *  (Purpose)                                                                 *
 6 *      get phase-lower position from linear-scale, then latch-off            *
 7 *  (Input)                                                                   *
 8 *      parameter (config.tbl)                                                *
 9 *  (Output)                                                                  *
10 *      di_data                                                               *
11 *  (Return)                                                                  *
12 *      0   : ok                                                              *
13 *      < 0 : fail                                                            *
14 *  (Physical unit)                                                           *
15 *      Pa, V, A, mm, sec                                                     *
16 *  (Relation)                                                                *
17 *                                                                            *
18 *  (History)                                                                 *
19 *      98- 1-7       Hiramatsu   created                                     *
20 *      98-10-14      Hiramatsu   updated                                     *
21 *      99-03-17      Hiramatsu   modify latchon,encoder_get,latchoff         *
22 *************************************************************(@header_end)*/
23
24 #include "em_id23.h"
25
26 int em_id23_position_phase_l_lin_get(Em_data in, Em_data adr, Em_data *out)
27 {
28    int err, i, ct ;
29    int fd_0 ;          /* file descriptor for latchon and off  */
30    int fd_1 ;          /* file descriptor for position reading */
31    int dio_num ;       /* bit number to read                   */
32    int time_num ;      /* delay time [microsec]                */
33    int dio_data ;      /* =0(latchon) , =1(latchoff)           */
34    int di_data[12];    /* read position data                   */
35
36 #ifdef EM_DEBUG_PRINTF
37    printf("Call func name is ¥"em_id23_position_phase_l_lin_get¥"¥n");
38 #endif
39
40 #ifndef EM_DEBUG_SIM
41 /* initialize */
42    for (i = 0 ; i < 12 ; i++) {
43       di_data[i] = 0x0;
44    }
45 #else
46    di_data[0]  = 0x45 ;  /* dummy data for simulation */
47    di_data[1]  = 0x28 ;
48    di_data[2]  = 0x00 ;
49    di_data[3]  = 0x23 ;
50    di_data[4]  = 0x18 ;
51    di_data[5]  = 0x26 ;
52    di_data[6]  = 0x00 ;
53    di_data[7]  = 0x00 ;
54    di_data[8]  = 0x18 ;
55    di_data[9]  = 0x26 ;
56    di_data[10] = 0x10 ;
57    di_data[11] = 0x00 ;
58 #endif
59
60 /* get file descriptor */
61    fd_0 = em_get_fd_multi(adr, 0) ;    /* TTLDIO device (latch) */
62    fd_1 = em_get_fd_multi(adr, 3) ;    /* DI or TTLDI device    */
63
64 /* read config.tbl */
```

```
65      err = em_data_get_int(adr, 1, &dio_num);
66      if (err < 0) {
67          return EM_ID23_POSITION_GET_ERR_ADR_NOT_SET;
68      }
69
70      err = em_data_get_int(adr, 2, &time_num);
71      if (err < 0) {
72          return EM_ID23_POSITION_GET_ERR_ADR_NOT_SET;
73      }
74
75  /* verify latch on/off */
76  #ifndef EM_DEBUG_SIM
77      err = dev_ttldio_get_dobit(fd_0, dio_num, &dio_data);
78      if (err < 0) {
79          return EM_ID23_POSITION_GET_ERR;
80      }
81
82      if (dio_data == 0) {        /* latchon */
83          ct = 0 ;
84          do {        /* wait for latch turn-off */
85              err = dev_ttldio_get_dobit(fd_0, dio_num, &dio_data);
86              if (err < 0) {
87                  return EM_ID23_POSITION_GET_ERR ;
88              }
89
90              ++ct;
91
92              if (ct > 100000) {
93                  dio_data = 1 ;     /* forced latchoff */
94              }
95
96          } while (dio_data == 0);
97      }                           /* latchoff */
98
99      dio_data = 0;       /* latchon yourself */
100     err = dev_ttldio_put_dobit(fd_0, dio_num, &dio_data);
101     if (err < 0) {
102         return EM_ID23_POSITION_GET_ERR;
103     }
104 #endif
105
106 /* sleep */
107     em_id23_util_delay_time(time_num);
108
109 #ifndef EM_DEBUG_SIM
110     err = dev_ttldi_get_diall(fd_1, di_data);
111     if (err < 0) {
112         return EM_ID23_POSITION_GET_ERR;
113     }
114
115     dio_data = 1;      /* latchoff yourself */
116     err = dev_ttldio_put_dobit(fd_0, dio_num, &dio_data);
117     if (err < 0) {
118         return EM_ID23_POSITION_GET_ERR;
119     }
120 #endif
121
122 /* prepare output */
123     em_data_clear(out);
124     for (i = 0 ; i < 12 ; i++) {
125         err = em_data_put_int(out, i , di_data[i]);
126     }
127
128     return 0;
```

```
/*(@header_begin)*************************************************************
*  (C Function)                                                               *
*      Name = em_id23_position_phase_l_rot_get                                *
*      Type = int                                                             *
*  (Purpose)                                                                  *
*      get phase-lower position from rotary-encoder, then latch-off           *
*  (Input)                                                                    *
*      parameter (config.tbl)                                                 *
*  (Output)                                                                   *
*      di_data                                                                *
*  (Return)                                                                   *
*      0   : ok                                                               *
*      < 0 : fail                                                             *
*  (Physical unit)                                                            *
*      Pa, V, A, mm, sec                                                      *
*  (Relation)                                                                 *
*                                                                             *
*  (History)                                                                  *
*      98- 1-7       Hiramatsu   created                                      *
*      98-10-14      Hiramatsu   updated                                      *
*      99-03-17      Hiramatsu   modify latchon,encoder_get,latchoff          *
*************************************************************(@header_end)*/

#include "em_id23.h"

int em_id23_position_phase_l_rot_get(Em_data in, Em_data adr, Em_data *out)
{
   int err, i, ct ;
   int fd_0 ;          /* file descriptor for latchon and off  */
   int fd_1 ;          /* file descriptor for position reading */
   int dio_num ;       /* bit number to read                   */
   int time_num ;      /* delay time [microsec]                */
   int dio_data ;      /* =0(latchon) , =1(latchoff)           */
   int di_data[12];    /* read position data                   */

#ifdef EM_DEBUG_PRINTF
   printf("Call func name is ¥"em_id23_position_phase_l_rot_get¥"¥n");
#endif

#ifndef EM_DEBUG_SIM
/* initialize */
   for (i = 0 ; i < 12 ; i++) {
      di_data[i] = 0x0;
   }
#else
   di_data[0]  = 0x45 ;  /* dummy data for simulation */
   di_data[1]  = 0x28 ;
   di_data[2]  = 0x00 ;
   di_data[3]  = 0x23 ;
   di_data[4]  = 0x18 ;
   di_data[5]  = 0x26 ;
   di_data[6]  = 0x00 ;
   di_data[7]  = 0x00 ;
   di_data[8]  = 0x18 ;
   di_data[9]  = 0x26 ;
   di_data[10] = 0x10 ;
   di_data[11] = 0x00 ;
#endif

/* get file descriptor */
   fd_0 = em_get_fd_multi(adr, 0) ;    /* TTLDIO device (latch) */
   fd_1 = em_get_fd_multi(adr, 3) ;    /* DI or TTLDI device    */

/* read config.tbl */
```

```
65    err = em_data_get_int(adr, 1, &dio_num);
66    if (err < 0) {
67        return EM_ID23_POSITION_GET_ERR_ADR_NOT_SET;
68    }
69
70    err = em_data_get_int(adr, 2, &time_num);
71    if (err < 0) {
72        return EM_ID23_POSITION_GET_ERR_ADR_NOT_SET;
73    }
74
75 /* verify latch on/off */
76 #ifndef EM_DEBUG_SIM
77    err = dev_ttldio_get_dobit(fd_0, dio_num, &dio_data);
78    if (err < 0) {
79        return EM_ID23_POSITION_GET_ERR ;
80    }
81
82    if (dio_data == 0) {        /* latchon */
83        ct = 0 ;
84        do {        /* wait for latchoff */
85            err = dev_ttldio_get_dobit(fd_0, dio_num, &dio_data);
86            if (err < 0) {
87                return EM_ID23_POSITION_GET_ERR;
88            }
89
90            ++ct;
91
92            if (ct > 100000) {
93                dio_data = 1 ;    /* forced latchoff */
94            }
95
96        } while (dio_data == 0);
97    }                        /* latchoff */
98
99    dio_data = 0;       /* latchon yourself */
100   err = dev_ttldio_put_dobit(fd_0, dio_num, &dio_data);
101   if (err < 0) {
102       return EM_ID23_POSITION_GET_ERR;
103   }
104 #endif
105
106 /* sleep */
107    em_id23_util_delay_time(time_num);
108
109 #ifndef EM_DEBUG_SIM
110    err = dev_di_get_diall(fd_1, di_data);
111    if (err < 0) {
112       return EM_ID23_POSITION_GET_ERR;
113    }
114
115    dio_data = 1;      /* latchoff yourself */
116    err = dev_ttldio_put_dobit(fd_0, dio_num, &dio_data);
117    if (err < 0) {
118        return EM_ID23_POSITION_GET_ERR;
119    }
120 #endif
121
122 /* prepare output */
123    em_data_clear(out);
124    for (i = 0 ; i < 12 ; i++) {
125        err = em_data_put_int(out, i , di_data[i]);
126    }
127
128    return 0;
```

```
 1 /*(@header_begin)*************************************************************
 2 *  (C Function)                                                              *
 3 *      Name = em_id23_position_phase_u_lin_get                               *
 4 *      Type = int                                                            *
 5 *  (Purpose)                                                                 *
 6 *      get phase-upper position from linear-scale, then latch-off            *
 7 *  (Input)                                                                   *
 8 *      parameter (config.tbl)                                                *
 9 *  (Output)                                                                  *
10 *      di_data                                                               *
11 *  (Return)                                                                  *
12 *      0   : ok                                                              *
13 *      < 0 : fail                                                            *
14 *  (Physical unit)                                                           *
15 *      Pa, V, A, mm, sec                                                     *
16 *  (Relation)                                                                *
17 *                                                                            *
18 *  (History)                                                                 *
19 *      98- 1-7       Hiramatsu   created                                     *
20 *      98-10-14      Hiramatsu   updated                                     *
21 *      99-03-17      Hiramatsu   modify latchon,encoder_get,latchoff         *
22 *************************************************************(@header_end)*/
23
24 #include <sys/timers.h>
25 #include "em_id23.h"
26
27 int em_id23_position_phase_u_lin_get(Em_data in, Em_data adr, Em_data *out)
28 {
29    int err, i, ct ;
30    int fd_0 ;        /* file descriptor for latchon and off  */
31    int fd_1 ;        /* file descriptor for position reading */
32    int dio_num ;     /* bit number to read                   */
33    int time_num ;    /* delay time [microsec]                */
34    int dio_data ;    /* =0(latchon) , =1(latchoff)           */
35    int di_data[12];  /* read position data                   */
36    struct timespec tp;
37
38 #ifdef EM_DEBUG_PRINTF
39    printf("Call func name is ¥"em_id23_position_phase_u_lin_get¥"¥n");
40 #endif
41
42 #ifndef EM_DEBUG_SIM
43 /* initialize */
44    for (i = 0 ; i < 12 ; i++) {
45       di_data[i] = 0x0;
46    }
47 #else
48    di_data[0]  = 0x45 ;  /* dummy data for simulation */
49    di_data[1]  = 0x28 ;
50    di_data[2]  = 0x00 ;
51    di_data[3]  = 0x23 ;
52    di_data[4]  = 0x18 ;
53    di_data[5]  = 0x26 ;
54    di_data[6]  = 0x00 ;
55    di_data[7]  = 0x00 ;
56    di_data[8]  = 0x18 ;
57    di_data[9]  = 0x26 ;
58    di_data[10] = 0x10 ;
59    di_data[11] = 0x00 ;
60 #endif
61
62 /* get file descriptor */
63    fd_0 = em_get_fd_multi(adr, 0) ;    /* TTLDIO device (latch) */
64    fd_1 = em_get_fd_multi(adr, 3) ;    /* DI or TTLDI device    */
```

```
65
66 /* read config.tbl */
67    err = em_data_get_int(adr, 1, &dio_num);
68    if (err < 0) {
69       return EM_ID23_POSITION_GET_ERR_ADR_NOT_SET;
70    }
71
72    err = em_data_get_int(adr, 2, &time_num);
73    if (err < 0) {
74        return EM_ID23_POSITION_GET_ERR_ADR_NOT_SET;
75    }
76
77 /* verify latch on/off */
78 #ifndef EM_DEBUG_SIM
79    err = dev_ttldio_get_dobit(fd_0, dio_num, &dio_data);
80    if (err < 0) {
81        return EM_ID23_POSITION_GET_ERR;
82    }
83
84    if (dio_data == 0) {       /* latchon */
85        ct = 0 ;
86        do {        /* wait for latchoff */
87            err = dev_ttldio_get_dobit(fd_0, dio_num, &dio_data);
88            if (err < 0) {
89                return EM_ID23_POSITION_GET_ERR;
90            }
91
92            ++ct ;
93
94            if (ct > 100000) {
95                dio_data = 1 ;    /* forced latchoff */
96            }
97
98        } while (dio_data == 0);
99    }                        /* latchoff */
100
101   dio_data = 0;      /* latchon yourself */
102   err = dev_ttldio_put_dobit(fd_0, dio_num, &dio_data);
103   if( err < 0 ) {
104       return EM_ID23_POSITION_GET_ERR;
105   }
106 #endif
107
108 /* sleep */
109   em_id23_util_delay_time(time_num);
110
111 #ifndef EM_DEBUG_SIM
112   err = dev_ttldi_get_diall(fd_1, di_data);
113   if (err < 0) {
114      return EM_ID23_POSITION_GET_ERR;
115   }
116
117   dio_data = 1;      /* latchoff yourself */
118   err = dev_ttldio_put_dobit(fd_0, dio_num, &dio_data);
119   if (err < 0) {
120       return EM_ID23_POSITION_GET_ERR;
121   }
122 #endif
123
124 /* prepare output */
125   em_data_clear(out);
126   for (i = 0 ; i < 12 ; i++) {
127       err = em_data_put_int(out, i, di_data[i]);
128   }
129
130   return 0;
131 }
```

```
/*(@header_begin)*************************************************************
 *  (C Function)                                                               *
 *      Name = em_id23_position_phase_u_rot_get                                *
 *      Type = int                                                             *
 *  (Purpose)                                                                  *
 *      get phase-upper position from rotary-encoder, then latch-off           *
 *  (Input)                                                                    *
 *      parameter (config.tbl)                                                 *
 *  (Output)                                                                   *
 *      di_data                                                                *
 *  (Return)                                                                   *
 *      0    : ok                                                              *
 *      < 0  : fail                                                            *
 *  (Physical unit)                                                            *
 *      Pa, V, A, mm, sec                                                      *
 *  (Relation)                                                                 *
 *                                                                             *
 *  (History)                                                                  *
 *      98- 1-7       Hiramatsu   created                                      *
 *      98-10-14      Hiramatsu   updated                                      *
 *      99-03-17      Hiramatsu   modify latchon,encoder_get,latchoff          *
 *************************************************************(@header_end)*/

#include "em_id23.h"

int em_id23_position_phase_u_rot_get(Em_data in, Em_data adr, Em_data *out)
{
    int err, i, ct ;
    int fd_0 ;          /* file descriptor for latchon and off  */
    int fd_1 ;          /* file descriptor for position reading */
    int dio_num ;       /* bit number to read                   */
    int time_num ;      /* delay time [microsec]                */
    int dio_data ;      /* =0(latchon) , =1(latchoff)           */
    int di_data[12];    /* read position data                   */

#ifdef EM_DEBUG_PRINTF
    printf("Call func name is ¥"em_id23_position_phase_u_rot_get¥"");
#endif

#ifndef EM_DEBUG_SIM
/* initialize */
    for (i = 0 ; i < 12 ; i++) {
        di_data[i] = 0x0;
    }
#else
    di_data[0]  = 0x45 ;  /* dummy data for simulation */
    di_data[1]  = 0x28 ;
    di_data[2]  = 0x00 ;
    di_data[3]  = 0x23 ;
    di_data[4]  = 0x18 ;
    di_data[5]  = 0x26 ;
    di_data[6]  = 0x00 ;
    di_data[7]  = 0x00 ;
    di_data[8]  = 0x18 ;
    di_data[9]  = 0x26 ;
    di_data[10] = 0x10 ;
    di_data[11] = 0x00 ;
#endif

/* get file descriptor */
    fd_0 = em_get_fd_multi(adr, 0) ;    /* TTLDIO device (latch) */
    fd_1 = em_get_fd_multi(adr, 3) ;    /* DI or TTLDI device    */

/* read config.tbl */
```

```
65    err = em_data_get_int(adr, 1, &dio_num);
66    if (err < 0) {
67        return EM_ID23_POSITION_GET_ERR_ADR_NOT_SET;
68    }
69
70    err = em_data_get_int(adr, 2, &time_num);
71    if (err < 0) {
72       return EM_ID23_POSITION_GET_ERR_ADR_NOT_SET;
73    }
74
75 /* verify latch on/off */
76 #ifndef EM_DEBUG_SIM
77    err = dev_ttldio_get_dobit(fd_0, dio_num, &dio_data);
78    if (err < 0) {
79        return EM_ID23_POSITION_GET_ERR;
80    }
81
82    if (dio_data == 0) {       /* latchon */
83        ct = 0;
84        do {        /* wait for latchoff */
85            err = dev_ttldio_get_dobit(fd_0, dio_num, &dio_data);
86            if (err < 0) {
87                return EM_ID23_POSITION_GET_ERR;
88            }
89
90            ++ct ;
91
92            if (ct > 100000) {
93                dio_data = 1 ;    /* forced latchoff */
94            }
95
96        } while (dio_data == 0);
97    }                         /* latchoff */
98
99    dio_data = 0;     /* latchon yourself */
100   err = dev_ttldio_put_dobit(fd_0, dio_num, &dio_data);
101   if (err < 0) {
102       return EM_ID23_POSITION_GET_ERR ;
103   }
104 #endif
105
106 /* sleep */
107   em_id23_util_delay_time(time_num);
108
109 #ifndef EM_DEBUG_SIM
110   err = dev_di_get_diall(fd_1, di_data);
111   if (err < 0) {
112       return EM_ID23_POSITION_GET_ERR;
113   }
114
115   dio_data = 1;      /* latchoff yourself */
116   err = dev_ttldio_put_dobit(fd_0, dio_num, &dio_data);
117   if (err < 0) {
118       return EM_ID23_POSITION_GET_ERR;
119   }
120 #endif
121
122 /* prepare output */
123   em_data_clear(out);
124   for (i = 0 ; i < 12 ; i++) {
125       err = em_data_put_int(out, i, di_data[i]);
126   }
127
128   return 0;
```

```
/*(@header_begin)*************************************************************
 *  (C Function)                                                               *
 *      Name = em_id23_rfbpm_position_get                                      *
 *      Type = int                                                             *
 *  (Purpose)                                                                  *
 *      read beam position                                                     *
 *  (Input)                                                                    *
 *      parameter (config.tbl)                                                 *
 *  (Output)                                                                   *
 *      out->data[0].eint : s_data                                             *
 *  (Return)                                                                   *
 *      0   : ok                                                               *
 *      < 0 : fail                                                             *
 *  (Physical unit)                                                            *
 *      Pa, V, A, mm, sec                                                      *
 *  (Relation)                                                                 *
 *                                                                             *
 *  (History)                                                                  *
 *      Jul-08-1997      F.Funayama /iST  created.                             *
 *      98- 1-11          Hiramatsu     revised                                *
 *      98-10-14          Hiramatsu     updated                                *
 *************************************************************(@header_end)*/

#include "em_id23.h"

int em_id23_rfbpm_position_get(Em_data in, Em_data adr, Em_data *out)
{
        int fd, err, count, rfbpm_num;
        int ave_num, samp_time;
        short rfbpm_num_s;
        unsigned short rfbpm_tmp[100];
        double  sdata;

#ifdef EM_DEBUG_PRINTF
        printf("Call func name is ¥"em_id23_rfbpm_position_get¥"¥n");
#endif

/* initialize */
        for (count=0 ; count < 100 ; count++)
            rfbpm_tmp[count] = 0;

/* get file descriptor */
        fd = em_get_fd(adr);

/* read config.tbl */
        err = em_data_get_int(adr, EM_ID23_RFBPM_POSITION_NUMBER, &rfbpm_num);
        if (err < 0) {
            return EM_ID23_RFBPM_POSITION_GET_ERR_ADR_NOT_SET;
        }
        rfbpm_num_s = (short)rfbpm_num;

        err = em_data_get_int(adr, EM_ID23_RFBPM_POSITION_AVE_NUM, &ave_num);
        if (err < 0) {
            return EM_ID23_RFBPM_POSITION_GET_ERR_ADR_NOT_SET;
        }
        err = em_data_get_int(adr, EM_ID23_RFBPM_POSITION_SAMP_TIM, &samp_time);

        if (err < 0) {
            return EM_ID23_RFBPM_POSITION_GET_ERR_ADR_NOT_SET;
        }

#ifndef EM_DEBUG_SIM
        err = dev_adc311_set_sched(fd, 1, &rfbpm_num_s,
              (unsigned short)ave_num, (unsigned short)samp_time, 0x00);
```

```
 1 /*(@header_begin)*************************************************************
 2 *  (C Function)                                                              *
 3 *      Name = em_id23_rfbpm_convmm_get                                       *
 4 *      Type = int                                                            *
 5 *  (Purpose)                                                                 *
 6 *      convert beam position to real value                                   *
 7 *  (Input)                                                                   *
 8 *      parameter (config.tbl)                                                *
 9 *  (Output)                                                                  *
10 *      real value                                                            *
11 *  (Return)                                                                  *
12 *      0   : ok                                                              *
13 *      < 0 : fail                                                            *
14 *  (Physical unit)                                                           *
15 *      Pa, V, A, mm, sec                                                     *
16 *  (Relation)                                                                *
17 *                                                                            *
18 *  (History)                                                                 *
19 *      98- 1-7         Hiramatsu         created                             *
20 *      98-10-14        Hiramatsu         updated                             *
21 *************************************************************(@header_end)*/
22
23 #include "em_id23.h"
24
25 int em_id23_rfbpm_convmm_get(Em_data in, Em_data conv, Em_svoc *out)
26 {
27    int err;
28    double ai_data, y_data, a_data, b_data;
29
30 #ifdef   EM_DEBUG_PRINTF
31    printf("Call func name is ¥"em_id23_rfbpm_convmm_get¥"¥n");
32 #endif
33
34 /* get rfbpm data */
35    err = em_data_get_double(in, EM_ID23_RFBPM_CONV_GET_AI_DATA, &ai_data);
36    if (err < 0) {
37        return EM_ID23_RFBPM_CONV_GET_ERR ;
38    }
39
40 /* read config.tbl */
41    err = em_data_get_double(conv, EM_ID23_RFBPM_CONV_GET_VALUE_1, &a_data);
42    if (err < 0) {
43        return EM_ID23_RFBPM_CONV_GET_ERR_CNV_NOT_SET;
44    }
45
46    err = em_data_get_double(conv, EM_ID23_RFBPM_CONV_GET_VALUE_2, &b_data);
47    if (err < 0) {
48        return EM_ID23_RFBPM_CONV_GET_ERR_CNV_NOT_SET;
49    }
50
51 /* convert */
52    y_data = a_data * ai_data + b_data ;
53
54 /* prepare output */
55    em_data_clear(out);
56
57    if (!err) {
58        sprintf(out->svoc, "%lfmm", y_data);
59    } else {
60        sprintf(out->svoc, "fail");
61    }
62
63    return 0 ;
64 }
```

```
 1 /*(@header_begin)*************************************************************
 2 *  (C Function)                                                              *
 3 *       Name = em_id23_rfbpm_conv_get                                        *
 4 *       Type = int                                                           *
 5 *  (Purpose)                                                                 *
 6 *      convert beam position to real value                                   *
 7 *  (Input)                                                                   *
 8 *      parameter (config.tbl)                                                *
 9 *  (Output)                                                                  *
10 *      real value                                                            *
11 *  (Return)                                                                  *
12 *      0   : ok                                                              *
13 *      < 0 : fail                                                            *
14 *  (Physical unit)                                                           *
15 *      Pa, V, A, mm, sec                                                     *
16 *  (Relation)                                                                *
17 *                                                                            *
18 *  (History)                                                                 *
19 *      Jul-08-1997       F.Funayama /iST  created.                           *
20 *      97-12-22          Hiramatsu  checked for unit(V)                      *
21 *      98-10-14          Hiramatsu  updated                                  *
22 *************************************************************(@header_end)*/
23
24 #include "em_id23.h"
25
26 int em_id23_rfbpm_conv_get(Em_data in, Em_data conv, Em_svoc *out)
27 {
28    int err;
29    double ai_data, y_data, a_data, b_data;
30
31 #ifdef EM_DEBUG_PRINTF
32    printf("Call func name is ¥"em_id23_rfbpm_conv_get¥"¥n");
33 #endif
34
35 /* get rfbpm data */
36    err = em_data_get_double(in, EM_ID23_RFBPM_CONV_GET_AI_DATA, &ai_data);
37    if (err < 0) {
38        return EM_ID23_RFBPM_CONV_GET_ERR;
39    }
40
41 /* read config.tbl */
42    err = em_data_get_double(conv, EM_ID23_RFBPM_CONV_GET_VALUE_1, &a_data);
43    if (err < 0) {
44        return EM_ID23_RFBPM_CONV_GET_ERR_CNV_NOT_SET;
45    }
46
47    err = em_data_get_double(conv, EM_ID23_RFBPM_CONV_GET_VALUE_2, &b_data);
48    if (err < 0) {
49        return EM_ID23_RFBPM_CONV_GET_ERR_CNV_NOT_SET;
50    }
51
52 /* convert */
53    y_data = a_data * ai_data + b_data ;
54
55 /* prepare output */
56    em_data_clear(out);
57
58    if (!err) {
59        sprintf(out->svoc, "%lfV", y_data);
60    } else {
61        sprintf(out->svoc, "fail");
62    }
63
64    return 0 ;
```

```
65          if (err < 0) {
66              return EM_ID23_RFBPM_POSITION_GET_ERR ;
67          }
68
69          err = dev_adc311_conv(fd, rfbpm_tmp, EM_ID23_TIME_OUT_NUM);
70          if (err < 0) {
71              return EM_ID23_RFBPM_POSITION_GET_ERR ;
72          }
73 #endif
74
75 /* get rfbpm position */
76          sdata = 0.0;
77          for (count = 0 ; count < ave_num ; count++) {
78              sdata += (int)rfbpm_tmp[count];
79          }
80
81 /* prepare output */
82          em_data_clear(out);
83          em_data_put_double( out, 0, (double)sdata / ((double)ave_num) );
84
85          return 0;
86 }
```

```
 1 /*(@header_begin)*************************************************************
 2 *  (C Function)                                                              *
 3 *      Name = em_id23_rfbpm_status_get                                       *
 4 *      Type = int                                                            *
 5 *  (Purpose)                                                                 *
 6 *      read device status                                                    *
 7 *  (Input)                                                                   *
 8 *      parameter (config.tbl)                                                *
 9 *  (Output)                                                                  *
10 *      out->data[0].eint : di_data                                           *
11 *  (Return)                                                                  *
12 *      0   : ok                                                              *
13 *      < 0 : fail                                                            *
14 *  (Physical unit)                                                           *
15 *      Pa, V, A, mm, sec                                                     *
16 *  (Relation)                                                                *
17 *                                                                            *
18 *  (History)                                                                 *
19 *      Jul-15-1997       F.Funayama /iST  created.                           *
20 *      98-1-11           Hiramatsu    revised                                *
21 *      98-10-14          Hiramatsu    updated                                *
22 *************************************************************(@header_end)*/
23
24 #include "em_id23.h"
25
26 int em_id23_rfbpm_status_get(Em_data in, Em_data adr, Em_data *out)
27 {
28         int fd, dev_num, di_data, err;
29
30 #ifdef EM_DEBUG_PRINTF
31         printf("Call func name is ¥"em_id23_rfbpm_status_get¥"¥n");
32 #endif
33
34 /* get file descriptor */
35         fd = em_get_fd(adr);
36
37 /* read config.tbl */
38         err = em_data_get_int(adr, EM_ID23_RFBPM_STATUS_INPUT_PORT, &dev_num);
39         if (err < 0) {
40             return EM_ID23_RFBPM_STATUS_GET_ERR_ADR_NOT_SET;
41         }
42
43 #ifndef EM_DEBUG_SIM
44         err = dev_ttldio_get_diport(fd, dev_num, &di_data);
45         if (err < 0) {
46             return EM_ID23_RFBPM_STATUS_GET_ERR ;
47         }
48 #else
49         di_data = 0x0 ;
50 #endif
51
52         di_data &= EM_ID23_RFBPM_STATUS_CHECK;
53
54 /* prepare output */
55         em_data_clear(out);
56         em_data_put_int(out, 0, di_data);
57
58         return 0 ;
59 }
```

```
 1 /*(@header_begin)*************************************************************
 2 *  (C Function)                                                              *
 3 *      Name = em_id23_rio_init                                               *
 4 *      Type = int                                                            *
 5 *  (Purpose)                                                                 *
 6 *      initialize RIO master for ID23                                        *
 7 *  (Input)                                                                   *
 8 *      none                                                                  *
 9 *  (Output)                                                                  *
10 *      none                                                                  *
11 *  (Return)                                                                  *
12 *      0   : ok                                                              *
13 *      < 0 : fail                                                            *
14 *  (Physical unit)                                                           *
15 *      Pa, V, A, mm, sec                                                     *
16 *  (Relation)                                                                *
17 *                                                                            *
18 *  (History)                                                                 *
19 *      Sep-20-1995       A.Taketani / SPring8      created.                  *
20 *      Jun-17-1996       A.Taketani EM_POLLER added                          *
21 *      20-Nov-1996       T.Fukui/SPring-8 Modified for bl_id23.              *
22 *      98-01-11          Hiramatsu      adapted for ID23                     *
23 *************************************************************(@header_end)*/
24
25 #include "em_id23.h"
26
27 int em_id23_rio_init()
28 {
29     static int flag = 0;
30     int err, mode, fd;
31     char device_name[EM_MIX_DATA_CHAR_MAX];
32
33 #ifdef EM_DEBUG_PRINTF
34     printf("Call func name is ¥"em_id23_rio_init¥"¥n");
35 #endif
36
37 #ifndef EM_DEBUG_SIM
38     if (flag == 0) {
39         device_name[0] = '¥0';
40         mode = EM_VAR_GET_ALL_FIRST;
41         while (em_save_fd(device_name, mode, &fd) != EM_SAVE_FD_END) {
42             mode = EM_VAR_GET_ALL;
43             if (strstr(device_name, "/dev/rio_") != NULL) {
44                 err = dev_rio_init(fd, 0);
45                 if (err != 0) {
46                     return EM_ID23_RIO_INIT_ERR;
47                 }
48                 flag = 1 ;
49             }
50         }
51     }
52 #endif
53
54     return 0;
55 }
```

```
/*(@header_begin)************************************************************
*  (C Function)                                                              *
*      Name = em_id23_rio_terminate                                          *
*      Type = int                                                            *
*  (Purpose)                                                                 *
*      terminate RIO master for ID23                                         *
*  (Input)                                                                   *
*      none                                                                  *
*  (Output)                                                                  *
*      none                                                                  *
*  (Return)                                                                  *
*      0   : ok                                                              *
*      < 0 : fail                                                            *
*  (Physical unit)                                                           *
*      Pa, V, A, mm, sec                                                     *
*  (Relation)                                                                *
*                                                                            *
*  (History)                                                                 *
*      Aug-14-1996      A.Taketani / SPring8      created.                   *
*      20-Nov-1996      T.Fukui/SPring-8 Modified for bl_id23.               *
*      98-1-11          Hiramatsu     adapted for ID23                       *
************************************************************(@header_end)*/

#include "em_id23.h"

int em_id23_rio_terminate()

{

#ifdef EM_DEBUG_PRINTF
   printf("Call func name is ¥"em_id23_rio_terminate¥"¥n");
#endif

/********** cancelled because of dev_rio_start ******
#ifndef EM_DEBUG_SIM
   int err, mode, fd;
   char device_name[EM_MIX_DATA_CHAR_MAX];
   device_name[0] = '¥0';
   mode = EM_VAR_GET_ALL_FIRST;

   while (em_save_fd(device_name, mode, &fd) != EM_SAVE_FD_END) {
      mode = EM_VAR_GET_ALL;
      if (strstr(device_name, "/dev/rio_") != NULL) {
        err = dev_rio_wait(fd);
        if (err != 0) {
            return EM_ID23_RIO_TERMINATE_ERR;
        }
      }
   }
#endif
*************************************************************/

   return 0;
}
```

```
/*(@header_begin)*************************************************************
 *  (C Function)                                                             *
 *      Name = em_id23_sip_status_get                                        *
 *      Type = int                                                           *
 *  (Purpose)                                                                *
 *      read device status                                                   *
 *  (Input)                                                                  *
 *      parameter (config.tbl)                                               *
 *  (Output)                                                                 *
 *      out->data[0].eint : status                                           *
 *  (Return)                                                                 *
 *      0   : ok                                                             *
 *      < 0 : fail                                                           *
 *  (Physical unit)                                                          *
 *      Pa, V, A, mm, sec                                                    *
 *  (Relation)                                                               *
 *                                                                           *
 *  (History)                                                                *
 *      Jul-16-1997      F.Funayama /iST  created.                           *
 *      98- 1-11         Hiramatsu       revised                             *
 *      98-10-14         Hiramatsu       updated                             *
 *************************************************************(@header_end)*/

#include "em_id23.h"

int em_id23_sip_status_get(Em_data in, Em_data adr, Em_data *out)
{
        int fd, dev_num, err, di_data, status = 0 ;

#ifdef EM_DEBUG_PRINTF
        printf("Call func name is ¥"em_id23_sip_status_get¥"¥n");
#endif

/* get file descriptor */
        fd = em_get_fd(adr);

/* read config.tbl */
        err = em_data_get_int(adr, EM_ID23_SIP_STATUS_INPUT_PORT, &dev_num);
        if (err < 0) {
            return EM_ID23_SIP_STATUS_GET_ERR_ADR_NOT_SET;
        }

#ifndef EM_DEBUG_SIM
        err = dev_di_get_diport(fd, dev_num, &di_data);
        if (err < 0) {
            return EM_ID23_SIP_STATUS_GET_ERR ;
        }
#else
        di_data = 0x23;
#endif

        status = di_data & 0x3f;

/* prepare output */
        em_data_clear(out);
        em_data_put_int(out, 0, status);

        return 0;
}
```

```
1 /*(@header_begin)*************************************************************
2 *  (C Function)                                                               *
3 *      Name = em_id23_st_adc_curr_get                                         *
4 *      Type = int                                                             *
5 *  (Purpose)                                                                  *
6 *      read ADC current                                                       *
7 *  (Input)                                                                    *
8 *      parameter (config.tbl)                                                 *
9 *  (Output)                                                                   *
10 *      out->data[0].eint : ai_data                                           *
11 *  (Return)                                                                  *
12 *      0   : ok                                                              *
13 *      < 0 : fail                                                            *
14 *  (Physical unit)                                                           *
15 *      Pa, V, A, mm, sec                                                     *
16 *  (Relation)                                                                *
17 *                                                                            *
18 *  (History)                                                                 *
19 *      Jul-15-1997      F.Funayama /iST  created.                            *
20 *      98- 1-11          Hiramatsu       revised                            *
21 *      98-10-15          Hiramatsu       updated                            *
22 *************************************************************(@header_end)*/
23
24 #include "em_id23.h"
25
26 int em_id23_st_adc_curr_get(Em_data in, Em_data adr, Em_data *out)
27 {
28         int fd, err, slave;
29         int ai_data;
30
31 #ifdef EM_DEBUG_PRINTF
32         printf("Call func name is ¥"em_id23_st_adc_curr_get¥"¥n");
33 #endif
34
35 /* get file descriptor */
36         fd = em_get_fd(adr);
37
38 /* read config.tbl */
39         err = em_data_get_int(adr, EM_ID23_ST_ADC_CURR_GET_SLAVE, &slave);
40         if (err < 0) {
41             return EM_ID23_ST_ADC_CURR_GET_ERR_ADR_NOT_SET;
42         }
43
44 /* get ADC current */
45 #ifndef EM_DEBUG_SIM
46         err = dev_rioA_get_ai(fd, slave, &ai_data);
47         if (err < 0) {
48             return EM_ID23_ST_ADC_CURR_GET_ERR;
49         }
50 #endif
51
52 /* prepare output */
53         em_data_clear(out);
54         em_data_put_int(out, 0, ai_data);
55
56         return 0;
57 }
```

```
 1 /*(@header_begin)************************************************************
 2 *  (C Function)                                                             *
 3 *      Name = em_id23_st_curr_conv_get                                      *
 4 *      Type = int                                                           *
 5 *  (Purpose)                                                                *
 6 *      convert current(digital) to real value                               *
 7 *  (Input)                                                                  *
 8 *      parameter (config.tbl)                                               *
 9 *  (Output)                                                                 *
10 *      real value                                                           *
11 *  (Return)                                                                 *
12 *      0   : ok                                                             *
13 *      < 0 : fail                                                           *
14 *  (Physical unit)                                                          *
15 *      Pa, V, A, mm, sec                                                    *
16 *  (Relation)                                                               *
17 *                                                                           *
18 *  (History)                                                                *
19 *      Jul-15-1997      F.Funayama /iST  created.                           *
20 *      97-12-22         Hiramatsu  checked for unit(A)                      *
21 *      98-10-15         Hiramatsu  updated                                  *
22 ************************************************************(@header_end)*/
23
24 #include "em_id23.h"
25
26 int em_id23_st_curr_conv_get(Em_data in, Em_data conv, Em_svoc *out)
27 {
28    int err, ai_data;
29    double y_data, a_data, b_data;
30
31 #ifdef EM_DEBUG_PRINTF
32    printf("Call func name is ¥"em_id23_st_curr_conv_get¥"¥n");
33 #endif
34
35 /* get current */
36    err = em_data_get_int(in, EM_ID23_ST_CURR_CONV_GET_AI_DATA, &ai_data);
37    if (err < 0) {
38       return EM_ID23_ST_CURR_CONV_GET_ERR ;
39    }
40
41 /* read config.tbl */
42    err = em_data_get_double(conv, EM_ID23_ST_CURR_CONV_GET_VALUE_1, &a_data);
43    if (err < 0) {
44        return EM_ID23_ST_CURR_CONV_GET_ERR_CNV_NOT_SET;
45    }
46
47    err = em_data_get_double(conv, EM_ID23_ST_CURR_CONV_GET_VALUE_2, &b_data);
48    if (err < 0) {
49        return EM_ID23_ST_CURR_CONV_GET_ERR_CNV_NOT_SET;
50    }
51
52 /* convert */
53    y_data = a_data * (double)ai_data + b_data;
54
55 /* prepare output */
56    em_data_clear(out);
57
58    if (err == 0) {
59        sprintf(out->svoc, "%lfA", y_data);
60    } else {
61        sprintf(out->svoc, "fail");
62    }
63
64    return 0;
```

```
 1 /*(@header_begin)*********************************************************
 2 *  (C Function)                                                           *
 3 *      Name = em_id23_st_curr_exec                                        *
 4 *      Type = int                                                         *
 5 *  (Purpose)                                                              *
 6 *      get current from cellar, then convert to digital value             *
 7 *  (Input)                                                                *
 8 *      parameter (config.tbl)                                             *
 9 *  (Output)                                                               *
10 *      current (digital)                                                  *
11 *  (Return)                                                               *
12 *      0   : ok                                                           *
13 *      < 0 : fail                                                         *
14 *  (Physical unit)                                                        *
15 *      Pa, V, A, mm, sec                                                  *
16 *  (Relation)                                                             *
17 *                                                                         *
18 *  (History)                                                              *
19 *      Jul-13-1997      F.Funayama /iST  created.                         *
20 *      98-01-11         Hiramatsu       revised                           *
21 *      98-10-15         Hiramatsu       updated                           *
22 *      99-07-05         Hiramatsu       join em_id23_lc_curr_exec         *
23 *********************************************************(@header_end)*/
24
25 #include "em_id23.h"
26
27 int em_id23_st_curr_exec(Em_data in, Em_data conv, Em_data *out)
28 {
29    int err, i;
30    double y_data, a_data, b_data;
31    Em_id23_current curr;
32
33 #ifdef  EM_DEBUG_PRINTF
34    printf("Call func name is ¥"em_id23_st_curr_exec¥"¥n");
35 #endif
36
37 /* read config.tbl */
38    err = em_data_get_double(conv, EM_ID23_ST_CURR_EXEC_CONV_VALUE_1, &a_data);
39    if (err < 0) {
40       return EM_ID23_ST_CURR_ERR_CONV_NOT_SET;
41    }
42
43    err = em_data_get_double(conv, EM_ID23_ST_CURR_EXEC_CONV_VALUE_2, &b_data);
44    if (err < 0) {
45        return EM_ID23_ST_CURR_ERR_CONV_NOT_SET;
46    }
47
48 /* prepare output */
49    em_data_clear(out);
50
51 /* get current from cellar */
52    for (i = 0 ; i < EM_ID23_ST_NUM ; i++) {
53       em_id23_util_st_curr_cellar(EM_ID23_GET_CURR, i, &curr);
54       if (curr.set_flg == 0) {
55           return EM_ID23_ST_CURR_ERR_DATA_NOT_SET;
56       }
57
58       y_data = a_data * curr.current_now + b_data;
59       curr.current_digital = (int)y_data;
60       em_data_put_int(out, i, curr.current_digital);
61    }
62
63    return 0;
64 }
```

```
 1 /*(@header_begin)*************************************************************
 2 *  (C Function)                                                              *
 3 *      Name = em_id23_st_curr_put                                            *
 4 *      Type = int                                                            *
 5 *  (Purpose)                                                                 *
 6 *      put current to long-steering and AC-steering magnet                   *
 7 *  (Input)                                                                   *
 8 *      parameter (config.tbl)                                                *
 9 *  (Output)                                                                  *
10 *      none                                                                  *
11 *  (Return)                                                                  *
12 *      0   : ok                                                              *
13 *      < 0 : fail                                                            *
14 *  (Physical unit)                                                           *
15 *      Pa, V, A, mm, sec                                                     *
16 *  (Relation)                                                                *
17 *                                                                            *
18 *  (History)                                                                 *
19 *      Aug- 8-1995      A.Taketani / SPring8      created.                   *
20 *      Jul-13-1997      H.Funayama / iST Modified for bl_id23.               *
21 *      98-01-11         Hiramatsu   revised                                  *
22 *      98-10-15         Hiramatsu   updated                                  *
23 *      99-03-16         Hiramatsu   deleted util_st_curr_direct_set.c        *
24 *      99-07-05         Hiramatsu   join em_id23_lc_curr_put                  *
25 *************************************************************(@header_end)*/
26
27 #include "em_id23.h"
28
29 int em_id23_st_curr_put(Em_data in, Em_data adr, Em_data *out)
30 {
31    int fd, i, err, slave_num[EM_ID23_ST_NUM], d_curr;
32
33 #ifdef  EM_DEBUG_PRINTF
34    printf("Call func name is ¥"em_id23_st_curr_put¥"¥n");
35 #endif
36
37 /* get file descriptor */
38    fd = em_get_fd(adr);
39
40 /* read config.tbl */
41    for (i = 0 ; i < EM_ID23_ST_NUM ; i++) {
42       err = em_data_get_int(adr, i+1 , &(slave_num[i]));
43       if (err < 0) {
44           return EM_ID23_ST_CURR_ERR_ADR_NOT_SET;
45       }
46    }
47
48 /* put current to RIO */
49    for (i = 0 ; i < EM_ID23_ST_NUM ; i++) {
50       em_data_get_int(in, i, &d_curr);
51
52 #ifndef EM_DEBUG_SIM
53       err = dev_rioA_put_ao(fd, slave_num[i], &d_curr);
54       if (err < 0) {
55           return EM_ID23_ST_CURR_PUT_ERR;
56       }
57 #endif
58    }
59
60 /* prepare output */
61    em_data_clear(out);
62    em_data_put_int(out, 0, 0);
63
64    return 0;
```

```
 1 /*(@header_begin)*************************************************************
 2 *  (C Function)                                                              *
 3 *      Name = em_id23_st_curr_set                                            *
 4 *      Type = int                                                            *
 5 *  (Purpose)                                                                 *
 6 *      set current to long-steering and AC-steering magnet                   *
 7 *  (Input)                                                                   *
 8 *      parameter (config.tbl)                                                *
 9 *  (Output)                                                                  *
10 *      none                                                                  *
11 *  (Return)                                                                  *
12 *      0   : ok                                                              *
13 *      < 0 : fail                                                            *
14 *  (Physical unit)                                                           *
15 *      Pa, V, A, mm, sec                                                     *
16 *  (Relation)                                                                *
17 *                                                                            *
18 *  (History)                                                                 *
19 *      Jul-13-1997       H.Funayama / iST          created.                  *
20 *      98-01-11          Hiramatsu     revised                               *
21 *      98-10-15          Hiramatsu     updated                               *
22 *      99-07-05          Hiramatsu     join em_id23_lc_curr_put              *
23 *************************************************************(@header_end)*/
24
25 #include "em_id23.h"
26
27 int em_id23_st_curr_set(Em_data in, Em_data adr, Em_data *out)
28 {
29    int err, st_num;
30    Em_id23_current curr;
31    curr.set_flg = 0;
32
33 #ifdef EM_DEBUG_PRINTF
34    printf("Call func name is ¥"em_id23_st_curr_set¥"¥n");
35 #endif
36
37 /* get current */
38    em_data_get_double(in, EM_ID23_ST_CURR_SET_DATA, &(curr.current_now));
39    if (curr.current_now > 14.9) {
40        return EM_ID23_ST_CURR_SET_ERR_15A_OVER;
41    } else if (curr.current_now < -14.9) {
42        return EM_ID23_ST_CURR_SET_ERR_15A_OVER;
43    }
44
45 /* read config.tbl */
46    err = em_data_get_int(adr, EM_ID23_ST_CURR_SET_ST_NUM, &st_num);
47    if (err < 0) {
48        return EM_ID23_ST_CURR_SET_ERR_ADR_NOT_SET;
49    }
50
51 /* set current to long-steering and AC-steering magnet */
52    err = em_id23_util_st_curr_cellar(EM_ID23_PUT_CURR, st_num, &curr);
53    if (err < 0) {
54        return EM_ID23_ST_CURR_SET_ERR;
55    }
56
57    if (curr.set_flg == 0) {
58        return EM_ID23_ST_CELLAR_NOT_PUT;
59    }
60
61 /* prepare output */
62    em_data_clear(out);
63    em_data_put_int(out, 0, 0);
64
65    return 0;
66 }
```

```
/*(@header_begin)*************************************************************
*  (C Function)                                                               *
*      Name = em_id23_st_dac_curr_get                                         *
*      Type = int                                                             *
*  (Purpose)                                                                  *
*      read DAC current                                                       *
*  (Input)                                                                    *
*      parameter (config.tbl)                                                 *
*  (Output)                                                                   *
*      DAC current                                                            *
*  (Return)                                                                   *
*      0   : ok                                                               *
*      < 0 : fail                                                             *
*  (Physical unit)                                                            *
*      Pa, V, A, mm, sec                                                      *
*  (Relation)                                                                 *
*                                                                             *
*  (History)                                                                  *
*      Jul-15-1997      F.Funayama /iST  created.                             *
*      98- 1-11          Hiramatsu       revised                              *
*      98-10-15          Hiramatsu       updated                              *
*************************************************************(@header_end)*/

#include "em_id23.h"

int em_id23_st_dac_curr_get(Em_data in, Em_data adr, Em_data *out)
{
        int fd, err, slave, ai_data;

#ifdef EM_DEBUG_PRINTF
        printf("Call func name is ¥"em_id23_st_dac_curr_get¥"¥n");
#endif

/* get file descriptor */
        fd = em_get_fd(adr);

/* read config.tbl */
        err = em_data_get_int(adr, EM_ID23_ST_DAC_CURR_GET_SLAVE, &slave);
        if (err < 0) {
            return EM_ID23_ST_DAC_CURR_GET_ERR_ADR_NOT_SET;
        }

/* get current */
#ifndef EM_DEBUG_SIM
        err = dev_rioA_get_ao(fd, slave, &ai_data);
        if (err < 0) {
            return EM_ID23_ST_DAC_CURR_GET_ERR;
        }
#endif

/* prepare output */
        em_data_clear(out);
        em_data_put_int(out, 0, ai_data);

        return 0;
}
```

```
 1 /*(@header_begin)*************************************************************
 2 *  (C Function)                                                              *
 3 *      Name = em_id23_st_err_reset                                           *
 4 *      Type = int                                                            *
 5 *  (Purpose)                                                                 *
 6 *      reset RIO device(error reset)                                         *
 7 *  (Input)                                                                   *
 8 *      parameter (config.tbl)                                                *
 9 *  (Output)                                                                  *
10 *      none                                                                  *
11 *  (Return)                                                                  *
12 *      0   : ok                                                              *
13 *      < 0 : fail                                                            *
14 *  (Physical unit)                                                           *
15 *      Pa, V, A, mm, sec                                                     *
16 *  (Relation)                                                                *
17 *                                                                            *
18 *  (History)                                                                 *
19 *      Jul-15-1997      F.Funayama /iST  created.                            *
20 *      98- 1-11          Hiramatsu      revised                             *
21 *      98-10-15          Hiramatsu      updated                             *
22 *************************************************************(@header_end)*/
23
24 #include "em_id23.h"
25
26 int em_id23_st_err_reset(Em_data in, Em_data adr, Em_data *out)
27 {
28     int fd, err, slave;
29     int dev_num;           /* reset bit                          */
30     int time_num;          /* delay time [microsec]              */
31     int do_data;           /* activate(=1) or desactivate(=0) */
32
33 #ifdef  EM_DEBUG_PRINTF
34     printf("Call func name is ¥"em_id23_st_err_reset¥"¥n");
35 #endif
36
37 /* get file descriptor */
38     fd = em_get_fd(adr);
39
40 /* read config.tbl */
41     err = em_data_get_int(adr, EM_ID23_ST_ERR_RESET_SLAVE, &slave);
42     if (err < 0) {
43         return EM_ID23_ST_ERR_RESET_ERR_ADR_NOT_SET;
44     }
45
46     err = em_data_get_int(adr, EM_ID23_ST_ERR_RESET_BIT, &dev_num);
47     if (err < 0) {
48         return EM_ID23_ST_ERR_RESET_ERR_ADR_NOT_SET;
49     }
50
51     err = em_data_get_int(adr, EM_ID23_ST_ERR_RESET_DELAY, &time_num);
52     if (err < 0) {
53         return EM_ID23_ST_ERR_RESET_ERR_ADR_NOT_SET;
54     }
55
56 /* activate reset bit */
57     do_data = 1;
58
59 /* reset on */
60 #ifndef EM_DEBUG_SIM
61     err = dev_rioA_put_dobit(fd, slave, dev_num, &do_data);
62     if (err < 0) {
63         return EM_ID23_ST_ERR_RESET_ERR;
64     }
```

```
65 #endif
66
67 /* sleep */
68     usleep(time_num);
69
70 /* desactivate reset bit */
71     do_data = 0;
72
73 /* reset off */
74 #ifndef EM_DEBUG_SIM
75     err = dev_rioA_put_dobit(fd, slave, dev_num, &do_data);
76     if (err < 0) {
77         return EM_ID23_ST_ERR_RESET_ERR;
78     }
79 #endif
80
81 /* prepare output */
82     em_data_clear(out);
83     em_data_put_int(out, 0, 0);
84
85     return 0;
86 }
```

```
 1 /*(@header_begin)*********************************************************
 2 *  (C Function)                                                           *
 3 *      Name = em_id23_st_power_off_put                                    *
 4 *      Type = int                                                         *
 5 *  (Purpose)                                                              *
 6 *      power off and put current 0[A]                                     *
 7 *  (Input)                                                                *
 8 *      parameter (config.tbl)                                             *
 9 *  (Output)                                                               *
10 *      none                                                               *
11 *  (Return)                                                               *
12 *      0   : ok                                                           *
13 *      < 0 : fail                                                         *
14 *  (Physical unit)                                                        *
15 *      Pa, V, A, mm, sec                                                  *
16 *  (Relation)                                                             *
17 *                                                                         *
18 *  (History)                                                              *
19 *      Jul-15-1997      F.Funayama /iST  created.                         *
20 *      98- 1-11          Hiramatsu       revised                          *
21 *      98-10-15          Hiramatsu       updated                          *
22 *********************************************************(@header_end)*/
23
24 #include "em_id23.h"
25
26 int em_id23_st_power_off_put(Em_data in, Em_data adr, Em_data *out)
27 {
28     int err, fd, slave;
29     int dev_num;            /* power-off bit                    */
30     int time_num ;          /* delay time [microsec]            */
31     int d_curr;             /* current 0[A]                     */
32     int do_data;            /* activate(=1) or desactivate(=0) */
33
34 #ifdef  EM_DEBUG_PRINTF
35     printf("Call func name is ¥"em_id23_st_power_off_put¥"¥n");
36 #endif
37
38 /* get file descriptor */
39     fd = em_get_fd(adr);
40
41 /* read config.tbl */
42     err = em_data_get_int(adr, EM_ID23_ST_POWER_PUT_SLAVE, &slave);
43     if (err < 0) {
44         return EM_ID23_ST_POWER_OFF_PUT_ERR_ADR_NOT_SET;
45     }
46
47     err = em_data_get_int(adr, EM_ID23_ST_POWER_PUT_BIT, &dev_num);
48     if (err < 0) {
49         return EM_ID23_ST_POWER_OFF_PUT_ERR_ADR_NOT_SET;
50     }
51
52     err = em_data_get_int(adr, EM_ID23_ST_POWER_PUT_DELAY, &time_num);
53     if (err < 0) {
54        return EM_ID23_ST_POWER_OFF_PUT_ERR_ADR_NOT_SET;
55     }
56
57     err = em_data_get_int(adr, EM_ID23_ST_POWER_PUT_ODIGIT, &d_curr);
58     if (err < 0) {
59         return EM_ID23_ST_POWER_OFF_PUT_ERR_ADR_NOT_SET;
60     }
61
62 #ifndef EM_DEBUG_SIM
63 /* put 0 Ampere to steering magnet */
64     err = dev_rioA_put_ao(fd, slave, &d_curr);
```

```
65      if (err < 0) {
66          return EM_ID23_ST_POWER_OFF_PUT_ERR;
67      }
68  #endif
69
70  /* activate power-off bit */
71      do_data = 1;
72
73  #ifndef EM_DEBUG_SIM
74      err = dev_rioA_put_dobit(fd, slave, dev_num, &do_data);
75      if (err < 0) {
76          return EM_ID23_ST_POWER_OFF_PUT_ERR;
77      }
78  #endif
79
80  /* sleep */
81      usleep(time_num);
82
83  /* desactivate power-off bit */
84      do_data = 0;
85
86  #ifndef EM_DEBUG_SIM
87      err = dev_rioA_put_dobit(fd, slave, dev_num, &do_data);
88      if (err < 0) {
89         return EM_ID23_ST_POWER_OFF_PUT_ERR;
90      }
91  #endif
92
93  /* prepare output */
94      em_data_clear(out);
95      em_data_put_int(out, 0, 0);
96
97      return 0;
98  }
```

```
/*(@header_begin)*************************************************************
*  (C Function)                                                               *
*      Name = em_id23_st_power_on_put                                         *
*      Type = int                                                             *
*  (Purpose)                                                                  *
*      put 0[A] to steering and power-on                                      *
*  (Input)                                                                    *
*      parameter (config.tbl)                                                 *
*  (Output)                                                                   *
*      none                                                                   *
*  (Return)                                                                   *
*      0   : ok                                                               *
*      < 0 : fail                                                             *
*  (Physical unit)                                                            *
*      Pa, V, A, mm, sec                                                      *
*  (Relation)                                                                 *
*                                                                             *
*  (History)                                                                  *
*      Jul-15-1997      F.Funayama /iST  created.                             *
*      98- 1-11          Hiramatsu       revised                              *
*      98-10-15          Hiramatsu       updated                              *
*****************************************************************(@header_end)*/

#include "em_id23.h"

int em_id23_st_power_on_put(Em_data in, Em_data adr, Em_data *out)
{
    int err, fd, slave;
    int dev_num;          /* power-on bit                      */
    int time_num;         /* delay time [microsec]             */
    int d_curr;           /* current 0[A]                      */
    int do_data;          /* activate(=1) or desactivate(=0)   */

#ifdef  EM_DEBUG_PRINTF
    printf("Call func name is ¥"em_id23_st_power_on_put¥"¥n");
#endif

/* get file descriptor */
    fd = em_get_fd(adr);

/* read config.tbl */
    err = em_data_get_int(adr, EM_ID23_ST_POWER_PUT_SLAVE, &slave);
    if (err < 0) {
        return EM_ID23_ST_POWER_ON_PUT_ERR_ADR_NOT_SET;
    }

    err = em_data_get_int(adr, EM_ID23_ST_POWER_PUT_BIT, &dev_num);
    if (err < 0) {
        return EM_ID23_ST_POWER_ON_PUT_ERR_ADR_NOT_SET;
    }

    err = em_data_get_int(adr, EM_ID23_ST_POWER_PUT_DELAY, &time_num);
    if (err < 0) {
    return EM_ID23_ST_POWER_ON_PUT_ERR_ADR_NOT_SET;
    }

    err = em_data_get_int(adr, EM_ID23_ST_POWER_PUT_ODIGIT, &d_curr);
    if (err < 0) {
        return EM_ID23_ST_POWER_ON_PUT_ERR_ADR_NOT_SET;
    }

/* put current 0[A] to steering magnet */
#ifndef EM_DEBUG_SIM
    err = dev_rioA_put_ao(fd, slave, &d_curr);
```

```
65      if (err < 0) {
66          return EM_ID23_ST_POWER_ON_PUT_ERR;
67      }
68  #endif
69
70  /* activate power-on bit */
71      do_data = 1;
72
73  #ifndef EM_DEBUG_SIM
74      err = dev_rioA_put_dobit(fd, slave, dev_num, &do_data);
75      if (err < 0) {
76          return EM_ID23_ST_POWER_ON_PUT_ERR;
77      }
78  #endif
79
80  /* sleep */
81      usleep(time_num);
82
83  /* desactivate power-on bit */
84      do_data = 0;
85
86  #ifndef EM_DEBUG_SIM
87      err = dev_rioA_put_dobit(fd, slave, dev_num, &do_data);
88      if (err < 0) {
89          return EM_ID23_ST_POWER_ON_PUT_ERR;
90      }
91  #endif
92
93  /* prepare output */
94      em_data_clear(out);
95      em_data_put_int(out, 0, 0);
96
97      return 0 ;
98  }
```

```
 1 /*(@header_begin)*************************************************************
 2 *  (C Function)                                                              *
 3 *      Name = em_id23_st_status_get                                          *
 4 *      Type = int                                                            *
 5 *  (Purpose)                                                                 *
 6 *      read device status                                                    *
 7 *  (Input)                                                                   *
 8 *      parameter (config.tbl)                                                *
 9 *  (Output)                                                                  *
10 *      out->data[0].eint : status                                            *
11 *  (Return)                                                                  *
12 *      0   : ok                                                              *
13 *      < 0 : fail                                                            *
14 *  (Physical unit)                                                           *
15 *      Pa, V, A, mm, sec                                                     *
16 *  (Relation)                                                                *
17 *                                                                            *
18 *  (History)                                                                 *
19 *      Jul-15-1997      F.Funayama /iST  created.                            *
20 *      98- 1-11          Hiramatsu       revised                             *
21 *      98-10-15          Hiramatsu       updated                             *
22 *      99-03-16          Hiramatsu       join _st_status_conv.c              *
23 *************************************************************(@header_end)*/
24
25 #include "em_id23.h"
26
27 int em_id23_st_status_get(Em_data in, Em_data adr, Em_data *out)
28 {
29         int fd, slave, di_data , st_status = 0 , err;
30
31 #ifdef EM_DEBUG_PRINTF
32         printf("¥"em_id23_st_status_get¥"¥n");
33 #endif
34
35 /* get file descriptor */
36         fd = em_get_fd(adr);
37
38 /* read config.tbl */
39         err = em_data_get_int(adr, EM_ID23_ST_STATUS_GET_SLAVE, &slave);
40         if (err < 0) {
41             return EM_ID23_ST_STATUS_GET_ERR_ADR_NOT_SET;
42         }
43
44 #ifndef EM_DEBUG_SIM
45 /* read device status */
46         err = dev_rioA_get_diall(fd, slave, &di_data);
47         if (err < 0) {
48             return EM_ID23_ST_STATUS_GET_ERR;
49         }
50 #else
51         st_status = 0xa1;
52 #endif
53
54 /* line-input-on */
55         if (di_data & EM_ID23_ST_STATUS_ON) {
56             st_status |= EM_ID23_ST_STATUS_ON_BIT;
57         }
58
59 /* fan-stop */
60         if (di_data & EM_ID23_ST_STATUS_FAN) {
61             st_status |= EM_ID23_ST_STATUS_FAN_BIT;
62         }
63
64 /* over-current */
```

```
65          if (di_data & EM_ID23_ST_STATUS_CURR) {
66              st_status |= EM_ID23_ST_STATUS_CURR_BIT;
67          }
68
69  /* over-voltage */
70          if (di_data & EM_ID23_ST_STATUS_VOLT) {
71              st_status |= EM_ID23_ST_STATUS_VOLT_BIT;
72          }
73
74  /* over-temperature  */
75          if (di_data & EM_ID23_ST_STATUS_TEMP) {
76              st_status |= EM_ID23_ST_STATUS_TEMP_BIT;
77          }
78
79  /* remote */
80          if (di_data & EM_ID23_ST_STATUS_REMOTE) {
81              st_status |= EM_ID23_ST_STATUS_REMOTE_BIT;
82          }
83
84  /* failure */
85          if (di_data & EM_ID23_ST_STATUS_FAULT) {
86              st_status |= EM_ID23_ST_STATUS_FAULT_BIT;
87          }
88
89  /* power-on */
90          if (di_data & EM_ID23_ST_STATUS_ON2) {
91              st_status |= EM_ID23_ST_STATUS_ON2_BIT;
92          }
93
94  /* prepare output */
95          em_data_clear(out);
96          em_data_put_int(out, 0, st_status);
97
98          return 0;
99  }
```

```
 1 /*(@header_begin)*************************************************************
 2 *  (C Function)                                                              *
 3 *      Name = em_id23_status_conv_get                                        *
 4 *      Type = int                                                            *
 5 *  (Purpose)                                                                 *
 6 *      convert status to character                                           *
 7 *  (Input)                                                                   *
 8 *      status                                                                *
 9 *  (Output)                                                                  *
10 *      status (Decimal)                                                      *
11 *  (Return)                                                                  *
12 *      0   : ok                                                              *
13 *      < 0 : fail                                                            *
14 *  (Physical unit)                                                           *
15 *      Pa, V, A, mm, sec                                                     *
16 *  (Relation)                                                                *
17 *                                                                            *
18 *  (History)                                                                 *
19 *      Jul-10-1997       F.Funayama /iST  created.                           *
20 *      98-1-11           Hiramatsu                                           *
21 *************************************************************(@header_end)*/
22
23 #include "em_id23.h"
24
25 int em_id23_status_conv_get(Em_data in, Em_data conv, Em_svoc *out)
26 {
27     int err, status;
28
29 #ifdef EM_DEBUG_PRINTF
30     printf("Call func name is ¥"em_id23_status_conv_get¥"¥n");
31 #endif
32
33 /* get status */
34     err = em_data_get_int(in, 0, &status);
35     if (err < 0) {
36         return EM_ID23_STATUS_CONV_GET_ERR ;
37     }
38
39 /* prepare output */
40     em_data_clear(out);
41
42 /* store status in Decimal */
43     sprintf(out->svoc, "%d", status);
44
45     return 0;
46 }
```

```
/*(@header_begin)*************************************************************
*  (C Function)                                                               *
*      Name = em_id23_ttldio_init                                             *
*      Type = int                                                             *
*  (Purpose)                                                                  *
*      TTLDIO initialize                                                      *
*  (Input)                                                                    *
*      none                                                                   *
*  (Output)                                                                   *
*      none                                                                   *
*  (Return)                                                                   *
*      0    : ok                                                              *
*      < 0 : fail                                                             *
*  (Physical unit)                                                            *
*      Pa, V, A, mm, sec                                                      *
*  (Relation)                                                                 *
*                                                                             *
*  (History)                                                                  *
*      Jan- 8-1997      A.Taketani / SPring8      created.                    *
*      99-04-14        Hiramatsu      updated for ID23                        *
*****************************************************************(@header_end)*/

#include "em_id23.h"

int em_id23_ttldio_init()

{
    int mode, err, fd;
    char device_name[EM_MIX_DATA_CHAR_MAX];
    device_name[0] = '¥0';
    mode = EM_VAR_GET_ALL_FIRST;

#ifdef EM_DEBUG_PRINTF
    printf("Call func name is ¥"em_id23_ttldio_init¥"¥n");
#endif

#ifndef EM_DEBUG_SIM
    while (em_save_fd(device_name, mode, &fd ) != EM_SAVE_FD_END) {
        mode = EM_VAR_GET_ALL;
        if (strstr(device_name, "/dev/ttldio_") != NULL) {
            err = dev_ttldio_init(fd);
            if (err != 0) {
                return EM_ID23_TTLDIO_INIT_ERR;
            }
        }
    }
#endif

    return 0;
}
```

```
 1 /*(@header_begin)*************************************************************
 2 *  (C Function)                                                              *
 3 *       Name = em_id23_util_delay_time                                       *
 4 *       Type = int                                                           *
 5 *  (Purpose)                                                                 *
 6 *       delay timer in micro-seconds                                         *
 7 *  (Input)                                                                   *
 8 *       int time_num :                                                       *
 9 *  (Output)                                                                  *
10 *                                                                            *
11 *  (Return)                                                                  *
12 *      0 : ok                                                                *
13 *     <0 : fail                                                              *
14 *  (Physical unit)                                                           *
15 *                                                                            *
16 *  (Relation)                                                                *
17 *                                                                            *
18 *  (History)                                                                 *
19 *      Jul-05-1997      H.Funayama / iST      created.                       *
20 *      98- 1-31          Hiramatsu            revised                        *
21 *      98-10-16          Hiramatsu            updated                        *
22 *************************************************************(@header_end)*/
23 #include <sys/timers.h>
24 #include <signal.h>
25 #include "em_id23.h"
26
27 void my_handler(void);
28
29 int em_id23_util_delay_time(int time_num)
30 {
31    int err = 0;
32    struct itimerspec  tv_new , tv_old;
33    timer_t     tid;
34
35    if(time_num <= 0) {
36    return err;
37    }
38
39 /*** Timer Setting ***/
40    if (signal(SIGALRM, my_handler) == SIG_ERR) {
41    perror("signal");
42    err = -1;
43    return err;
44    }
45
46    tid = mktimer( RTTIMER0, DELIVERY_SIGNALS, (struct itimercb *)0 ) ;
47    if (tid == -1) {
48    printf("Timer get error in em_id23_util_delay_time !!¥n");
49    err = -1;
50    return err;
51    }
52
53    tv_new.it_value.tv_sec  = 0 ;
54    tv_new.it_value.tv_nsec = 1000 * (long)time_num ;
55    tv_new.it_interval.tv_sec  = 0;
56    tv_new.it_interval.tv_nsec = 1000 * (long)time_num ;
57
58    if(reltimer(tid, &tv_new, &tv_old) == -1) {
59    rmtimer(tid);
60    perror("reltimer");
61    err = -1;
62    return err;
63    }
64
```

```
65      pause();
66      rmtimer(tid);
67
68      return 0 ;
69  }
70
71  void my_handler(void)
72  {
73  }
```

```
 1 /*(@header_begin)*************************************************************
 2 *  (C Function)                                                              *
 3 *       Name = em_id23_util_position_cellar                                  *
 4 *       Type = int                                                           *
 5 *  (Purpose)                                                                 *
 6 *       Cellar to simulate gap and phase position                            *
 7 *       (PUT: set_flg = 1, GET: set_flg = 0)                                 *
 8 *  (Input)                                                                   *
 9 *                                                                            *
10 *  (Output)                                                                  *
11 *                                                                            *
12 *  (Return)                                                                  *
13 *       = 0 : ok                                                             *
14 *       < 0 : fail                                                           *
15 *  (Physical unit)                                                           *
16 *       Pa, V, A, mm, sec                                                    *
17 *  (Relation)                                                                *
18 *                                                                            *
19 *  (History)                                                                 *
20 *       99-05-14    Hiramatsu       created                                 *
21 *                                                                            *
22 *************************************************************(@header_end)*/
23
24 #include "em_id23.h"
25
26 int em_id23_util_position_cellar(int command, int k, Em_id23_position *pos)
27 {
28    static Em_id23_position store[3];
29
30 #ifdef EM_DEBUG_PRINTF
31    printf("Call func name is ¥"em_id23_util_position_cellar¥"¥n");
32 #endif
33
34    if (command == EM_ID23_PUT_POSITION) {
35
36        pos->set_flg = 1;           /* PUT position to cellar */
37        memcpy(&(store[k]), pos, sizeof(Em_id23_position));
38
39    } else {
40
41        memcpy(pos, &(store[k]), sizeof(Em_id23_position));
42
43    }
44
45    return 0;
46 }
```

```
/*(@header_begin)*************************************************************
 *  (C Function)                                                             *
 *      Name = em_id23_util_position_conv                                    *
 *      Type = int                                                           *
 *  (Purpose)                                                                *
 *      convert BCD data to Decimal                                          *
 *  (Input)                                                                  *
 *      data_kind (config.tbl)                                               *
 *      BCD data                                                             *
 *  (Output)                                                                 *
 *      position                                                             *
 *  (Return)                                                                 *
 *      0   : ok                                                             *
 *      < 0 : fail                                                           *
 *  (Physical unit)                                                          *
 *      Pa, V, A, mm, sec                                                    *
 *  (Relation)                                                               *
 *                                                                           *
 *  (History)                                                                *
 *      Jul-08-1997      H.Funayama / iST          created.                  *
 *      Aug-25-1997      H.Funayama / iST          Modified.                 *
 *      98-10-16         Hiramatsu                 updated                   *
 *************************************************************(@header_end)*/

#include "em_id23.h"

int em_id23_util_position_conv(int data_kind, const int *bcd_data,
                                                    double *position)
{
    int err;
    int tmp_data[10];          /* store Decimal Number        */
    char tmp_char[15];
    double sign_flg = 1.0;     /* upper-phase and lower-phase */

#ifdef EM_DEBUG_PRINTF
    printf("Call func name is ¥"em_id23_util_position_conv¥"¥n");
#endif

    switch (data_kind) {
    case EM_ID23_GAP_ROT:
         /* third value under point */
         tmp_data[0] = bcd_data[0] & EM_ID23_BCD_1;
         /* second value under point */
         tmp_data[1] = (bcd_data[0] & EM_ID23_BCD_2) >> EM_ID23_R_SHIFT_4;
         /* first value under point */
         tmp_data[2] = (bcd_data[0] & EM_ID23_BCD_3) >> EM_ID23_R_SHIFT_8;
         /* first value over point */
         tmp_data[3] = (bcd_data[0] & EM_ID23_BCD_4) >> EM_ID23_R_SHIFT_12;
         /* second value over point */
         tmp_data[4] = bcd_data[1] & EM_ID23_BCD_1;
         /* third value over point */
         tmp_data[5] = (bcd_data[1] & EM_ID23_BCD_2) >> EM_ID23_R_SHIFT_4;
         sprintf(tmp_char, "%d%d%d.%d%d%d",
                           tmp_data[5], tmp_data[4], tmp_data[3],
                           tmp_data[2], tmp_data[1], tmp_data[0]);
         break;
    case EM_ID23_PHASE_U_ROT:
         /* second value under point */
         tmp_data[0] = bcd_data[2] & EM_ID23_BCD_1;
         /* first value under point */
         tmp_data[1] = (bcd_data[2] & EM_ID23_BCD_2) >> EM_ID23_R_SHIFT_4;
         /* first value over point */
         tmp_data[2] = (bcd_data[2] & EM_ID23_BCD_3) >> EM_ID23_R_SHIFT_8;
         /* second value over point */
```

```
65          tmp_data[3] = (bcd_data[2] & EM_ID23_BCD_4) >> EM_ID23_R_SHIFT_12;
66          sprintf(tmp_char, "%d%d.%d%d",
67                  tmp_data[3], tmp_data[2], tmp_data[1], tmp_data[0]);
68          if (bcd_data[1] & EM_ID23_PHASE_U_SIGN_BIT_ROT)
69              sign_flg = -1.0;
70          break;
71      case EM_ID23_PHASE_L_ROT:
72          /* second value under point */
73          tmp_data[0] = bcd_data[3] & EM_ID23_BCD_1;
74          /* first value under point */
75          tmp_data[1] = (bcd_data[3] & EM_ID23_BCD_2) >> EM_ID23_R_SHIFT_4;
76          /* first value over point */
77          tmp_data[2] = (bcd_data[3] & EM_ID23_BCD_3) >> EM_ID23_R_SHIFT_8;
78          /* second value over point */
79          tmp_data[3] = (bcd_data[3] & EM_ID23_BCD_4) >> EM_ID23_R_SHIFT_12;
80          sprintf(tmp_char, "%d%d.%d%d",
81                  tmp_data[3], tmp_data[2], tmp_data[1], tmp_data[0]);
82          if (bcd_data[1] & EM_ID23_PHASE_L_SIGN_BIT_ROT)
83              sign_flg = -1.0;
84          break;
85      case EM_ID23_GAP_LIN:
86          /* third value under point */
87          tmp_data[0] = bcd_data[1] & EM_ID23_BCD_1;
88          /* second value under point */
89          tmp_data[1] = (bcd_data[1] & EM_ID23_BCD_2) >> EM_ID23_R_SHIFT_4;
90          /* first value under point */
91          tmp_data[2] = bcd_data[0] & EM_ID23_BCD_1;
92          /* first value over point */
93          tmp_data[3] = (bcd_data[0] & EM_ID23_BCD_2) >> EM_ID23_R_SHIFT_4;
94          /* second value over point */
95          tmp_data[4] = bcd_data[3] & EM_ID23_BCD_1;
96          /* third value over point */
97          tmp_data[5] = (bcd_data[3] & EM_ID23_BCD_2) >> EM_ID23_R_SHIFT_4;
98          /* fourth */
99          tmp_data[6] = bcd_data[2] & EM_ID23_BCD_1;
100         sprintf(tmp_char, "%d%d%d%d.%d%d%d", tmp_data[6],
101                                 tmp_data[5], tmp_data[4], tmp_data[3],
102                                 tmp_data[2], tmp_data[1], tmp_data[0]);
103         break;
104     case EM_ID23_PHASE_U_LIN:
105         /* third value under point */
106         tmp_data[0] = bcd_data[5] & EM_ID23_BCD_1;
107         /* second value under point */
108         tmp_data[1] = (bcd_data[5] & EM_ID23_BCD_2) >> EM_ID23_R_SHIFT_4;
109         /* first value under point */
110         tmp_data[2] = bcd_data[4] & EM_ID23_BCD_1;
111         /* first value over point */
112         tmp_data[3] = (bcd_data[4] & EM_ID23_BCD_2) >> EM_ID23_R_SHIFT_4;
113         /* second value over point */
114         tmp_data[4] = bcd_data[7] & EM_ID23_BCD_1;
115         /* third value over point */
116         tmp_data[5] = (bcd_data[7] & EM_ID23_BCD_2) >> EM_ID23_R_SHIFT_4;
117         /* fourth value over point */
118         tmp_data[6] = bcd_data[6] & EM_ID23_BCD_1;
119         sprintf(tmp_char, "%d%d%d%d.%d%d%d", tmp_data[6], tmp_data[5],
120                                 tmp_data[4], tmp_data[3], tmp_data[2],
121                                 tmp_data[1], tmp_data[0]);
122         if (bcd_data[6] & EM_ID23_SIGN_BIT_LIN)
123             sign_flg = -1.0;
124         break;
125     case EM_ID23_PHASE_L_LIN:
126         /* third value under point */
127         tmp_data[0] = bcd_data[9] & EM_ID23_BCD_1;
128         /* second value under point */
```

```
129         tmp_data[1] = (bcd_data[9] & EM_ID23_BCD_2) >> EM_ID23_R_SHIFT_4;
130         /* first value under point */
131         tmp_data[2] = bcd_data[8] & EM_ID23_BCD_1;
132         /* first value over point */
133         tmp_data[3] = (bcd_data[8] & EM_ID23_BCD_2) >> EM_ID23_R_SHIFT_4;
134         /* second value over point */
135         tmp_data[4] = bcd_data[11] & EM_ID23_BCD_1;
136         /* third value over point */
137         tmp_data[5] = (bcd_data[11] & EM_ID23_BCD_2) >> EM_ID23_R_SHIFT_4;
138         /* fourth value over point */
139         tmp_data[6] = bcd_data[10] & EM_ID23_BCD_1;
140         sprintf(tmp_char, "%d%d%d%d.%d%d%d", tmp_data[6], tmp_data[5],
141                                 tmp_data[4], tmp_data[3], tmp_data[2],
142                                 tmp_data[1], tmp_data[0]);
143         if (bcd_data[10] & EM_ID23_SIGN_BIT_LIN)
144             sign_flg = -1.0;
145         break;
146     default:
147         return EM_ID23_UTIL_POSITION_CONV_ERR_UNSUPPORTED_MODE;
148     }
149
150     sscanf(tmp_char, "%lf", position);
151     *position *= sign_flg;
152
153     switch (data_kind) {
154     case EM_ID23_GAP_LIN:
155     case EM_ID23_GAP_ROT:
156        if (*position > EM_ID23_GAP_MAX_NUM) {
157            return EM_ID23_POSITION_CONV_ERR_GAP_MAX_OVER;
158        } else if (*position < EM_ID23_GAP_MIN_NUM) {
159            return EM_ID23_POSITION_CONV_ERR_GAP_MIN_OVER;
160        }
161        break;
162     case EM_ID23_PHASE_U_LIN:
163     case EM_ID23_PHASE_L_LIN:
164     case EM_ID23_PHASE_U_ROT:
165     case EM_ID23_PHASE_L_ROT:
166        if (*position > EM_ID23_PHASE_MAX_NUM) {
167            return EM_ID23_POSITION_CONV_ERR_PHASE_MAX_OVER;
168        } else if (*position < EM_ID23_PHASE_MIN_NUM) {
169            return EM_ID23_POSITION_CONV_ERR_PHASE_MIN_OVER;
170        }
171        break;
172     default:
173        return EM_ID23_UTIL_POSITION_CONV_ERR_UNSUPPORTED_MODE;
174     }
175
176     return 0;
177 }
```

```
/*(@header_begin)*************************************************************
 *  (C Function)                                                             *
 *      Name = em_id23_util_st_curr_cellar                                   *
 *      Type = int                                                           *
 *  (Purpose)                                                                *
 *      transfer current from *curr to cellar,                               *
 *      or transfer current from cellar to *curr                             *
 *  (Input)                                                                  *
 *      command                                                              *
 *  (Output)                                                                 *
 *      set_flg                                                              *
 *  (Return)                                                                 *
 *      0    : ok                                                            *
 *      < 0 : fail                                                           *
 *  (Physical unit)                                                          *
 *      Pa, V, A, mm, sec                                                    *
 *  (Relation)                                                               *
 *                                                                           *
 *  (History)                                                                *
 *      Jul-11-1997      H.Funayama / iST     created.                       *
 *      98- 2-10          Hiramatsu           revised                        *
 *      98-10-16          Hiramatsu           updated                        *
 *      99-03-16          Hiramatsu           deleted  store_flg             *
 *      99-07-05          Hiramatsu           join em_id23_ic_curr_exec      *
 *************************************************************(@header_end)*/

#include "em_id23.h"

int em_id23_util_st_curr_cellar(int command,  int st_num,
                                              Em_id23_current *curr)
{
   static Em_id23_current store_dt[EM_ID23_ST_NUM];

#ifdef EM_DEBUG_PRINTF
   printf("Call func name is ¥"em_id23_util_st_curr_cellar¥"¥n");
#endif

   if (command == EM_ID23_PUT_CURR) {

      curr->set_flg = 1;            /* current was PUT to cellar */
      memcpy(&(store_dt[st_num]), curr, sizeof(Em_id23_current));

   } else {

      memcpy(curr, &(store_dt[st_num]), sizeof(Em_id23_current));

   }

   return 0;
}
```

```
 1 /*(@header_begin)*************************************************************
 2 *  (C Function)                                                              *
 3 *      Name = em_id23_util_table_cellar                                      *
 4 *      Type = int                                                            *
 5 *  (Purpose)                                                                 *
 6 *      transfer current from *table to cellar,                               *
 7 *      or transfer current from cellar to *table                             *
 8 *  (Input)                                                                   *
 9 *                                                                            *
10 *  (Output)                                                                  *
11 *                                                                            *
12 *  (Return)                                                                  *
13 *      0    : ok                                                             *
14 *      < 0 : fail                                                            *
15 *  (Physical unit)                                                           *
16 *      Pa, V, A, mm, sec                                                     *
17 *  (Relation)                                                                *
18 *                                                                            *
19 *  (History)                                                                 *
20 *      Jul-12-1997      H.Funayama / iST    created.                         *
21 *      98- 1-26         Hiramatsu           revised                          *
22 *      98-10-16         Hiramatsu           updated                          *
23 *      99-03-16         Hiramatsu           deleted  store_flg               *
24 *************************************************************(@header_end)*/
25
26 #include "em_id23.h"
27
28 int em_id23_util_table_cellar(int command, Em_id23_table *table)
29 {
30    static Em_id23_table store_dt;
31
32 #ifdef EM_DEBUG_PRINTF
33    printf("Call func name is ¥"em_id23_util_table_cellar¥"¥n");
34 #endif
35
36    if (command == EM_ID23_PUT_TABLE) {
37
38       table->set_flg = 1;          /* table was PUT to cellar */
39       store_dt = *table;
40
41    } else {
42
43       *table = store_dt;
44
45    }
46
47    return 0;
48 }
```

```
 1 /*(@header_begin)*************************************************************
 2 *  (C Function)                                                              *
 3 *      Name = em_id23_util_table_file_get                                     *
 4 *      Type = int                                                             *
 5 *  (Purpose)                                                                 *
 6 *      open the correction table (ASCII file),                                *
 7 *      construct a structure Em_id23_table *table                             *
 8 *  (Input)                                                                   *
 9 *      correction table(bl_id23_steer0.tbl)                                   *
10 *  (Output)                                                                  *
11 *      structure Em_id23_table *table                                         *
12 *  (Return)                                                                  *
13 *      0   : ok                                                               *
14 *      < 0 : fail                                                             *
15 *  (Physical unit)                                                           *
16 *      Pa, V, A, mm, sec                                                      *
17 *  (Relation)                                                                *
18 *                                                                            *
19 *  (History)                                                                 *
20 *      Jul-12-1997     H.Funayama /iST   created.                             *
21 *      98-1-26         Hiramatsu         totally revised                      *
22 *      98-10-16        Hiramatsu         updated                              *
23 *      99-05-09        Hiramatsu         added 4 AC-steering magnets          *
24 *************************************************************(@header_end)*/
25
26 #include "em_id23.h"
27
28 int em_id23_util_table_file_get(char *fname, Em_id23_table *table)
29 {
30    int err, i, j, p_cnt;
31    FILE *fp;
32    char *p, str_tmp[EM_ID23_TABLE_R_MAX_DATA];
33
34 #ifdef EM_DEBUG_PRINTF
35    printf("Call func name is ¥"em_id23_util_table_file_get¥"¥n");
36    printf("Correction table is %s¥n", fname);
37 #endif
38
39    if ((fp = fopen(fname, "r")) == NULL) {
40        return EM_ID23_TABLE_FILE_ERR_OPEN;
41    }
42    do {
43       fgets(str_tmp, sizeof(str_tmp), fp);
44    } while (('#' == str_tmp[0]) || ('<' == str_tmp[0]));
45
46 /* read phase_h */
47    p_cnt = 0;
48    p=strtok(str_tmp,",");
49    table->bx_phase[p_cnt] = atof(p);
50    do {
51       p=strtok(NULL, ",");
52       table->bx_phase[++p_cnt] = atof(p);
53    } while (p != NULL);
54    do {
55       fgets(str_tmp, sizeof(str_tmp), fp);
56    } while (('#' == str_tmp[0]) || ('<' == str_tmp[0]));
57
58 /* read lc_h */
59    j = 0;
60    do {
61       p=strtok(str_tmp,",");
62       table->bx_gap[j] = atof(p);
63       i = 0;
64       do {
```

```
65          p=strtok(NULL, ",");
66          table->bx_table[j][i] = atof(p);
67          i = i + 1 ;
68       } while (p != NULL);
69       j = j + 1 ;
70       fgets(str_tmp, sizeof(str_tmp), fp);
71    } while (('#' == str_tmp[0]) || ('<' != str_tmp[0]));
72    do {
73       fgets(str_tmp, sizeof(str_tmp), fp);
74    } while (('#' == str_tmp[0]) || ('<' == str_tmp[0]));
75
76 /* read phase_v */
77    p_cnt = 0;
78    p=strtok(str_tmp,",");
79    table->by_phase[p_cnt] = atof(p);
80    do {
81       p=strtok(NULL, ",");
82       table->by_phase[++p_cnt] = atof(p);
83    } while (p != NULL);
84    do {
85       fgets(str_tmp, sizeof(str_tmp), fp);
86    } while (('#' == str_tmp[0]) || ('<' == str_tmp[0]));
87
88 /* read lc_v */
89    j = 0;
90    do {
91       p=strtok(str_tmp,",");
92       table->by_gap[j] = atof(p);
93       i = 0;
94       do {
95          p=strtok(NULL, ",");
96          table->by_table[j][i] = atof(p);
97          i = i + 1;
98       } while (p != NULL);
99       j = j + 1;
100      fgets(str_tmp, sizeof(str_tmp), fp);
101   } while (('#' == str_tmp[0]) || ('<' != str_tmp[0]));
102   do {
103      fgets(str_tmp, sizeof(str_tmp), fp);
104   } while (('#' == str_tmp[0]) || ('<' == str_tmp[0]));
105
106 /* read s1phase_h */
107   p_cnt = 0;
108   p=strtok(str_tmp,",");
109   table->s1x_phase[p_cnt] = atof(p);
110 #ifdef EM_DEBUG_PRINTF
111   printf("atof(p) is %f¥n", atof(p));
112 #endif
113   do {
114      p=strtok(NULL, ",");
115      table->s1x_phase[++p_cnt] = atof(p);
116 #ifdef EM_DEBUG_PRINTF
117   printf("atof(p) is %f¥n", atof(p));
118 #endif
119   } while (p != NULL);
120   do {
121      fgets(str_tmp, sizeof(str_tmp), fp);
122   } while (('#' == str_tmp[0]) || ('<' == str_tmp[0]));
123
124 /* read s1_h */
125   j = 0;
126   do {
127      p=strtok(str_tmp,",");
128      table->s1x_gap[j] = atof(p);
```

```
129         i = 0;
130         do {
131            p=strtok(NULL, ",");
132            table->s1x_table[j][i] = atof(p);
133            i = i + 1;
134         } while (p != NULL);
135         j = j + 1;
136         fgets(str_tmp, sizeof(str_tmp), fp);
137      } while (('#' == str_tmp[0]) || ('<' != str_tmp[0]));
138      do {
139         fgets(str_tmp, sizeof(str_tmp), fp);
140      } while (('#' == str_tmp[0]) || ('<' == str_tmp[0]));
141
142 /* read s1phase_v */
143      p_cnt = 0;
144      p=strtok(str_tmp,",");
145      table->s1y_phase[p_cnt] = atof(p);
146 #ifdef EM_DEBUG_PRINTF
147      printf("atof(p) is %f¥n", atof(p));
148 #endif
149      do {
150         p=strtok(NULL, ",");
151         table->s1y_phase[++p_cnt] = atof(p);
152 #ifdef EM_DEBUG_PRINTF
153      printf("atof(p) is %f¥n", atof(p));
154 #endif
155      } while (p != NULL);
156      do {
157         fgets(str_tmp, sizeof(str_tmp), fp);
158      } while (('#' == str_tmp[0]) || ('<' == str_tmp[0]));
159
160 /* read s1_v */
161      j = 0;
162      do {
163         p=strtok(str_tmp,",");
164         table->s1y_gap[j] = atof(p);
165         i = 0;
166         do {
167            p=strtok(NULL, ",");
168            table->s1y_table[j][i] = atof(p);
169            i = i + 1;
170         } while (p != NULL);
171         j = j + 1;
172         fgets(str_tmp, sizeof(str_tmp), fp);
173      } while (('#' == str_tmp[0]) || ('<' != str_tmp[0]));
174      do {
175         fgets(str_tmp, sizeof(str_tmp), fp);
176      } while (('#' == str_tmp[0]) || ('<' == str_tmp[0]));
177
178 /* read s2phase_h */
179      p_cnt = 0;
180      p=strtok(str_tmp,",");
181      table->s2x_phase[p_cnt] = atof(p);
182      do {
183         p=strtok(NULL, ",");
184         table->s2x_phase[++p_cnt] = atof(p);
185      } while (p != NULL);
186      do {
187         fgets(str_tmp, sizeof(str_tmp), fp);
188      } while (('#' == str_tmp[0]) || ('<' == str_tmp[0]));
189
190 /* read s2_h */
191      j = 0;
192      do {
```

```
193         p=strtok(str_tmp,",");
194         table->s2x_gap[j] = atof(p);
195         i = 0;
196         do {
197            p=strtok(NULL, ",");
198            table->s2x_table[j][i] = atof(p);
199            i = i + 1;
200         } while (p != NULL);
201         j = j + 1;
202         fgets(str_tmp, sizeof(str_tmp), fp);
203      } while (('#' == str_tmp[0]) || ('<' != str_tmp[0]));
204      do {
205         fgets(str_tmp, sizeof(str_tmp), fp);
206      } while (('#' == str_tmp[0]) || ('<' == str_tmp[0]));
207
208   /* read s2phase_v */
209      p_cnt = 0;
210      p=strtok(str_tmp,",");
211      table->s2y_phase[p_cnt] = atof(p);
212      do {
213         p=strtok(NULL, ",");
214         table->s2y_phase[++p_cnt] = atof(p);
215      } while (p != NULL);
216      do {
217         fgets(str_tmp, sizeof(str_tmp), fp);
218      } while (('#' == str_tmp[0]) || ('<' == str_tmp[0]));
219
220   /* read s2_v */
221      j = 0;
222      do {
223         p=strtok(str_tmp,",");
224         table->s2y_gap[j] = atof(p);
225         i = 0;
226         do {
227            p=strtok(NULL, ",");
228            table->s2y_table[j][i] = atof(p);
229            i = i + 1;
230         } while(p != NULL);
231         j = j + 1;
232         fgets(str_tmp, sizeof(str_tmp), fp);
233      } while (('#' == str_tmp[0]) || ('<' != str_tmp[0]));
234      do {
235         fgets(str_tmp, sizeof(str_tmp), fp);
236      } while (('#' == str_tmp[0]) || ('<' == str_tmp[0]));
237
238   /* read s3phase_h */
239      p_cnt = 0 ;
240      p=strtok(str_tmp,",");
241      table->s3x_phase[p_cnt] = atof(p);
242      do {
243         p=strtok(NULL, ",");
244         table->s3x_phase[++p_cnt] = atof(p);
245      } while (p != NULL);
246      do {
247         fgets(str_tmp, sizeof(str_tmp), fp);
248      } while (('#' == str_tmp[0]) || ('<' == str_tmp[0]));
249
250   /* read s3_h */
251      j = 0;
252      do {
253         p=strtok(str_tmp,",");
254         table->s3x_gap[j] = atof(p);
255         i = 0;
256         do {
```

```
257           p=strtok(NULL, ",");
258           table->s3x_table[j][i] = atof(p);
259           i = i + 1;
260        } while (p != NULL);
261        j = j + 1;
262        fgets(str_tmp, sizeof(str_tmp), fp);
263     } while (('#' == str_tmp[0]) || ('<' != str_tmp[0]));
264     do {
265        fgets(str_tmp, sizeof(str_tmp), fp);
266     } while (('#' == str_tmp[0]) || ('<' == str_tmp[0]));
267
268  /* read s3phase_v */
269     p_cnt = 0;
270     p=strtok(str_tmp,",");
271     table->s3y_phase[p_cnt] = atof(p);
272     do {
273        p=strtok(NULL, ",");
274        table->s3y_phase[++p_cnt] = atof(p);
275     } while (p != NULL);
276     do {
277        fgets(str_tmp, sizeof(str_tmp), fp);
278     } while (('#' == str_tmp[0]) || ('<' == str_tmp[0]));
279
280  /* read s3_v */
281     j = 0;
282     do {
283        p=strtok(str_tmp,",");
284        table->s3y_gap[j] = atof(p);
285        i = 0;
286        do {
287           p=strtok(NULL, ",");
288           table->s3y_table[j][i] = atof(p);
289           i = i + 1;
290        } while (p != NULL);
291        j = j + 1;
292        fgets(str_tmp, sizeof(str_tmp), fp);
293     } while (('#' == str_tmp[0]) || ('<' != str_tmp[0]));
294     do {
295        fgets(str_tmp, sizeof(str_tmp), fp);
296     } while (('#' == str_tmp[0]) || ('<' == str_tmp[0]));
297
298  /* read s4phase_h */
299     p_cnt = 0;
300     p=strtok(str_tmp,",");
301     table->s4x_phase[p_cnt] = atof(p);
302     do {
303        p=strtok(NULL, ",");
304        table->s4x_phase[++p_cnt] = atof(p);
305     } while (p != NULL);
306     do {
307        fgets(str_tmp, sizeof(str_tmp), fp);
308     } while (('#' == str_tmp[0]) || ('<' == str_tmp[0]));
309
310  /* read s4_h */
311     j = 0;
312     do {
313        p=strtok(str_tmp,",");
314        table->s4x_gap[j] = atof(p);
315        i = 0;
316        do {
317           p=strtok(NULL, ",");
318           table->s4x_table[j][i] = atof(p);
319           i = i + 1;
320        } while (p != NULL);
```

```
321         j = j + 1;
322         fgets(str_tmp, sizeof(str_tmp), fp);
323      } while (('#' == str_tmp[0]) || ('<' != str_tmp[0]));
324      do {
325         fgets(str_tmp, sizeof(str_tmp), fp);
326      } while (('#' == str_tmp[0]) || ('<' == str_tmp[0]));
327
328   /* read s4phase_v */
329      p_cnt = 0;
330      p=strtok(str_tmp,",");
331      table->s4y_phase[p_cnt] = atof(p);
332      do {
333         p=strtok(NULL, ",");
334         table->s4y_phase[++p_cnt] = atof(p);
335      } while (p != NULL);
336      do {
337         fgets(str_tmp, sizeof(str_tmp), fp);
338      } while (('#' == str_tmp[0]) || ('<' == str_tmp[0]));
339
340   /* read s4_v */
341      j = 0;
342      do {
343         p=strtok(str_tmp,",");
344         table->s4y_gap[j] = atof(p);
345         i = 0;
346         do {
347            p=strtok(NULL, ",");
348            table->s4y_table[j][i] = atof(p);
349            i = i + 1;
350         } while (p != NULL);
351         j = j + 1;
352         fgets(str_tmp, sizeof(str_tmp), fp);
353      } while (('#' == str_tmp[0]) || ('<' != str_tmp[0]));
354
355      fclose(fp);
356
357      return 0;
358   }
```

```
/*(@header_begin)*************************************************************
*  (C Function)                                                               *
*      Name = em_id23_vac_conv_get                                            *
*      Type = int                                                             *
*  (Purpose)                                                                  *
*      read vacuum level                                                      *
*  (Input)                                                                    *
*      di_data                                                                *
*  (Output)                                                                   *
*      out->svoc (vacuum level)                                               *
*  (Return)                                                                   *
*      0   : ok                                                               *
*      < 0 : fail                                                             *
*  (Physical unit)                                                            *
*      Pa, V, A, mm, sec                                                      *
*  (Relation)                                                                 *
*                                                                             *
*  (History)                                                                  *
*      Jul-17-1997      F.Funayama /iST  created.                             *
*      98- 1-8          Hiramatsu     revised                                 *
*      98-10-14         Hiramatsu     updated                                 *
*************************************************************(@header_end)*/

#include "em_id23.h"

int em_id23_vac_conv_get(Em_data in, Em_data conv, Em_svoc *out)
{
   int di_data, data_tmp[3];

#ifdef EM_DEBUG_PRINTF
   printf("Call func name is ¥"em_id23_vac_conv_get¥"¥n");
#endif

/* get vacuum level */
   em_data_get_int(in, 0, &di_data);

/* mantissa up 4 bit */
   data_tmp[0] = (di_data >> 4) & 0x000f;

/* mantissa low 4 bit */
   data_tmp[1] = di_data & 0x000f;

/* exponent */
   data_tmp[2] = (di_data >> 8) & 0x000f;

/* prepare output */
   em_data_clear(out);
   sprintf(out->svoc,"%d.%de-%dPa", data_tmp[0], data_tmp[1], data_tmp[2]);

   return 0;
}
```

```
 1 /*(@header_begin)*************************************************************
 2 *  (C Function)                                                              *
 3 *      Name = em_id23_vac_get                                                *
 4 *      Type = int                                                            *
 5 *  (Purpose)                                                                 *
 6 *      read vacuum level                                                     *
 7 *  (Input)                                                                   *
 8 *      parameter (config.tbl)                                                *
 9 *  (Output)                                                                  *
10 *      vacuum level                                                          *
11 *  (Return)                                                                  *
12 *      0   : ok                                                              *
13 *      < 0 : fail                                                            *
14 *  (Physical unit)                                                           *
15 *      Pa, V, A, mm, sec                                                     *
16 *  (Relation)                                                                *
17 *                                                                            *
18 *  (History)                                                                 *
19 *      Jul-17-1997      F.Funayama /iST  created.                            *
20 *      98- 1-11          Hiramatsu     revised                               *
21 *      98-10-14          Hiramatsu     updated                               *
22 *************************************************************(@header_end)*/
23
24 #include "em_id23.h"
25
26 int em_id23_vac_get(Em_data in, Em_data adr, Em_data *out)
27 {
28     int err, fd, loop_count = 1000;
29     int port_num, di_data;
30
31 #ifdef EM_DEBUG_PRINTF
32     printf("Call func name is ¥"em_id23_vac_get¥"¥n");
33 #endif
34
35 /* get file descriptor */
36     fd = em_get_fd(adr);
37
38 /* read config.tbl */
39     err = em_data_get_int(adr, EM_ID23_VAC_GET_PORT, &port_num);
40     if (err < 0) {
41         return EM_ID23_VAC_GET_ERR_ADR_NOT_SET;
42     }
43
44 #ifndef EM_DEBUG_SIM
45 /* read vacuum level */
46     do {
47         err = dev_di_get_diport(fd, port_num, &di_data);
48         if (err < 0) {
49             return EM_ID23_VAC_GET_ERR;
50         }
51         loop_count--;
52     } while ((di_data & 0x1000) != 0 && 0 < loop_count);
53 #else
54     di_data = 0x1967;
55 #endif
56
57 /* prepare output */
58     em_data_clear(out);
59     em_data_put_int(out, 0, di_data);
60
61     return 0;
62 }
```

```
 1 /*(@header_begin)*************************************************************
 2 *  (C Function)                                                              *
 3 *      Name = em_id23_vac_status_get                                         *
 4 *      Type = int                                                            *
 5 *  (Purpose)                                                                 *
 6 *      read device status                                                    *
 7 *  (Input)                                                                   *
 8 *      parameter (config.tbl)                                                *
 9 *  (Output)                                                                  *
10 *      out->data[0].eint : status                                            *
11 *  (Return)                                                                  *
12 *      0   : ok                                                              *
13 *      < 0 : fail                                                            *
14 *  (Physical unit)                                                           *
15 *      Pa, V, A, mm, sec                                                     *
16 *  (Relation)                                                                *
17 *                                                                            *
18 *  (History)                                                                 *
19 *      Jul-11-1997       F.Funayama /iST  created.                           *
20 *      98- 1-11          Hiramatsu    revised                                *
21 *      98-10-14          Hiramatsu    updated                                *
22 *************************************************************(@header_end)*/
23
24 #include "em_id23.h"
25
26 int em_id23_vac_status_get(Em_data in, Em_data adr, Em_data *out)
27 {
28     int fd, dev_num, kind_flg, err, di_data, status = 0 ;
29
30 #ifdef EM_DEBUG_PRINTF
31     printf("Call func name is ¥"em_id23_vac_status_get¥"¥n");
32 #endif
33
34 /* get file descriptor */
35     fd = em_get_fd(adr);
36
37 /* read config.tbl */
38     err = em_data_get_int(adr, EM_ID23_VAC_STATUS_INPUT_PORT, &dev_num);
39     if (err < 0) {
40         return EM_ID23_VAC_STATUS_GET_ERR_ADR_NOT_SET;
41     }
42
43     err = em_data_get_int(adr, EM_ID23_VAC_STATUS_KIND, &kind_flg);
44     if (err < 0) {
45         return EM_ID23_VAC_STATUS_GET_ERR_ADR_NOT_SET;
46     }
47
48 #ifndef EM_DEBUG_SIM
49 /* get status */
50     err = dev_di_get_diport(fd, dev_num, &di_data);
51     if (err < 0) {
52         return EM_ID23_VAC_STATUS_GET_ERR;
53     }
54 #else
55     status = 0x99;
56 #endif
57
58     if (kind_flg == EM_ID23_VAC_STATUS_KIND_1) {
59         status = di_data & 0xff;
60     } else if (kind_flg == EM_ID23_VAC_STATUS_KIND_2) {
61         status = (di_data >> EM_ID23_VAC_STATUS_SHIFT_NUM) & 0xff;
62     } else {
63         return EM_ID23_VAC_STATUS_GET_ERR_KIND;
64     }
```

```
65
66 /* prepare output */
67     em_data_clear(out);
68     em_data_put_int(out, 0, status);
69
70     return 0;
71 }
```

Appendix L: DRAM Log of Periodic Phase Movement (phase-shift = 32.0 mm, hold-time = 200 msec, and linear-scale latch period = 1000 microsec)

	elapsed-time	phase-shift	movement start	trigger-on	excitation
0	73.057449		-32.0		
1	73.072220	-32.028			
2	73.088341				-32.028
3	73.096802	-32.027			
4	73.112518				-32.027
5	73.121056	-31.899			
6	73.136673				-31.899
7	73.145226	-31.151			
8	73.161201				-31.151
9	73.169678	-29.834			
10	73.185371				-29.834
11	73.193855	-28.132			
12	73.209442				-28.132
13	73.217979	-26.054			
14	73.233803				-26.054
15	73.242363	-23.544			
16	73.257980				-23.544
17	73.266510	-20.660			
18	73.282219				-20.660
19	73.290749	-17.397			
20	73.306404				-17.397
21	73.314949	-13.707			
22	73.330658				-13.707
23	73.339119	-9.645			
24	73.354843				-9.645
25	73.363373	-5.149			
26	73.378960				-5.149
27	73.387413	-0.282			
28	73.403152				-0.282
29	73.411606	4.846			
30	73.427238				4.846
31	73.435883	9.519			
32	73.451630				9.519
33	73.460152	13.624			
34	73.475800				13.624

	elapsed-time	phase-shift	movement start	trigger-on	excitation
35	73.484276	17.251			
36	73.499908				17.251
37	73.508469	20.479			
38	73.524239				20.479
39	73.532890	23.322			
40	73.548553				23.322
41	73.557030	25.749			
42	73.572807				25.749
43	73.581284	27.793			
44	73.596947				27.793
45	73.605522	29.486			
46	73.621262				29.486
47	73.629730	30.714			
48	73.645538				30.714
49	73.654015	31.589			
50	73.669632				31.589
51	73.678185	32.085			
52	73.693924				32.085
53	73.702484	32.209			
54	73.718132				32.209
55	73.911026			32.0	
56	74.120865		32.0		
57	74.135078	32.068			
58	74.150902				32.068
59	74.159370	32.066			
60	74.175079				32.066
61	74.183571	31.885			
62	74.199181				31.885
63	74.207718	31.125			
64	74.223457				31.125
65	74.231934	29.873			
66	74.247581				29.873
67	74.256119	28.212			
68	74.271858				28.212
69	74.280380	26.144			
70	74.296043				26.144
71	74.304520	23.656			
72	74.320221				23.656
73	74.328682	20.781			
74	74.344444				20.781

	elapsed-time	phase-shift	movement start	trigger-on	excitation
75	74.352989	17.517			
76	74.368591				17.517
77	74.377060	13.876			
78	74.392822				13.876
79	74.401283	9.821			
80	74.416916				9.821
81	74.425484	5.347			
82	74.441299				5.347
83	74.449760	0.438			
84	74.465469				0.438
85	74.473946	-4.685			
86	74.489555				-4.685
87	74.498093	-9.357			
88	74.513847				-9.357
89	74.522400	-13.491			
90	74.538132				-13.491
91	74.546646	-17.155			
92	74.562393				-17.155
93	74.570923	-20.384			
94	74.586563				-20.384
95	74.595116	-23.239			
96	74.610847				-23.239
97	74.619324	-25.684			
98	74.635132				-25.684
99	74.643616	-27.731			
100	74.659203				-27.731
101	74.667740	-29.399			
102	74.683472				-29.399
103	74.691940	-30.683			
104	74.707573				-30.683
105	74.716110	-31.570			
106	74.731827				-31.570
107	74.740356	-32.073			
108	74.755997				-32.073
109	74.764481	-32.213			
110	74.780182				-32.213
111	74.788658	-32.211			
112	74.804398				-32.211
113	74.812958	-32.194			
114	74.828568				-32.194

	elapsed-time	phase-shift	movement start	trigger-on	excitation
115	74.837051	-32.164			
116	74.852814				-32.164
117	74.861275	-32.143			
118	74.876915				-32.143
119	74.885483	-32.124			
120	74.901199				-32.124
121	74.909660	-32.100			
122	74.925385				-32.100
123	74.961037			-32.0	
124	75.171455		-32.0		
125	75.185715	-32.074			
126	75.201561				-32.074
127	75.210091	-32.075			
128	75.225731				-32.075
129	75.234215	-31.886			
130	75.249832				-31.886
131	75.258369	-31.160			
132	75.274094				-31.160
133	75.282654	-29.913			
134	75.298264				-29.913
135	75.306740	-28.233			
136	75.322449				-28.233
137	75.330971	-26.158			
138	75.346603				-26.158
139	75.355400	-23.664			
140	75.371147				-23.664
141	75.379608	-20.763			
142	75.395317				-20.763
143	75.403786	-17.508			
144	75.419380				-17.508
145	75.427940	-13.847			
146	75.443741				-13.847
147	75.452286	-9.766			
148	75.467934				-9.766
149	75.476456	-5.287			
150	75.492172				-5.287
151	75.500702	-0.408			
152	75.516350				-0.408
153	75.524910	4.751			
154	75.540695				4.751

	elapsed-time	phase-shift	movement start	trigger-on	excitation
155	75.549171	9.422			
156	75.564911				9.422
157	75.573456	13.618			
158	75.589058				13.618
159	75.597534	17.173			
160	75.613281				17.173
161	75.621750	20.403			
162	75.637489				20.403
163	75.646019	23.251			
164	75.661743				23.251
165	75.670288	25.672			
166	75.685936				25.672
167	75.694412	27.723			
168	75.710136				27.723
169	75.718605	29.400			
170	75.734352				29.400
171	75.742897	30.679			
172	75.758530				30.679
173	75.767006	31.566			
174	75.782768				31.566
175	75.791237	32.067			
176	75.806885				32.067
177	75.815453	32.199			
178	75.831268				32.199
179	76.031113			32.0	
180	76.240860		32.0		
181	76.255074	32.063			
182	76.270889				32.063
183	76.279350	32.062			
184	76.295074				32.062
185	76.303574	31.878			
186	76.319183				31.878
187	76.327721	31.119			
188	76.343468				31.119
189	76.351936	29.866			
190	76.367577				29.866
191	76.376106	28.203			
192	76.391853				28.203
193	76.400383	26.143			
194	76.416023				26.143

	elapsed-time	phase-shift	movement start	trigger-on	excitation
195	76.424484	23.661			
196	76.440262				23.661
197	76.448746	20.773			
198	76.464485				20.773
199	76.473030	17.517			
200	76.488663				17.517
201	76.497124	13.861			
202	76.512894				13.861
203	76.521362	9.806			
204	76.537086				9.806
205	76.545654	5.307			
206	76.561394				5.307
207	76.569847	0.419			
208	76.585564				0.419
209	76.594048	-4.721			
210	76.609665				-4.721
211	76.618210	-9.384			
212	76.634048				-9.384
213	76.642601	-13.539			
214	76.658218				-13.539
215	76.666687	-17.167			
216	76.682426				-17.167
217	76.690948	-20.402			
218	76.706566				-20.402
219	76.715111	-23.253			
220	76.730812				-23.253
221	76.739288	-25.683			
222	76.754997				-25.683
223	76.763550	-27.725			
224	76.779137				-27.725
225	76.787613	-29.398			
226	76.803337				-29.398
227	76.811813	-30.683			
228	76.827446				-30.683
229	76.836067	-31.566			
230	76.851799				-31.566
231	76.860329	-32.081			
232	76.875969				-32.081
233	76.884445	-32.211			
234	76.900146				-32.211

	elapsed-time	phase-shift	movement start	trigger-on	excitation
235	77.081032			-32.0	
236	77.291443		-32.0		
237	77.305710	-32.069			
238	77.321548				-32.069
239	77.330078	-32.069			
240	77.345711				-32.069
241	77.354347	-31.887			
242	77.369995				-31.887
243	77.378555	-31.143			
244	77.394295				-31.143
245	77.402840	-29.888			
246	77.418442				-29.888
247	77.426910	-28.212			
248	77.442726				-28.212
249	77.451218	-26.123			
250	77.466866				-26.123
251	77.475441	-23.630			
252	77.491158				-23.630
253	77.499619	-20.756			
254	77.515335				-20.756
255	77.523819	-17.502			
256	77.539528				-17.502
257	77.548065	-13.810			
258	77.563812				-13.810
259	77.572365	-9.741			
260	77.588005				-9.741
261	77.596519	-5.266			
262	77.612251				-5.266
263	77.620781	-0.376			
264	77.636528				-0.376
265	77.645065	4.799			
266	77.660759				4.799
267	77.669220	9.448			
268	77.684952				9.448
269	77.693489	13.563			
270	77.709084				13.563
271	77.717552	17.185			
272	77.733276				17.185
273	77.741753	20.415			
274	77.757401				20.415

	elapsed-time	phase-shift	movement start	trigger-on	excitation
275	77.765945	23.250			
276	77.781685				23.250
277	77.790222	25.684			
278	77.805862				25.684
279	77.814339	27.774			
280	77.829979				27.774
281	77.838615	29.408			
282	77.854378				29.408
283	77.862938	30.686			
284	77.878571				30.686
285	77.887047	31.585			
286	77.902809				31.585
287	77.911285	32.081			
288	77.926926				32.081
289	77.935501	32.214			
290	77.951225				32.214
291	78.151016			32.0	
292	78.360855		32.0		
293	78.375069	32.065			
294	78.390884				32.065
295	78.399353	32.062			
296	78.415062				32.062
297	78.423576	31.881			
298	78.439278				31.881
299	78.447830	31.119			
300	78.463585				31.119
301	78.472130	29.860			
302	78.487785				29.860
303	78.496330	28.195			
304	78.512093				28.195
305	78.520622	26.125			
306	78.536377				26.125
307	78.544922	23.610			
308	78.560661				23.610
309	78.569130	20.729			
310	78.584885				20.729
311	78.593422	17.471			
312	78.609039				17.471
313	78.617516	13.804			
314	78.633354				13.804

	elapsed-time	phase-shift	movement start	trigger-on	excitation
315	78.641815	9.726			
316	78.657448				9.726
317	78.665970	5.242			
318	78.681702				5.242
319	78.690231	0.369			
320	78.705841				0.369
321	78.714310	-4.800			
322	78.729927				-4.800
323	78.738472	-9.418			
324	78.754227				-9.418
325	78.762779	-13.554			
326	78.778397				-13.554
327	78.786865	-17.175			
328	78.802589				-17.175
329	78.811096	-20.415			
330	78.826736				-20.415
331	78.835373	-23.270			
332	78.851089				-23.270
333	78.859550	-25.684			
334	78.875298				-25.684
335	78.883774	-27.735			
336	78.899376				-27.735
337	78.907921	-29.408			
338	78.923645				-29.408
339	78.932182	-30.689			
340	78.947815				-30.689
341	78.956352	-31.580			
342	78.972076				-31.580
343	78.980629	-32.080			
344	78.996246				-32.080
345	79.004784	-32.203			
346	79.020493				-32.203
347	79.028954	-32.200			
348	79.044777				-32.200
349	79.053314	-32.182			
350	79.068893				-32.182
351	79.077362	-32.162			
352	79.093079				-32.162
353	79.101532	-32.135			
354	79.117165				-32.135

Appendix M: Ladder Program for Dynamic Measurement of Gap-distance, Upper-phase and Lower-phase

//
//

ラダープログラム作成ツールM３

ID23 Measurement of Gap and PhasCanon BJC-43Ca

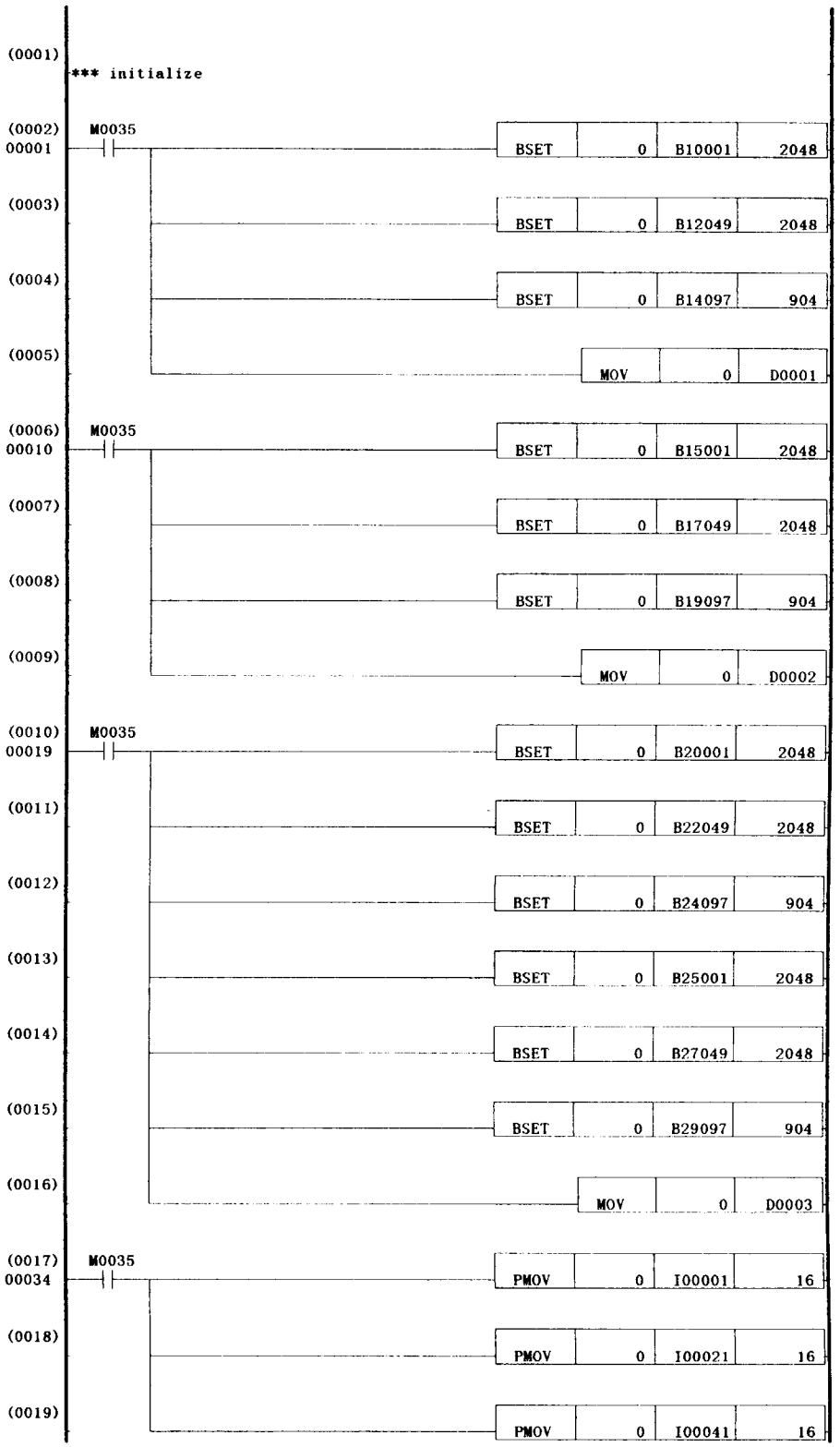
99/09/03 13:21:24

実行プログラム名 : PROGRAM
ｺﾝﾌｨｷﾞｭﾚｰｼｮﾝ名 : SP25
ユーザログ名 :

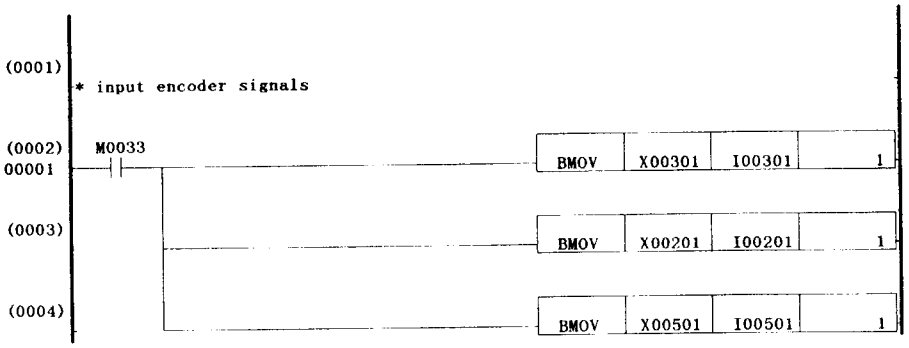
構成ブロック名 : INITIAL XY GET TRIGGER
INPUT01 INPUT02 INPUT03

//
//

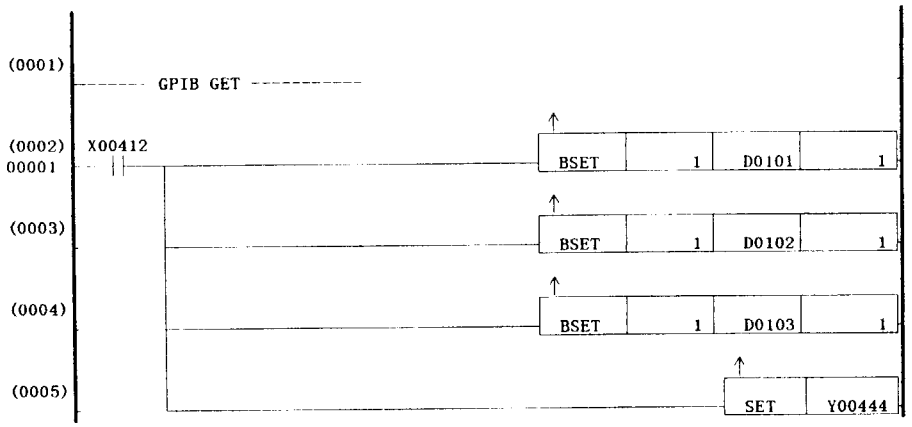
実行プログラム名 : Program　　ブロック名 : INITIAL 日付 : 99/09/03 13:21:24　　ページ : 1



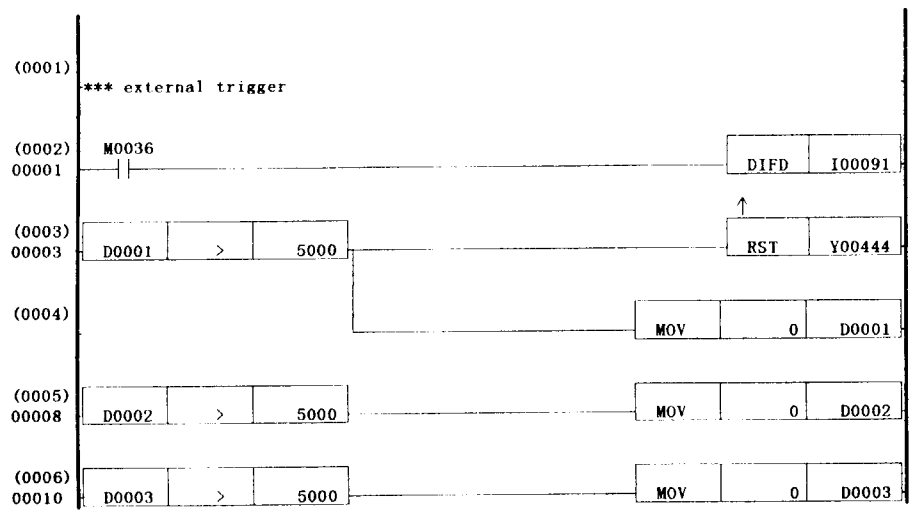
実行プログラム名：Program　　ブロック名：XY　　日付：99/09/03 13:21:24　　ページ：1



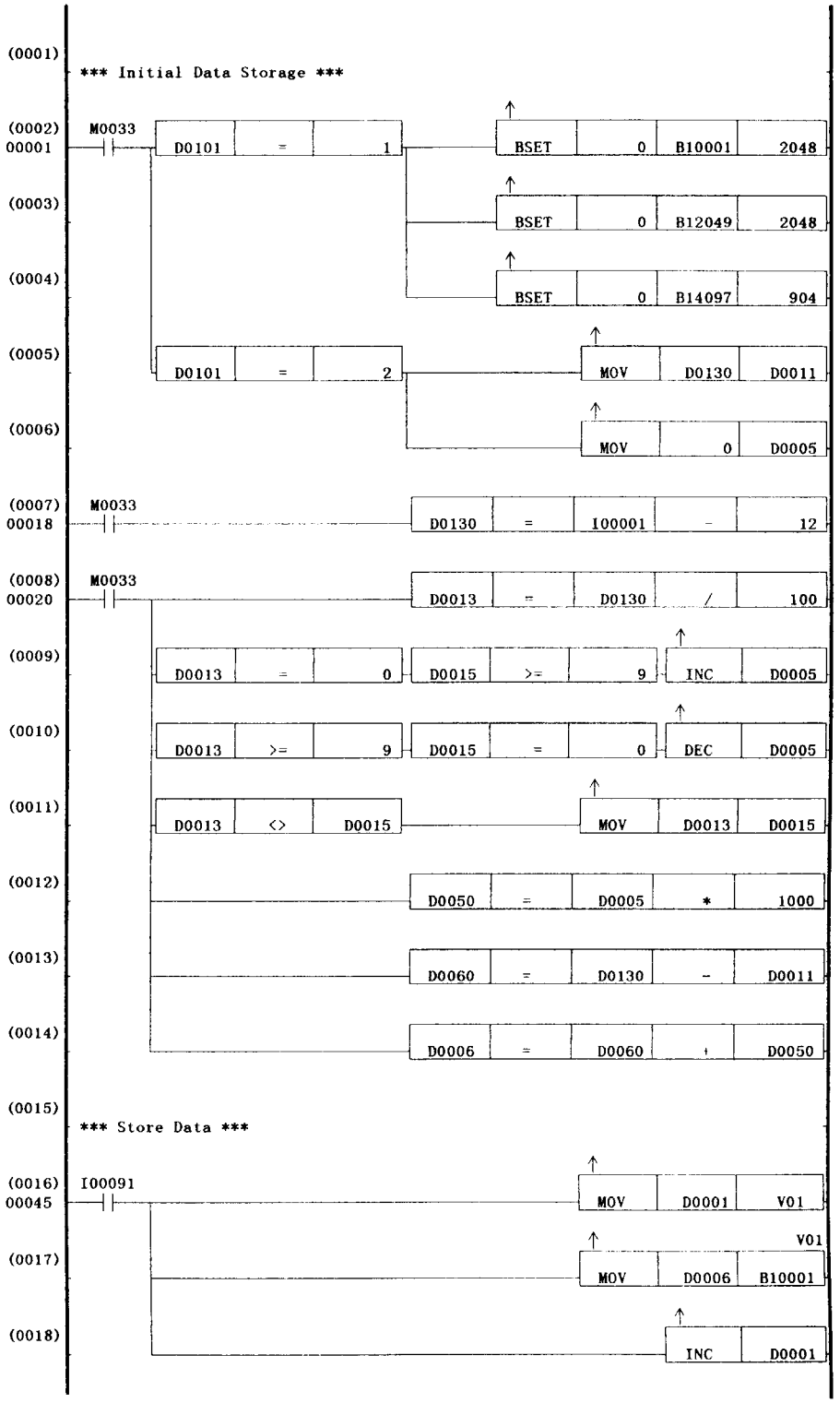
実行プログラム名：Program　　ブロック名：GET　　日付：99/09/03 13:21:24　　ページ：1

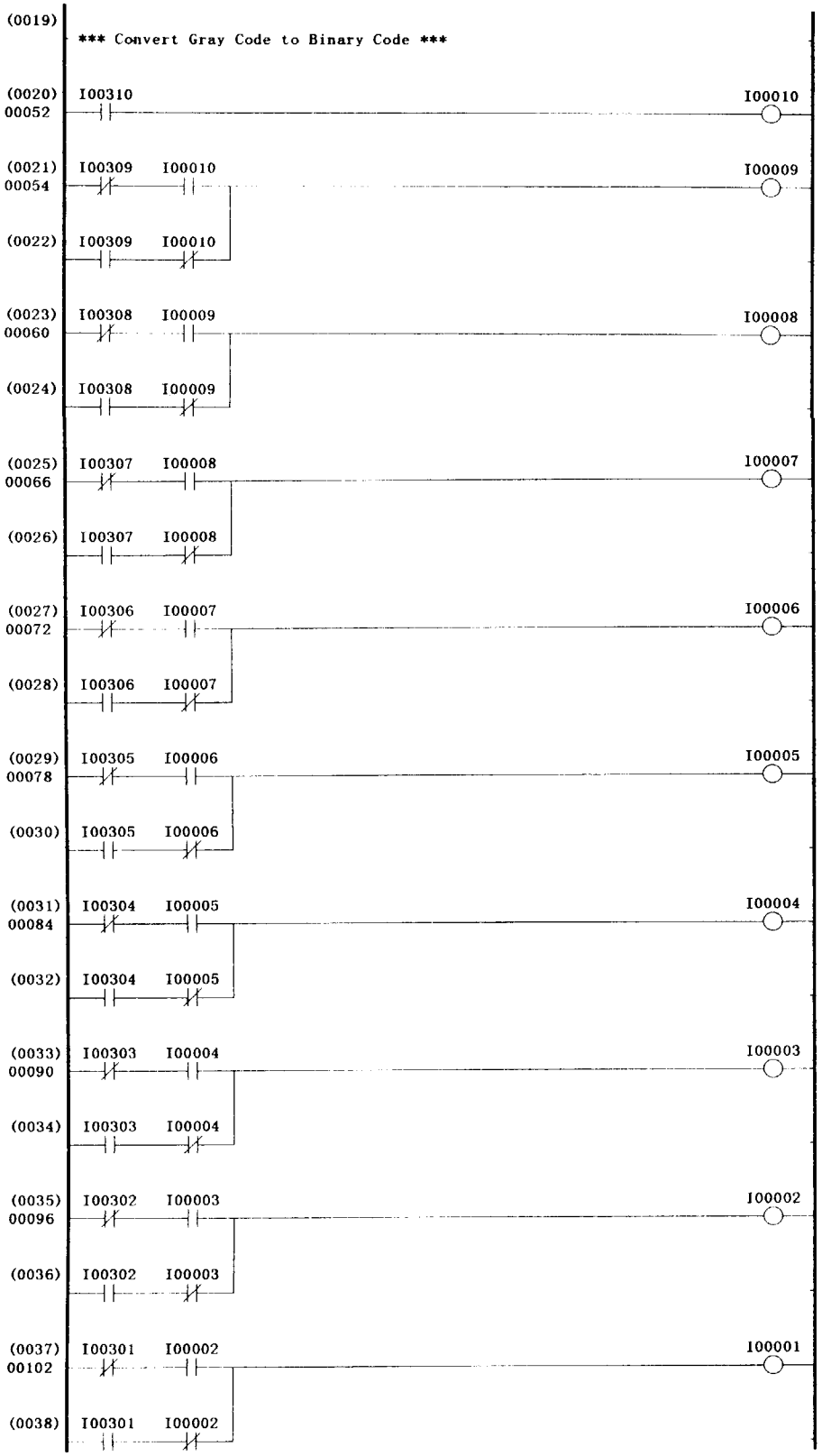


実行プログラム名：Program　　ブロック名：TRIGGER　日付：99/09/03 13:21:24　　ページ：1

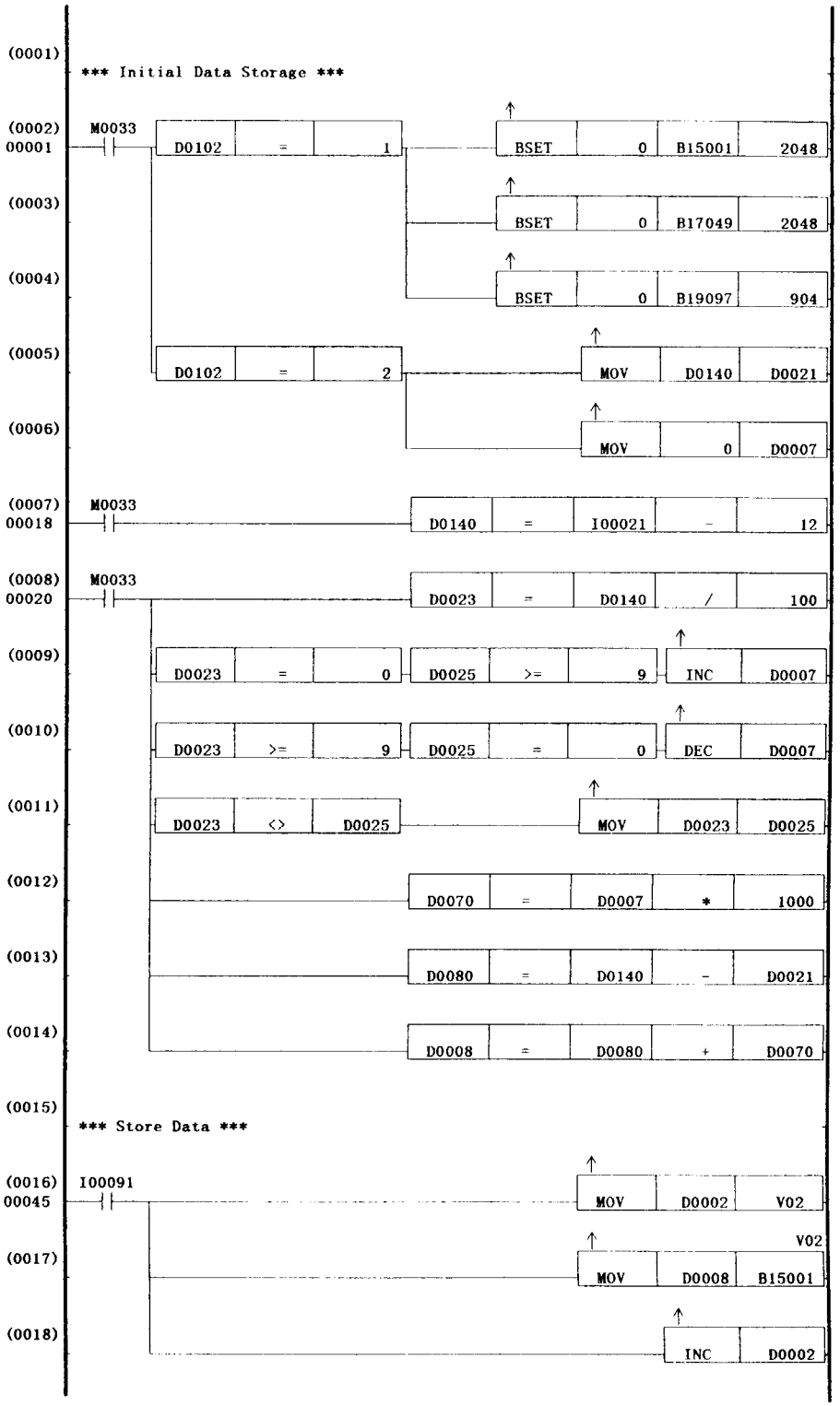


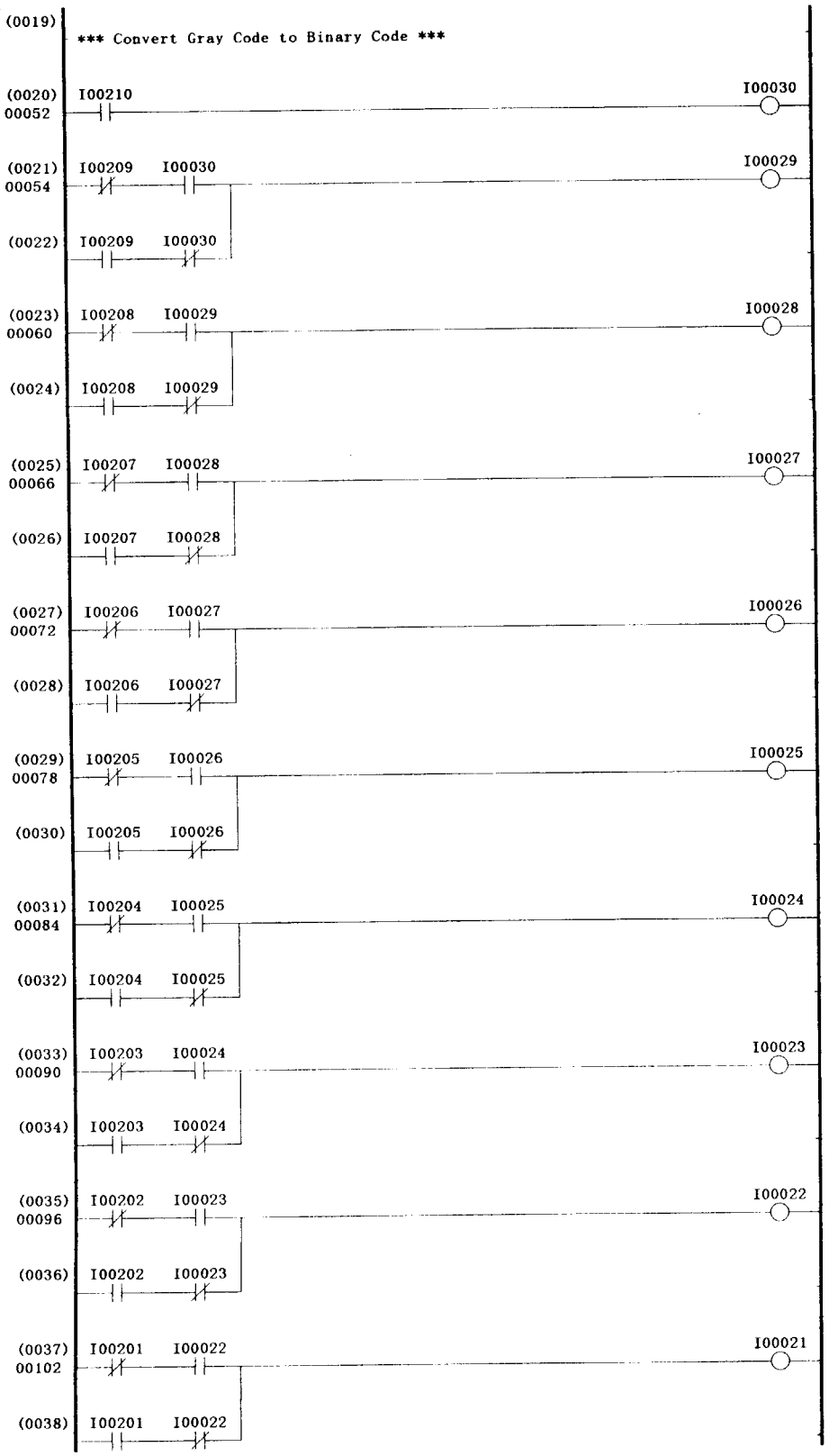
実行プログラム名：Program　　ブロック名：INPUT01 日付：99/09/03 13:21:25　　ページ：1



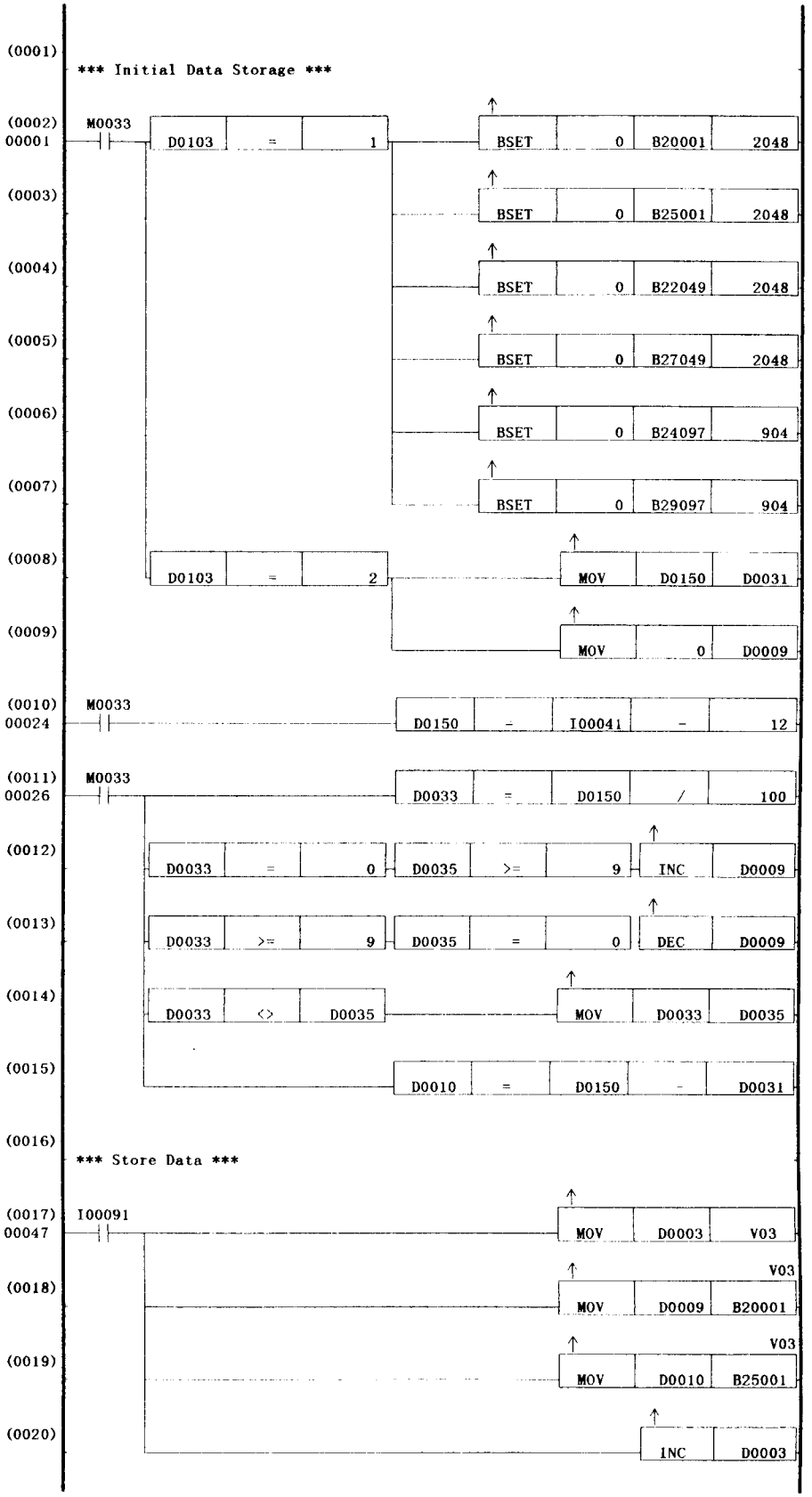

(0019)
*** Convert Gray Code to Binary Code ***
(0020) 00052 I00310 I00010
(0021) 00054 I00309 I00010 I00009
(0022) I00309 I00010
(0023) 00060 I00308 I00009 I00008
(0024) I00308 I00009
(0025) 00066 I00307 I00008 I00007
(0026) I00307 I00008
(0027) 00072 I00306 I00007 I00006
(0028) I00306 I00007
(0029) 00078 I00305 I00006 I00005
(0030) I00305 I00006
(0031) 00084 I00304 I00005 I00004
(0032) I00304 I00005
(0033) 00090 I00303 I00004 I00003
(0034) I00303 I00004
(0035) 00096 I00302 I00003 I00002
(0036) I00302 I00003
(0037) 00102 I00301 I00002 I00001
(0038) I00301 I00002

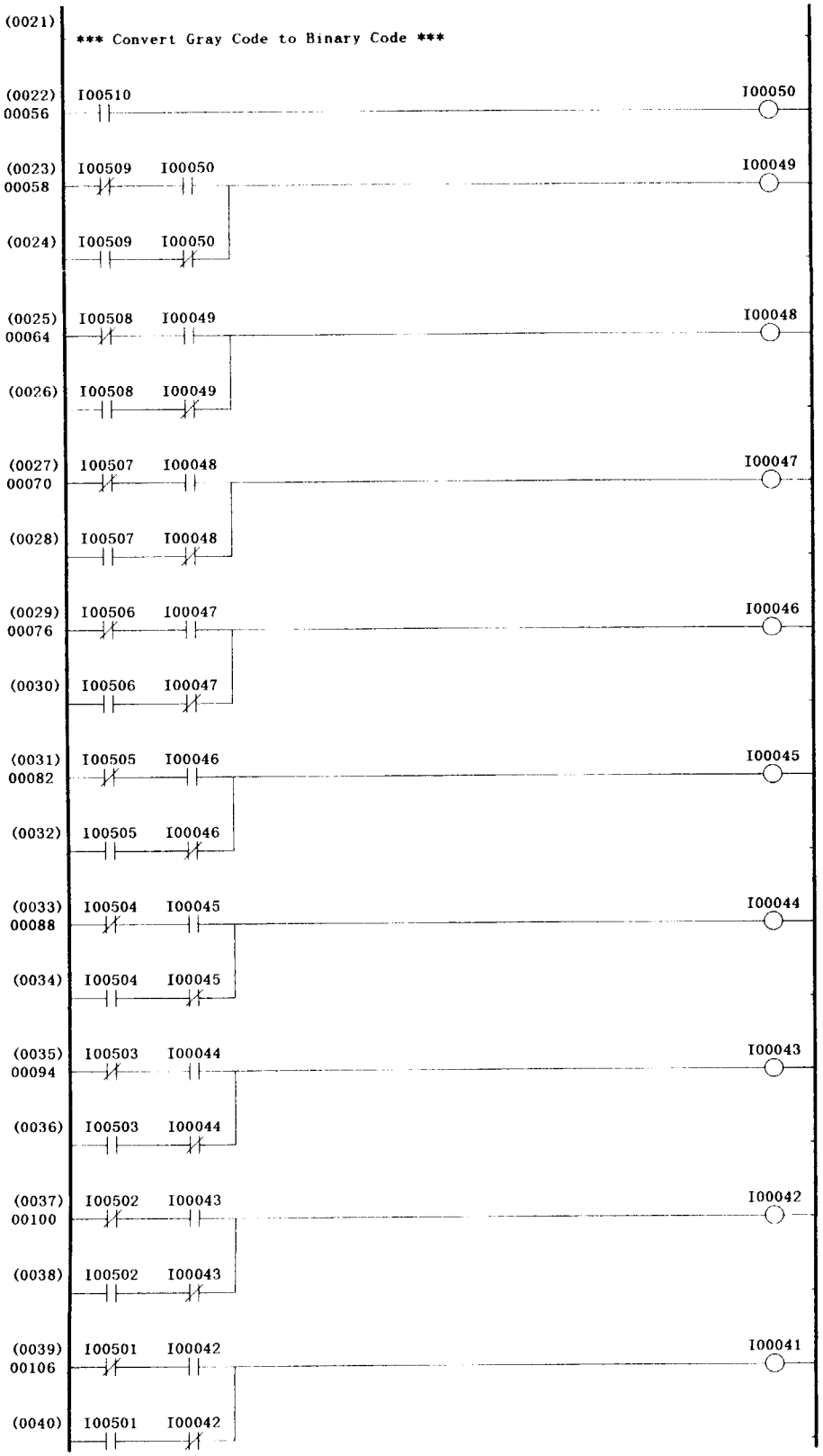
実行プログラム名：Program　　ブロック名：INPUT02 日付：99/09/03 13:21:25　　ページ：1




(0019)
*** Convert Gray Code to Binary Code ***
(0020) 00052 I00210 I00030
(0021) 00054 I00209 I00030 I00029
(0022) I00209 I00030
(0023) 00060 I00208 I00029 I00028
(0024) I00208 I00029
(0025) 00066 I00207 I00028 I00027
(0026) I00207 I00028
(0027) 00072 I00206 I00027 I00026
(0028) I00206 I00027
(0029) 00078 I00205 I00026 I00025
(0030) I00205 I00026
(0031) 00084 I00204 I00025 I00024
(0032) I00204 I00025
(0033) 00090 I00203 I00024 I00023
(0034) I00203 I00024
(0035) 00096 I00202 I00023 I00022
(0036) I00202 I00023
(0037) 00102 I00201 I00022 I00021
(0038) I00201 I00022

実行プログラム名：Program　　ブロック名：INPUT03 日付：99/09/03 13:21:25　　ページ：1




(0021)
*** Convert Gray Code to Binary Code ***
(0022) 00056 I00510 I00050
(0023) 00058 I00509 I00050 I00049
(0024) I00509 I00050
(0025) 00064 I00508 I00049 I00048
(0026) I00508 I00049
(0027) 00070 I00507 I00048 I00047
(0028) I00507 I00048
(0029) 00076 I00506 I00047 I00046
(0030) I00506 I00047
(0031) 00082 I00505 I00046 I00045
(0032) I00505 I00046
(0033) 00088 I00504 I00045 I00044
(0034) I00504 I00045
(0035) 00094 I00503 I00044 I00043
(0036) I00503 I00044
(0037) 00100 I00502 I00043 I00042
(0038) I00502 I00043
(0039) 00106 I00501 I00042 I00041
(0040) I00501 I00042

Appendix N: BASIC Program to convert Gray Code (of Gap-distance, Upper-phase and Lower-phase) output from Rotary-encoder and then, to transfer to PC via GPIB (up to 4960 readings for PLC FAM3 and 1426 readings for Digital Multimeter Keithley-2001)

```
10 'save "genten.bas",a
20 'created  98-11-17  Hiramatsu
30 'updated  98-12-28  Hiramatsu
40 '
50 'purpose: Genten (send number 2 to FAM3)
60 '
70 '
80 CLS : ISET IFC : ISET REN
90 DIM MSG$(16), SND1$(32), SND2$(32), SND3$(32)
100 SND1$ = CHR$(&H2) + "05010WWRD00101,01,0002" + CHR$(&H3)
110 SND2$ = CHR$(&H2) + "05010WWRD00102,01,0002" + CHR$(&H3)
120 SND3$ = CHR$(&H2) + "05010WWRD00103,01,0002" + CHR$(&H3)
130 '
140 INPUT "Hit a key to send number 2 !!", A$
150 PRINT "Genten is OK (low-phase, upp-phase, gap) !!"
160 '
170 PRINT @5; SND1$                    'start readings
180 INPUT @5; MSG$ : PRINT MSG$
190 PRINT @5; SND2$                    'start readings
200 INPUT @5; MSG$ : PRINT MSG$
210 PRINT @5; SND3$                    'start readings
220 INPUT @5; MSG$ : PRINT MSG$
230 '
240 STOP
250 END
.
```

```
10 'save "b:¥test¥encoder¥enco0903.bas",a
20 '99-07-18  Hiramatsu  created  for ID23 tellem test
30 '99-09-03  Hiramatsu  adapted for 4 DMM
40 '
50 '(1)read FAM3 for lower, upper, gap(4960 reading)
60 '(2)read 4 DMM Keithley-2001, 32KB-memory (1300 reading)
70 '(3)trigger source is external(100Hz, TTL, quadratic wave)
80 '(4)latch-box is required for FAM3 and DMM
90 '
100 CLS 3 : CONSOLE , , 0 : SCREEN 3
110 '
120 DIM MSG$(300), SND$(32), SEND$(32)
130 DIM E1(4960), E2(4960), E3(4960), E4(4960), E5(4960)
140 DIM R1(4960), R2(4960), R3(4960), R4(4960)
150 DIM T1(4960), T2(4960), T3(4960), T4(4960)
160 '
170 DEF FNNTOS$( L,M ) = RIGHT$( STR$(L), M )
180 SND$  = CHR$(&H2) + "05010WRDB"     'send STX + FAM3(adr=5)
190 '
200 ISET IFC : ISET REN                 'clear interface, remote-enable
210 '
220 PRINT @11;":syst:pres"              'initialize DMM
230 PRINT @11;"*cls"                    'clear register and error queue
240 PRINT @11;":volt:dc:nplc 0.01"      'dcv meas, integration rate
250 PRINT @11;":volt:dc:rang:auto on"   'auto-range(Volt)
260 PRINT @11;":volt:dc:aver off"       'disable filter
270 PRINT @11;":arm:sour bus"           'wait for bus trigger(GET)
280 PRINT @11;":trig:sour ext"          'wait for external trigger
290 PRINT @11;":trac:egr full"          'full storage mode
300 PRINT @11;":form:elem read,time"    'read data and time-stamp
310 PRINT @11;":trac:clear"             'clear memory
320 PRINT @11;":trac:feed calc"         'put calculated reading to memory
330 PRINT @11;":trac:poin 1426"         'number of reading
340 PRINT @11;":init:cont off;:abor"    'disable continuous init of trigger
350 PRINT @11;":syst:azer:stat off"     'disable auto-zero
360 PRINT @11;":init:cont on"           'enable continuous init of trigger
370 '
380 PRINT @16;":syst:pres"              'initialize DMM
390 PRINT @16;"*cls"                    'clear register and error queue
400 PRINT @16;":volt:dc:nplc 0.01"      'dcv meas, integration rate
410 PRINT @16;":volt:dc:rang:auto on"   'auto-range(Volt)
420 PRINT @16;":volt:dc:aver off"       'disable filter
430 PRINT @16;":arm:sour bus"           'wait for bus trigger(GET)
440 PRINT @16;":trig:sour ext"          'wait for external trigger
450 PRINT @16;":trac:egr full"          'full storage mode
460 PRINT @16;":form:elem read,time"    'read data and time-stamp
470 PRINT @16;":trac:clear"             'clear memory
480 PRINT @16;":trac:feed calc"         'put calculated reading to memory
490 PRINT @16;":trac:poin 1426"         'number of reading
500 PRINT @16;":init:cont off;:abor"    'disable continuous init of trigger
510 PRINT @16;":syst:azer:stat off"     'disable auto-zero
520 PRINT @16;":init:cont on"           'enable continuous init of trigger
530 '
540 PRINT @21;":syst:pres"              'initialize DMM
550 PRINT @21;"*cls"                    'clear register and error queue
```

```
560 PRINT @21;":volt:dc:nplc 0.01"      'dcv meas, integration rate
570 PRINT @21;":volt:dc:rang:auto on"   'auto-range(Volt)
580 PRINT @21;":volt:dc:aver off"       'disable filter
590 PRINT @21;":arm:sour bus"           'wait for bus trigger(GET)
600 PRINT @21;":trig:sour ext"          'wait for external trigger
610 PRINT @21;":trac:egr full"          'full storage mode
620 PRINT @21;":form:elem read,time"    'read data and time-stamp
630 PRINT @21;":trac:clear"             'clear memory
640 PRINT @21;":trac:feed calc"         'put calculated reading to memory
650 PRINT @21;":trac:poin 1426"         'number of reading
660 PRINT @21;":init:cont off;:abor"    'disable continuous init of trigger
670 PRINT @21;":syst:azer:stat off"     'disable auto-zero
680 PRINT @21;":init:cont on"           'enable continuous init of trigger
690 '
700 PRINT @26;":syst:pres"              'initialize DMM
710 PRINT @26;"*cls"                    'clear register and error queue
720 PRINT @26;":volt:dc:nplc 0.01"      'dcv meas, integration rate
730 PRINT @26;":volt:dc:rang:auto on"   'auto-range(Volt)
740 PRINT @26;":volt:dc:aver off"       'disable filter
750 PRINT @26;":arm:sour bus"           'wait for bus trigger(GET)
760 PRINT @26;":trig:sour ext"          'wait for external trigger
770 PRINT @26;":trac:egr full"          'full storage mode
780 PRINT @26;":form:elem read,time"    'read data and time-stamp
790 PRINT @26;":trac:clear"             'clear memory
800 PRINT @26;":trac:feed calc"         'put calculated reading to memory
810 PRINT @26;":trac:poin 1426"         'number of reading
820 PRINT @26;":init:cont off;:abor"    'disable continuous init of trigger
830 PRINT @26;":syst:azer:stat off"     'disable auto-zero
840 PRINT @26;":init:cont on"           'enable continuous init of trigger
850 '
860 ''''' Start measurement '''''
870 '
880 LOCATE 0,1 : PRINT "present time= "; TIME$          'present time
890 LOCATE 0,3 : INPUT "key-in start time HH:MM:SS="; QTIME$
900 IF TIME$ <> QTIME$ THEN GOTO 900 ELSE 910           'wait
910 LOCATE 0,5 : PRINT "started !!!   "; TIME$          'start all meas
920 FIL$ = MID$(TIME$,1,2) + MID$(TIME$,4,2) + MID$(TIME$,7,2)
930 '
940 PRINT @11;":trac:feed:cont next"    'memory control mode
950 WBYTE &H8;                          'send GET to DMM(adr=11)
960 PRINT @16;":trac:feed:cont next"    'memory control mode
970 WBYTE &H8;                          'send GET to DMM(adr=16)
980 PRINT @21;":trac:feed:cont next"    'memory control mode
990 WBYTE &H8;                          'send GET to DMM(adr=21)
1000 PRINT @26;":trac:feed:cont next"   'memory control mode
1010 WBYTE &H8;                         'send GET to DMM(adr=26)
1020 WBYTE &H3F,&H20+5,&H8;             'send GET to FAM3(adr=5)
1030 '
1040 ''''' Wait to get reading '''''
1050 '
1060 S$=TIME$                           'present time
1070 FOR I=1 TO 16                      'wait for reading
1080     WHILE S$=TIME$
1090     LOCATE 0,7: PRINT "reading ...   ";TIME$
1100     WEND
```

```
1110     S$=TIME$
1120 NEXT I
1130 '
1140 PRINT @11;":init:cont off;:abor"   'disable continuous init of trigger
1150 PRINT @16;":init:cont off;:abor"   'disable continuous init of trigger
1160 PRINT @21;":init:cont off;:abor"   'disable continuous init of trigger
1170 PRINT @26;":init:cont off;:abor"   'disable continuous init of trigger
1180 '
1190 ''''' Transfer data to PC '''''
1200 '
1210 PRINT @11;":trac:data?"          'PC requests data to DMM
1220 N = 1 : JP = 1 : WORK$ = ""
1230 WBYTE &H3F, (64+11), (32);       'UNL, talk&H40, listen&H20
1240 D = 0                            'D is 8bit-data
1250 WHILE D <> &HA                   'loop until line-feed
1260     RBYTE; D                     'read 8bit by 8bit
1270     IF D <> &H2C THEN WORK$ = WORK$ + CHR$(D)  'if differ to comma
1280     IF D = &H2C AND JP = 1 THEN GOSUB *DTSTORE  'if equal to comma
1290     R1(N) = RDG
1300     IF D = &H2C AND JP = 2 THEN GOSUB *TMSTORE  'if equal to comma
1310     T1(N-1) = TIM
1320 WEND
1330 IF D = &HA THEN T1(N) = VAL(WORK$)
1340 '
1350 PRINT @11;":syst:pres"             'initialize DMM
1360 '
1370 PRINT @16;":trac:data?"          'PC requests data to DMM
1380 N = 1 : JP = 1 : WORK$ = ""
1390 WBYTE &H3F, (64+16), (32);       'UNL, talk&H40, listen&H20
1400 D = 0                            'D is 8bit-data
1410 WHILE D <> &HA                   'loop until line-feed
1420     RBYTE; D                     'read 8bit by 8bit
1430     IF D <> &H2C THEN WORK$ = WORK$ + CHR$(D)  'if differ to comma
1440     IF D = &H2C AND JP = 1 THEN GOSUB *DTSTORE  'if equal to comma
1450     R2(N) = RDG
1460     IF D = &H2C AND JP = 2 THEN GOSUB *TMSTORE  'if equal to comma
1470     T2(N-1) = TIM
1480 WEND
1490 IF D = &HA THEN T2(N) = VAL(WORK$)
1500 '
1510 PRINT @16;":syst:pres"             'initialize DMM
1520 '
1530 PRINT @21;":trac:data?"          'PC requests data to DMM
1540 N = 1 : JP = 1 : WORK$ = ""
1550 WBYTE &H3F, (64+21), (32);       'UNL, talk&H40, listen&H20
1560 D = 0                            'D is 8bit-data
1570 WHILE D <> &HA                   'loop until line-feed
1580     RBYTE; D                     'read 8bit by 8bit
1590     IF D <> &H2C THEN WORK$ = WORK$ + CHR$(D)  'if differ to comma
1600     IF D = &H2C AND JP = 1 THEN GOSUB *DTSTORE  'if equal to comma
1610     R3(N) = RDG
1620     IF D = &H2C AND JP = 2 THEN GOSUB *TMSTORE  'if equal to comma
1630     T3(N-1) = TIM
1640 WEND
1650 IF D = &HA THEN T3(N) = VAL(WORK$)
```

```
1660 '
1670 PRINT @21;":syst:pres"              'initialize DMM
1680 '
1690 PRINT @26;":trac:data?"         'PC requests data to DMM
1700 N = 1 : JP = 1 : WORK$ = ""
1710 WBYTE &H3F, (64+26), (32);          'UNL, talk&H40, listen&H20
1720 D = 0                               'D is 8bit-data
1730 WHILE D <> &HA                      'loop until line-feed
1740     RBYTE; D                        'read 8bit by 8bit
1750     IF D <> &H2C THEN WORK$ = WORK$ + CHR$(D)  'if differ to comma
1760     IF D = &H2C AND JP = 1 THEN GOSUB *DTSTORE  'if equal to comma
1770     R4(N) = RDG
1780     IF D = &H2C AND JP = 2 THEN GOSUB *TMSTORE  'if equal to comma
1790     T4(N-1) = TIM
1800 WEND
1810 IF D = &HA THEN T4(N) = VAL(WORK$)
1820 '
1830 PRINT @26;":syst:pres"              'initialize DMM
1840 '
1850 J = 1
1860 FOR L=10001 TO 11426 STEP 62          'read lower(z)
1870     SEND$ = SND$ + FNNTOS$(L, 5) + ",62" + CHR$(&H3)
1880     PRINT @5;SEND$
1890     INPUT @5;MSG$
1900     GOSUB *RESPONSE1
1910 NEXT L
1920 '
1930 J = 1
1940 FOR L=15001 TO 16426 STEP 62          'read upper(z)
1950     SEND$ = SND$ + FNNTOS$(L, 5) + ",62" + CHR$(&H3)
1960     PRINT @5;SEND$
1970     INPUT @5;MSG$
1980     GOSUB *RESPONSE2
1990 NEXT L
2000 '
2010 J = 1
2020 FOR L=20001 TO 21426 STEP 62          'read gap(number of rotat)
2030     SEND$ = SND$ + FNNTOS$(L, 5) + ",62" + CHR$(&H3)
2040     PRINT @5;SEND$
2050     INPUT @5;MSG$
2060     GOSUB *RESPONSE3
2070 NEXT L
2080 '
2090 J = 1
2100 FOR L=25001 TO 26426 STEP 62          'read gap(inside 1 rotat)
2110     SEND$ = SND$ + FNNTOS$(L, 5) + ",62" + CHR$(&H3)
2120     PRINT @5;SEND$
2130     INPUT @5;MSG$
2140     GOSUB *RESPONSE4
2150 NEXT L
2160 '
2170 FOR K=1 TO 1426                  '(rotation*pitch)+(inside 1 rotat)
2180     E3(K) = 304.565 + E4(K) + E5(K)    'obtain gap distance
2190 NEXT K
2200 '
```

```
2210 ''''' Store data '''''
2220 '
2230 KEKKA$ = "B:¥TEST¥KEKKA¥" + FIL$ + ".dat"
2240 OPEN KEKKA$ FOR OUTPUT AS #1
2250 WRITE #1, FIL$
2260 '
2270 FOR N=1 TO 1426
2280     PRINT N,
2290     PRINT USING "####.##, ####.##, ####.##, ####.##, ####.##, ####.##, ####.#
#, ####.##, ###.##, ###.##, ####.##";T1(N), T2(N), T3(N), T4(N), R1(N), R2(N), R3(N),
R4(N), E1(N), E2(N), E3(N)
2300     WRITE #1, N, T1(N), T2(N), T3(N), T4(N), R1(N), R2(N), R3(N), R4(N), E1(N), E2
(N), E3(N)
2310     PSET(N * 639 / 1364, 200 - 5 * R1(N)), 1      'long-x(marine-blue)
2320     PSET(N * 639 / 1364, 200 - 5 * R2(N)), 3      'long-y(violet)
2330     PSET(N * 639 / 1364, 200 - 5 * R3(N)), 5      'Bx(light-blue)
2340     PSET(N * 639 / 1364, 200 - 5 * R4(N)), 7      'By(white)
2350     PSET(N * 639 / 1364, 200), 7                  'zero(white)
2360     PSET(N * 639 / 1364, 300 + 3 * E1(N)), 2      'lower(red)
2370     PSET(N * 639 / 1364, 110 + 3 * E2(N)), 4      'upper(green)
2380     PSET(N * 639 / 1364, 255 - .85 * E3(N)), 5    'gap(light-blue)
2390 NEXT N
2400 '
2410 K$=INPUT$(1)
2420 CLOSE #1 : PRINT "normal end; file closed !!"
2430 STOP
2440 '
2450 *DTSTORE
2460     RDG = VAL(WORK$)
2470     D = 0 : JP = 2 : WORK$ = ""
2480 RETURN
2490 '
2500 *TMSTORE
2510     TIM = VAL(WORK$)
2520     D = 0 : JP = 1 : N = N + 1 : WORK$ = ""
2530 RETURN
2540 '
2550 *RESPONSE1
2560 FOR I=1 TO  248  STEP 4              'read 62 data(=248 Byte)
2570     A$ = MID$(MSG$ , I + 7 , 4 )     'sum 7 Byte to skip STX05010K
2580     B = VAL("&H" + A$)               'convert Hexa to Decimal
2590     E1(J) = - 62.11 + B * (.01)      'lower position(z)
2600     J = J + 1 : NEXT I
2610 RETURN
2620 '
2630 *RESPONSE2
2640 FOR I=1 TO  248  STEP 4              'read 62 data(=248 Byte)
2650     A$ = MID$(MSG$ , I + 7 , 4 )     'sum 7 Byte to skip STX05010K
2660     B = VAL("&H" + A$)               'convert Hexa to Decimal
2670     E2(J) = - 59.56 - B * (.01)      'upper position(z)
2680     J = J + 1 : NEXT I
2690 RETURN
2700 '
2710 *RESPONSE3
2720 FOR I=1 TO  248  STEP 4              'read 62 data(=248 Byte)
```

```
2730      A$ = MID$(MSG$ , I + 7 , 4 )    'sum 7 Byte to skip STX05010K
2740      B = VAL("&H" + A$)              'convert Hexa to Decimal
2750      E4(J) = B * (.1)                'gap(rotation * pitch)
2760      J = J + 1 : NEXT I
2770 RETURN
2780 '
2790 *RESPONSE4
2800 FOR I=1 TO  248  STEP 4              'read 62 data(=248 Byte)
2810      A$ = MID$(MSG$ , I + 7 , 4 )    'sum 7 Byte to skip STX05010K
2820      B = VAL("&H" + A$)              'convert Hexa to Decimal
2830      E5(J) = B * (.0001)             'gap(inside 1 rotation)
2840      J = J + 1 : NEXT I
2850 RETURN
2860 '
2870 END
```

Appendix O: Test sheet for EM and EMA (tellem*)

<Instructions for the test>

(a) test place : Storage Ring Maintenance Corridor, C-23 Area.
(b) host name : id23rt (HP9000/743rt, HP-RT version 2.21)
(c) login name : blcntl
(d) test directory:
/prj/bin/id23rt/test/tellem* (EM)
/test/ema_rt* (EMA)
/test/config.tbl (Device configuration table)
/test/bl_id23_dummy.tbl (This is dummy table used to check the reading of "correction table")

(e) home directory:
/home/bl/blcntl/bl_id23_steer0.tbl
(This is actual correction table for 10 steering magnets)

Test sheet for ID23-tellem

テスト日　：　　-　　-

テスト環境：

実際の制御(id23rt)　　：/HP-RT.srfs01/prj/bin/id23rt/test/tellem*

シミュレータ(bisyamon)：/hprt/users/yoichi/tellem_sim*

資　料：添付資料（A）ステータス戻り値表

添付資料（B）補正電磁石の励磁テーブル(ダミー値)

(注意： ギャップと位相の “put/.../initial”と “put/.../preset” 及び VME－PLC－機器間 の配線チェックは別に試験する)

SVOC command	Reply	盤面表示
%get/bl_id23_gap_rot/position	＿＿＿＿ mm	＿＿＿＿ mm
%get/bl_id23_phase_upper_rot/position	＿＿＿＿ mm	＿＿＿＿ mm
%get/bl_id23_phase_lower_rot/position	＿＿＿＿ mm	＿＿＿＿ mm
%get/bl_id23_gap/position	＿＿＿＿ mm	＿＿＿＿ mm
%get/bl_id23_phase/position	＿＿＿＿ mm	
%get/bl_id23_phase_upper/position	＿＿＿＿ mm	＿＿＿＿ mm
%get/bl_id23_phase_lower/position	＿＿＿＿ mm	＿＿＿＿ mm
%get/bl_id23_gap/status	＿＿＿＿ (添付資料A)	
%get/bl_id23_phase/status	＿＿＿＿ (添付資料A)	
%get/bl_id23_phase_trigger_r/status	＿＿＿＿ (添付資料A)	
%get/bl_id23_phase_trigger_l/status	＿＿＿＿ (添付資料A)	
%get/bl_id23_rfbpm/status	＿＿＿＿ (添付資料A)	
%get/bl_id23_ivg_1/status	＿＿＿＿ (添付資料A)	
%get/bl_id23_ivg_2/status	＿＿＿＿ (添付資料A)	
%get/bl_id23_sip/status	＿＿＿＿ (添付資料A)	
%get/bl_id23_lmt_gap/status	＿＿＿＿ (添付資料A)	

%get/bl_id23_lmt_phase/status	(添付資料A)
%get/bl_id23_lmt_emergency/status	(添付資料A)
%get/bl_id23_rfbpm_intlk/status	(添付資料A)
%get/bl_id23_lc_h/status	(添付資料A)
%get/bl_id23_lc_v/status	(添付資料A)
%get/bl_id23_st_h_1/status	(添付資料A)
%get/bl_id23_st_h_2/status	(添付資料A)
%get/bl_id23_st_h_3/status	(添付資料A)
%get/bl_id23_st_h_4/status	(添付資料A)
%get/bl_id23_st_v_1/status	(添付資料A)
%get/bl_id23_st_v_2/status	(添付資料A)
%get/bl_id23_st_v_3/status	(添付資料A)
%get/bl_id23_st_v_4/status	(添付資料A)

		盤面表示
%get/bl_id23_lc_h/current	A	A
%get/bl_id23_st_h_1/current	A	A
%get/bl_id23_st_h_2/current	A	A
%get/bl_id23_st_h_3/current	A	A
%get/bl_id23_st_h_4/current	A	A
%get/bl_id23_lc_v/current	A	A
%get/bl_id23_st_v_1/current	A	A
%get/bl_id23_st_v_2/current	A	A
%get/bl_id23_st_v_3/current	A	A
%get/bl_id23_st_v_4/current	A	A

（5ページのコマンドを実行してから、ここに戻ること）

		盤面表示
%get/bl_id23_lc_dac_h/current	A	A
%get/bl_id23_st_dac_h_1/current	A	A
%get/bl_id23_st_dac_h_2/current	A	A
%get/bl_id23_st_dac_h_3/current	A	A
%get/bl_id23_st_dac_h_4/current	A	A
%get/bl_id23_lc_dac_v/current	A	A
%get/bl_id23_st_dac_v_1/current	A	A

%get/bl_id23_st_dac_v_2/current	A	A
%get/bl_id23_st_dac_v_3/current	A	A
%get/bl_id23_st_dac_v_4/current	A	A

入力模擬電圧

%get/bl_id23_rfbpm_1_x/voltage	V	V
%get/bl_id23_rfbpm_1_y/voltage	V	V
%get/bl_id23_rfbpm_2_x/voltage	V	V
%get/bl_id23_rfbpm_2_y/voltage	V	V
%get/bl_id23_rfbpm_3_x/voltage	V	V
%get/bl_id23_rfbpm_3_y/voltage	V	V
%get/bl_id23_rfbpm_4_x/voltage	V	V
%get/bl_id23_rfbpm_4_y/voltage	V	V

%get/bl_id23_rfbpm_1_x/position	mm	V
%get/bl_id23_rfbpm_1_y/position	mm	V
%get/bl_id23_rfbpm_2_x/position	mm	V
%get/bl_id23_rfbpm_2_y/position	mm	V
%get/bl_id23_rfbpm_3_x/position	mm	V
%get/bl_id23_rfbpm_3_y/position	mm	V
%get/bl_id23_rfbpm_4_x/position	mm	V
%get/bl_id23_rfbpm_4_y/position	mm	V

盤面表示

%get/bl_id23_ivg_1/pressure	Pa	Pa
%get/bl_id23_ivg_2/pressure	Pa	Pa

%put/bl_id23_lc_h/on	盤面で確認する
%put/bl_id23_lc_h/off	盤面で確認する
%put/bl_id23_st_h_1/on	盤面で確認する
%put/bl_id23_st_h_1/off	盤面で確認する
%put/bl_id23_st_h_2/on	盤面で確認する
%put/bl_id23_st_h_2/off	盤面で確認する
%put/bl_id23_st_h_3/on	盤面で確認する
%put/bl_id23_st_h_3/off	盤面で確認する

%put/bl_id23_st_h_4/on	盤面で確認する
%put/bl_id23_st_h_4/off	盤面で確認する
%put/bl_id23_lc_v/on	盤面で確認する
%put/bl_id23_lc_v/off	盤面で確認する
%put/bl_id23_st_v_1/on	盤面で確認する
%put/bl_id23_st_v_1/off	盤面で確認する
%put/bl_id23_st_v_2/on	盤面で確認する
%put/bl_id23_st_v_2/off	盤面で確認する
%put/bl_id23_st_v_3/on	盤面で確認する
%put/bl_id23_st_v_3/off	盤面で確認する
%put/bl_id23_st_v_4/on	盤面で確認する
%put/bl_id23_st_v_4/off	盤面で確認する

%put/bl_id23_lc_table/dummy.tbl _____ (添付資料B)

以下の４行では、gap=300.0mm, phase=0.0mm で発行し、盤面表示で動かないことを確認すること。

%put/bl_id23_gap/34.0mm	(for check)	_____
%put/bl_id23_phase/124.0mm	(for check)	_____
%put/bl_id23_phase_upper/62.0mm	(for check)	_____
%put/bl_id23_phase_lower/-62.0mm	(for check)	_____

		盤面表示
%put/bl_id23_gap/300.0mm	_____ mm	_____ mm
%put/bl_id23_phase/0.0mm	_____ mm	_____ mm
%put/bl_id23_phase_upper/10.0mm	_____ mm	_____ mm
%put/bl_id23_phase_lower/-10.0mm	_____ mm	_____ mm

%put/bl_id23_phase_brake/on	_____	(VME-LED で確認)
%put/bl_id23_phase_brake/off	_____	(VME-LED で確認)
%put/bl_id23_phase_trigger_r/on	_____	(VME-LED で確認)
%put/bl_id23_phase_trigger_l/off	_____	(VME-LED で確認)
%put/bl_id23_phase_trigger_r/on	_____	(VME-LED で確認)
%put/bl_id23_phase_trigger_l/off	_____	(VME-LED で確認)

```
%set/bl_id23_lc_h/15.0A      (for check)  _____
%set/bl_id23_lc_h/-15.0A     (for check)  _____
%set/bl_id23_st_h_1/15.0A    (for check)  _____
%set/bl_id23_st_h_1/-15.0A   (for check)  _____
%set/bl_id23_st_h_2/15.0A    (for check)  _____
%set/bl_id23_st_h_2/-15.0A   (for check)  _____
%set/bl_id23_st_h_3/15.0A    (for check)  _____
%set/bl_id23_st_h_3/-15.0A   (for check)  _____
%set/bl_id23_st_h_4/15.0A    (for check)  _____
%set/bl_id23_st_h_4/-15.0A   (for check)  _____
%set/bl_id23_lc_v/15.0A      (for check)  _____
%set/bl_id23_lc_v/-15.0A     (for check)  _____
%set/bl_id23_st_v_1/15.0A    (for check)  _____
%set/bl_id23_st_v_1/-15.0A   (for check)  _____
%set/bl_id23_st_v_2/15.0A    (for check)  _____
%set/bl_id23_st_v_2/-15.0A   (for check)  _____
%set/bl_id23_st_v_3/15.0A    (for check)  _____
%set/bl_id23_st_v_3/-15.0A   (for check)  _____
%set/bl_id23_st_v_4/15.0A    (for check)  _____
%set/bl_id23_st_v_4/-15.0A   (for check)  _____
```

（2ページから）

```
%set/bl_id23_lc_h/1.0A                    _____
%set/bl_id23_st_h_1/2.0A                  _____
%set/bl_id23_st_h_2/3.0A                  _____
%set/bl_id23_st_h_3/4.0A                  _____
%set/bl_id23_st_h_4/5.0A                  _____
%set/bl_id23_lc_v/-1.0A                   _____
%set/bl_id23_st_v_1/-2.0A                 _____
%set/bl_id23_st_v_2/-3.0A                 _____
%set/bl_id23_st_v_3/-4.0A                 _____
%set/bl_id23_st_v_4/-5.0A                 _____
%put/bl_id23_st/exec                      _____
```

（2ページへ戻る）

以下を実行するに先立ち、収納部内で温度スイッチの中継端子台を一旦はずし、failure を出しておくこと。

%put/bl_id23_lc_h/reset	_____	(盤面表示で reset 確認)
%put/bl_id23_st_h_1/reset	_____	(盤面表示で reset 確認)
%put/bl_id23_st_h_2/reset	_____	(盤面表示で reset 確認)
%put/bl_id23_st_h_3/reset	_____	(盤面表示で reset 確認)
%put/bl_id23_st_h_4/reset	_____	(盤面表示で reset 確認)
%put/bl_id23_lc_v/reset	_____	(盤面表示で reset 確認)
%put/bl_id23_st_v_1/reset	_____	(盤面表示で reset 確認)
%put/bl_id23_st_v_2/reset	_____	(盤面表示で reset 確認)
%put/bl_id23_st_v_3/reset	_____	(盤面表示で reset 確認)
%put/bl_id23_st_v_4/reset	_____	(盤面表示で reset 確認)

%put/bl_id23_phase/reset のテスト

emergency を押す
emergency を戻す
local で位相を駆動する (動かないはず)
remote にする
%put/bl_id23_phase/reset _____
local で位相を駆動する (動くはず)

%put/bl_id23_gap/200.0mm (before start EMA) _____

********** EMA emitted from EM **********

%get/bl_id23_pattern_ema/status _____
%put/bl_id23_pattern_ema/create (1st) _____
%get/bl_id23_pattern_ema/status _____
%put/bl_id23_pattern_ema/destroy _____
%get/bl_id23_pattern_ema/status _____
%put/bl_id23_pattern_ema/create (2nd) _____
%put/bl_id23_pattern_ema/destroy _____
%put/bl_id23_pattern_ema/create (3rd) _____
%put/bl_id23_pattern_ema/destroy _____
%put/bl_id23_pattern_ema/create (4th) _____
%put/bl_id23_pattern_ema/destroy _____
%put/bl_id23_pattern_ema/create (5th) _____
%put/bl_id23_pattern_ema/destroy _____
%put/bl_id23_pattern_ema/create (6th) _____
%put/bl_id23_pattern_ema/destroy _____
%put/bl_id23_pattern_ema/create (7th) _____
%put/bl_id23_pattern_ema/destroy _____
%put/bl_id23_pattern_ema/create (8th) _____
%put/bl_id23_pattern_ema/destroy _____
%put/bl_id23_pattern_ema/create (9th) _____
%put/bl_id23_pattern_ema/destroy _____
%put/bl_id23_pattern_ema/create (10th) _____
%get/bl_id23_pattern_ema/status _____

%put/bl_id23_pattern_ema/phase_r_60.0mm _____
%put/bl_id23_pattern_ema/phase_l_-60.0mm _____
%put/bl_id23_pattern_ema/hold_r_200msec _____
%put/bl_id23_pattern_ema/hold_l_200msec _____
%get/bl_id23_pattern_ema/status _____
%get/bl_id23_pattern_ema/phase_r _____ mm
%get/bl_id23_pattern_ema/phase_l _____ mm
%get/bl_id23_pattern_ema/hold_r _____ msec
%get/bl_id23_pattern_ema/hold_l _____ msec
%put/bl_id23_pattern_ema/start _____
%get/bl_id23_pattern_ema/status _____

%put/bl_id23_pattern_ema/stop _____

%get/bl_id23_pattern_ema/status _____

%put/bl_id23_pattern_ema/phase_r_0.0mm _____

%put/bl_id23_pattern_ema/phase_l_123.0mm _____

%get/bl_id23_pattern_ema/phase_r _____ mm

%get/bl_id23_pattern_ema/phase_l _____ mm

%put/bl_id23_pattern_ema/start _____

%put/bl_id23_pattern_ema/hold_r_0msec _____

%put/bl_id23_pattern_ema/hold_l_100msec _____

%get/bl_id23_pattern_ema/hold_r _____ msec

%get/bl_id23_pattern_ema/hold_l _____ msec

%put/bl_id23_pattern_ema/start _____

%put/bl_id23_pattern_ema/destroy _____

********** EMA emitted from GUI **********

%get/bl_id23_pattern/status _____

%put/bl_id23_pattern/create (1st) _____

%get/bl_id23_pattern/status _____

%put/bl_id23_pattern/destroy _____

%get/bl_id23_pattern/status _____

%put/bl_id23_pattern/create (2nd) _____

%put/bl_id23_pattern/destroy _____

%put/bl_id23_pattern/create (3rd) _____

%put/bl_id23_pattern/destroy _____

%put/bl_id23_pattern/create (4th) _____

%put/bl_id23_pattern/destroy _____

%put/bl_id23_pattern/create (5th) _____

%put/bl_id23_pattern/destroy _____

%put/bl_id23_pattern/create (6th) _____

%put/bl_id23_pattern/destroy _____

%put/bl_id23_pattern/create (7th) _____

%put/bl_id23_pattern/destroy _____

%put/bl_id23_pattern/create (8th) _____

%put/bl_id23_pattern/destroy _____

%put/bl_id23_pattern/create (9th) _____

%put/bl_id23_pattern/destroy _____

%put/bl_id23_pattern/create (10th) _____

%get/bl_id23_pattern/status _____

%put/bl_id23_pattern_phase/32.0mm_-32.0mm _____

%put/bl_id23_pattern_hold/800msec_1200msec _____

%get/bl_id23_pattern/status _____

%get/bl_id23_pattern_phase/position _____mm _____mm

%get/bl_id23_pattern_hold/time _____msec _____msec

%put/bl_id23_pattern/start _____

%get/bl_id23_pattern/status _____

%put/bl_id23_pattern/stop _____

%get/bl_id23_pattern/status _____

%put/bl_id23_pattern_phase/0.0mm_-123.0mm _____

%get/bl_id23_pattern_phase/position _____mm _____mm

%put/bl_id23_pattern/start _____

%put/bl_id23_pattern_hold/0msec_100msec _____

%get/bl_id23_pattern_hold/time _____msec_____msec

%put/bl_id23_pattern/start _____

%put/bl_id23_pattern/destroy _____

添付資料（A）

get/bl_id23_gap/status の状況

AVME-9421(di_1)　　24V

bit no.	Port	Channel	0	1
1	A	04	normal	over-heat

VME-LED 点灯状況	Reply（10 進）

get/bl_id23_phase/status の状況

AVME-9421(di_1)　　24V

bit no.	Port	Channel	0	1
1	A	05	normal	up-amp-alarm
2	A	06	up-servo-no-ready	normal
3	A	07	up-pos-no-complete	normal
4	A	08	normal	up-warning
5	A	09	normal	up-brake-on
6	A	10	normal	low-amp-alarm
7	A	11	low-servo-no-ready	normal
8	A	12	low-pos-no-complete	normal
9	A	13	normal	low-warning
10	A	14	normal	low-brake-on

	Reply(2 進)										Reply(10 進)	状　況
	10	9	8	7	6	5	4	3	2	1		
上段												all ブレーカ on
下段												
上段												all ブレーカ off
下段												
上段												上ブレーカのみ on
下段												
上段												下ブレーカのみ on
下段												
上段												上ブレーキ on
下段												下ブレーキ off
上段												上ブレーキ off
下段												下ブレーキ off
上段												上ブレーキ off
下段												下ブレーキ on

上段：get の reply

下段：VME-LED 点灯状況

get/bl_id23_lc_h/status の状況

RIO TypeA (rio_0)

bit no.	Pin	Channel	0	1
1	18	c	line-off	line-on
2	19	c	normal	fan-stop
3	20	a	normal	over-current
4	20	c	normal	over-voltage
5	21	a	normal	over-temperature
6	19	a	local	remote
7	21	c	normal	failure
8	18	a	power-off	power-on

	Reply(2 進)	Reply(10 進)	状　況
	8 7 6 5 4 3 2 1		
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			

上段：get の reply

下段：VME-LED 点灯状況

get/bl_id23_lc_v/status の状況

RIO TypeA (rio_0)

bit no.	Pin	Channel	0	1
1	18	c	line-off	line-on
2	19	c	normal	fan-stop
3	20	a	normal	over-current
4	20	c	normal	over-voltage
5	21	a	normal	over-temperature
6	19	a	local	remote
7	21	c	normal	failure
8	18	a	power-off	power-on

	Reply(2 進)	Reply(10 進)	状 況
	8 7 6 5 4 3 2 1		
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			

上段：get の reply

下段：VME-LED 点灯状況

get/bl_id23_st_h_1/status の状況

RIO TypeA (rio_0)

bit no.	Pin	Channel	0	1
1	18	c	line-off	line-on
2	19	c	normal	fan-stop
3	20	a	normal	over-current
4	20	c	normal	over-voltage
5	21	a	normal	over-temperature
6	19	a	local	remote
7	21	c	normal	failure
8	18	a	power-off	power-on

	Reply(2 進)	Reply(10 進)	状 況
	8 7 6 5 4 3 2 1		
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			

上段：get の reply

下段：VME-LED 点灯状況

get/bl_id23_st_v_1/status の状況

RIO TypeA (rio_0)

bit no.	Pin	Channel	0	1
1	18	c	line-off	line-on
2	19	c	normal	fan-stop
3	20	a	normal	over-current
4	20	c	normal	over-voltage
5	21	a	normal	over-temperature
6	19	a	local	remote
7	21	c	normal	failure
8	18	a	power-off	power-on

	Reply(2 進)	Reply(10 進)	状　況
	8 7 6 5 4 3 2 1		
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			

上段：get の reply

下段：VME-LED 点灯状況

get/bl_id23_st_h_2/status の状況

RIO TypeA (rio_0)

bit no.	Pin	Channel	0	1
1	18	c	line-off	line-on
2	19	c	normal	fan-stop
3	20	a	normal	over-current
4	20	c	normal	over-voltage
5	21	a	normal	over-temperature
6	19	a	local	remote
7	21	c	normal	failure
8	18	a	power-off	power-on

	Reply(2 進)	Reply(10 進)	状　況
	8 7 6 5 4 3 2 1		
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			

上段：get の reply

下段：VME-LED 点灯状況

get/bl_id23_st_v_2/status の状況

RIO TypeA (rio_0)

bit no.	Pin	Channel	0	1
1	18	c	line-off	line-on
2	19	c	normal	fan-stop
3	20	a	normal	over-current
4	20	c	normal	over-voltage
5	21	a	normal	over-temperature
6	19	a	local	remote
7	21	c	normal	failure
8	18	a	power-off	power-on

	Reply(2 進)	Reply(10 進)	状 況
	8 7 6 5 4 3 2 1		
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			

上段：get の reply

下段：VME-LED 点灯状況

get/bl_id23_st_h_3/status の状況

RIO TypeA (rio_0)

bit no.	Pin	Channel	0	1
1	18	c	line-off	line-on
2	19	c	normal	fan-stop
3	20	a	normal	over-current
4	20	c	normal	over-voltage
5	21	a	normal	over-temperature
6	19	a	local	remote
7	21	c	normal	failure
8	18	a	power-off	power-on

	Reply(2 進)	Reply(10 進)	状 況
	8 7 6 5 4 3 2 1		
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			

上段：get の reply

下段：VME-LED 点灯状況

get/bl_id23_st_v_3/status の状況

RIO TypeA (rio_0)

bit no.	Pin	Channel	0	1
1	18	c	line-off	line-on
2	19	c	normal	fan-stop
3	20	a	normal	over-current
4	20	c	normal	over-voltage
5	21	a	normal	over-temperature
6	19	a	local	remote
7	21	c	normal	failure
8	18	a	power-off	power-on

	Reply(2 進)	Reply(10 進)	状　況
	8 7 6 5 4 3 2 1		
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			

上段：get の reply

下段：VME-LED 点灯状況

get/bl_id23_st_h_4/status の状況

RIO TypeA (rio_0)

bit no.	Pin	Channel	0	1
1	18	c	line-off	line-on
2	19	c	normal	fan-stop
3	20	a	normal	over-current
4	20	c	normal	over-voltage
5	21	a	normal	over-temperature
6	19	a	local	remote
7	21	c	normal	failure
8	18	a	power-off	power-on

	Reply(2 進)	Reply(10 進)	状 況
	8 7 6 5 4 3 2 1		
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			

上段：get の reply

下段：VME-LED 点灯状況

get/bl_id23_st_v_4/status の状況

RIO TypeA (rio_0)

bit no.	Pin	Channel	0	1
1	18	c	line-off	line-on
2	19	c	normal	fan-stop
3	20	a	normal	over-current
4	20	c	normal	over-voltage
5	21	a	normal	over-temperature
6	19	a	local	remote
7	21	c	normal	failure
8	18	a	power-off	power-on

	Reply(2 進)	Reply(10 進)	状 況
	8 7 6 5 4 3 2 1		
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			

上段：get の reply

下段：VME-LED 点灯状況

get/bl_id23_rfbpm/status の状況

HIMV-630/TIO-96 (ttldio_0)　　5V

bit no.	Port	Channel	0	1
1	2	00	normal (pll-lock)	mon 1 fail (pll-unlock)
2	2	01	normal (pll-lock)	mon 2 fail (pll-unlock)
3	2	02	normal (pll-lock)	mon 3 fail (pll-unlock)
4	2	03	normal (pll-lock)	mon 4 fail (pll-unlock)

	Reply(2 進)	Reply(10 進)	状　況(Bergoz module PLL-lock 点灯状況)
	4 3 2 1		
上段			
下段			
上段			
下段			
上段			
下段			
上段			
下段			

上段：get の reply

下段：VME-LED 点灯状況

get/bl_id23_phase_trigger_r/status の状況

HIMV-630/TIO-96 (ttldio_0)　　5V

bit no.	Port	Channel	0	1
1	0	08	trigger-off	trigger-on

SVOC command	Reply （2進）	VME-LED 点灯状況
put/bl_id23_phase_trigger_r/on		
put/bl_id23_phase_trigger_r/off		
put/bl_id23_phase_trigger_r/on		

get/bl_id23_phase_trigger_l/status の状況

HIMV-630/TIO-96 (ttldio_0)　　5V

bit no.	Port	Channel	0	1
1	0	09	trigger-off	trigger-on

SVOC command	Reply （2進）	VME-LED 点灯状況
put/bl_id23_phase_trigger_l/on		
put/bl_id23_phase_trigger_l/off		
put/bl_id23_phase_trigger_l/on		

get/bl_id23_ivg_1/status の状況

AVME-9421(di_2)　　24V

bit no.	Port	Channel	0	1
1	C	00	filament-off	filament-on
2	C	01	degas-off	degas-on
3	C	02	local	remote
4	C	03	set-point 1 off	set-point 1 on
5	C	04	set-point 2 off	set-point 2 on
6	C	05	gauge 1 on	gauge 2 on
7	C	06	normal	alarm
8	C	07	power-off	power-on

	Reply(2 進)	Reply(10 進)	コントローラの LED 点灯状況
	8 7 6 5 4 3 2 1		
上段			filament-on
下段			
上段			filament-off
下段			
上段			power-off
下段			
上段			power-on
下段			
上段			filament-on
下段			

上段：get の reply

下段：VME-LED 点灯状況

get/bl_id23_ivg_2/status の状況

AVME-9421(di_2)　　24V

bit no.	Port	Channel	0	1
1	C	08	filament-off	filament-on
2	C	09	degas-off	degas-on
3	C	10	local	remote
4	C	11	set-point 1 off	set-point 1 on
5	C	12	set-point 2 off	set-point 2 on
6	C	13	gauge 1 on	gauge 2 on
7	C	14	normal	alarm
8	C	15	power-off	power-on

	Reply(2 進)	Reply(10 進)	コントローラの LED 点灯状況
	8 7 6 5 4 3 2 1		
上段			filament-on
下段			
上段			filament-off
下段			
上段			power-off
下段			
上段			power-on
下段			
上段			filament-on
下段			

上段：get の reply

下段：VME-LED 点灯状況

get/bl_id23_sip/status の状況

AVME-9421(di_2)　　24V

bit no.	Port	Channel	0	1
1	D	00	set point 1 off	set point 1 on
2	D	01	set point 2 off	set point 1 on
3	D	02	local	remote
4	D	03	normal	failure
5	D	04	normal	over-load
6	D	05	HV-off	HV-on

	Reply(2 進)	Reply(10 進)	状　況
	6 5 4 3 2 1		
上段			HV-on
下段			
上段			HV-off
下段			
上段			HV-on
下段			

上段：get の reply

下段：VME-LED 点灯状況

get/bl_id23_lmt_gap/status の状況

AVME-9421(di_1)　　24V

bit no.	Port	Channel	0	1
1	B	00	up-upstr-open-on(LS1)	normal
2	B	04	up-upstr-close-on(LS5)	normal
3	B	01	up-dostr-open-on(LS2)	normal
4	B	05	up-dostr-close-on(LS6)	normal
5	B	02	lo-upstr-open-on(LS3)	normal
6	B	06	lo-upstr-close-on(LS7)	normal
7	B	03	lo-dostr-open-on(LS4)	normal
8	B	07	lo-dostr-close-on(LS8)	normal
9	C	06	normal	gap-reference-on(LS23)

	Reply(2 進)	Reply(10 進)	状　況
	9 8 7 6 5 4 3 2 1		
上段			LS1 のみ abnormal
下段			
上段			LS5 のみ abnormal
下段			
上段			LS2 のみ abnormal
下段			
上段			LS6 のみ abnormal
下段			
上段			LS3 のみ abnormal
下段			
上段			LS7 のみ abnormal
下段			
上段			LS4 のみ abnormal
下段			
上段			LS8 のみ abnormal
下段			
上段			LS23 のみ abnormal
下段			
上段			all LS normal
下段			

上段：get の reply

下段：VME-LED 点灯状況

get/bl_id23_lmt_phase/status の状況

AVME-9421(di_1)　　24V

bit no.	Port	Channel	0	1
1	B	08	row1-dowst-on(LS9)	normal
2	B	09	row1-upstr-on(LS10)	normal
3	B	10	row2-dowstr-on(LS11)	normal
4	B	11	row2-upstr-on(LS12)	normal
5	B	12	row3-dowstr-on(LS13)	normal
6	B	13	row3-upstr-on(LS14)	normal
7	B	14	row4-dowstr-on(LS15)	normal
8	B	15	row4-upstr-on(LS16)	normal
9	C	07	normal	up-reference-on(LS24)
10	C	08	normal	lo-reference-on(LS25)

	Reply(2 進)	Reply(10 進)	状　況
	10 9 8 7 6 5 4 3 2 1		
上段			LS9 のみ abnormal
下段			
上段			LS10 のみ abnormal
下段			
上段			LS11 のみ abnormal
下段			
上段			LS12 のみ abnormal
下段			
上段			LS13 のみ abnormal
下段			
上段			LS14 のみ abnormal
下段			
上段			LS15 のみ abnormal
下段			
上段			LS16 のみ abnormal
下段			
上段			LS24 のみ abnormal
下段			
上段			LS25 のみ abnormal
下段			
上段			all LS normal
下段			

上段：get の reply

下段：VME-LED 点灯状況

get/bl_id23_lmt_emergency/status の状況

AVME-9421(di_1)　　24V

bit no.	Port	Channel	0	1
1	C	00	CS-upper-on(LS17)	normal
2	C	01	CS-lower-on(LS18)	normal
3	A	00	emergency-panel-on	normal
4	A	01	emergency-ID-on	normal
5	A	02	emergency-hall-on	normal
6	A	03	remote	local

	Reply(2 進)	Reply(10 進)	状　況
	6 5 4 3 2 1		
上段			LS17 のみ abnormal
下段			
上段			LS18 のみ abnormal
下段			
上段			emergency-panel のみ abnormal
下段			
上段			emergency-ID のみ abnormal
下段			
上段			emergency-hall のみ abnormal
下段			
上段			all LS normal
下段			
上段			remote
下段			
上段			local
下段			

上段：get の reply

下段：VME-LED 点灯状況

get/bl_id23_rfbpm_intlk/status の状況

HIMV-602A(dio602a_0)

bit no.	Port	Channel	0	1
1		00	beam-shift 2x+ on	
2		01	beam-shift 2x- on	
3		02	beam-shift 2y+ on	
4		03	beam-shift 2y- on	
5		04	beam-shift 4x+ on	
6		05	beam-shift 4x- on	
7		06	beam-shift 4y+ on	
8		07	beam-shift 4y- on	
9		08	GFO	not-GFO
10		09	normal	RC < 1 mA
11		10	bpm-enable	bpm-disable
12		11	normal	beam-shift-on
13		12	beam-abort-required	normal

"Long-steering" and "AC-steering magnets"の get status 発行手順

以下の各手順（１－１６）を実行後、 get/bl_id23_lc/status と
get/bl_id23_st/status を発行すること。

手順	操 作	方 法
1	ブレーカ・オフ	盤面操作
2	ブレーカ・オン、ローカル	盤面操作
3	オペレート・オン	盤面操作
4	ext-int AB	収納部（中継端子台で open/close）
5	ext-int N	収納部（中継端子台で open/close）
6	盤面リセット	盤面操作
7	オペレート・オン	盤面操作
8	オペレート・オフ	盤面操作
9	リモート	盤面操作
10	オペレート・オン	WS(tellem)
11	ext-int AB	収納部（中継端子台で open/close）
12	ext-int N	収納部（中継端子台で open/close）
13	tellem で リセット	WS(tellem)
14	オペレート・オン	WS(tellem)
15	オペレート・オフ	WS(tellem)
16	ブレーカ・オフ	盤面操作

添付資料（B）　補正電磁石の励磁テーブル（ダミー値）

```
# Correction Table  99-08-04    Hiramatsu   created
#                               to put a constant current (dummy data)
#
# phase-shift(D, unit: mm) for long-steering magnet Bx
<phase_h>
-121., 0. ,121.
</phase_h>
#
# excitation current (unit: A)
<Ic_h>
35. , -1.0, -1.0, -1.0
300., -1.0, -1.0, -1.0
350., -1.0, -1.0, -1.0
</Ic_h>
#
# phase-shift(D, unit: mm) for long-steering magnet By
<phase_v>
-121., 0. ,121.
</phase_v>
#
# excitation current (unit: A)
<Ic_v>
35. , 1.0, 1.0, 1.0
300., 1.0, 1.0, 1.0
350., 1.0, 1.0, 1.0
</Ic_v>
#
# phase-shift(D, unit: mm) for AC-steering magnet S1x
<s1phase_h>
-121., 0. ,121.
</s1phase_h>
#
# excitation current (unit: A)
<s1_h>
35. , -2.0, -2.0, -2.0
300., -2.0, -2.0, -2.0
350., -2.0, -2.0, -2.0
</s1_h>
#
# phase-shift(D, unit: mm) for AC-steering magnet S1y
<s1phase_v>
-121., 0. ,121.
</s1phase_v>
#
# excitation current (unit: A)
<s1_v>
35. , 2.0, 2.0, 2.0
300., 2.0, 2.0, 2.0
350., 2.0, 2.0, 2.0
</s1_v>
#
# phase-shift(D, unit: mm) for AC-steering magnet S2x
<s2phase_h>
-121., 0. ,121.
</s2phase_h>
#
# excitation current (unit: A)
<s2_h>
35. , -3.0, -3.0, -3.0
300., -3.0, -3.0, -3.0
350., -3.0, -3.0, -3.0
</s2_h>
#
# phase-shift(D, unit: mm) for AC-steering magnet S2y
<s2phase_v>
-121., 0. ,121.
</s2phase_v>
#
# excitation current (unit: A)
<s2_v>
35. , 3.0, 3.0, 3.0
300., 3.0, 3.0, 3.0
```

```
350., 3.0, 3.0, 3.0
</s2_v>
#
# phase-shift(D, unit: mm) for AC-steering magnet S3x
<s3phase_h>
-121., 0. ,121.
</s3phase_h>
#
# excitation current (unit: A)
<s3_h>
35. , -4.0, -4.0, -4.0
300., -4.0, -4.0, -4.0
350., -4.0, -4.0, -4.0
</s3_h>
#
# phase-shift(D, unit: mm) for AC-steering magnet S3y
<s3phase_v>
-121., 0. ,121.
</s3phase_v>
#
# excitation current (unit: A)
<s3_v>
35. , 4.0, 4.0, 4.0
300., 4.0, 4.0, 4.0
350., 4.0, 4.0, 4.0
</s3_v>
#
# phase-shift(D, unit: mm) for AC-steering magnet S4x
<s4phase_h>
-121., 0. ,121.
</s4phase_h>
#
# excitation current (unit: A)
<s4_h>
35. , -5.0, -5.0, -5.0
300., -5.0, -5.0, -5.0
350., -5.0, -5.0, -5.0
</s4_h>
#
# phase-shift(D, unit: mm) for AC-steering magnet S4y
<s4phase_v>
-121., 0. ,121.
</s4phase_v>
#
# excitation current (unit: A)
<s4_v>
35. , 5.0, 5.0, 5.0
300., 5.0, 5.0, 5.0
350., 5.0, 5.0, 5.0
</s4_v>
#
#
```

国際単位系 (SI) と換算表

表 1　SI 基本単位および補助単位

量	名称	記号
長さ	メートル	m
質量	キログラム	kg
時間	秒	s
電流	アンペア	A
熱力学温度	ケルビン	K
物質量	モル	mol
光度	カンデラ	cd
平面角	ラジアン	rad
立体角	ステラジアン	sr

表 2　SI と併用される単位

名称	記号
分，時，日	min，h，d
度，分，秒	°，′，″
リットル	l，L
トン	t
電子ボルト	eV
原子質量単位	u

1 eV = 1.60218 × 10^{-19} J

1 u = 1.66054 × 10^{-27} kg

表 5　SI 接頭語

倍数	接頭語	記号
10^{18}	エクサ	E
10^{15}	ペタ	P
10^{12}	テラ	T
10^{9}	ギガ	G
10^{6}	メガ	M
10^{3}	キロ	k
10^{2}	ヘクト	h
10^{1}	デカ	da
10^{-1}	デシ	d
10^{-2}	センチ	c
10^{-3}	ミリ	m
10^{-6}	マイクロ	μ
10^{-9}	ナノ	n
10^{-12}	ピコ	p
10^{-15}	フェムト	f
10^{-18}	アト	a

表 3　固有の名称をもつ SI 組立単位

量	名称	記号	他の SI 単位による表現
周波数	ヘルツ	Hz	s^{-1}
力	ニュートン	N	$m \cdot kg/s^2$
圧力，応力	パスカル	Pa	N/m^2
エネルギー，仕事，熱量	ジュール	J	N·m
工率，放射束	ワット	W	J/s
電気量，電荷	クーロン	C	A·s
電位，電圧，起電力	ボルト	V	W/A
静電容量	ファラド	F	C/V
電気抵抗	オーム	Ω	V/A
コンダクタンス	ジーメンス	S	A/V
磁束	ウェーバ	Wb	V·s
磁束密度	テスラ	T	Wb/m^2
インダクタンス	ヘンリー	H	Wb/A
セルシウス温度	セルシウス度	℃	
光束	ルーメン	lm	cd·sr
照度	ルクス	lx	lm/m^2
放射能	ベクレル	Bq	s^{-1}
吸収線量	グレイ	Gy	J/kg
線量当量	シーベルト	Sv	J/kg

表 4　SI と共に暫定的に維持される単位

名称	記号
オングストローム	Å
バーン	b
バール	bar
ガル	Gal
キュリー	Ci
レントゲン	R
ラド	rad
レム	rem

1 Å = 0.1 nm = 10^{-10} m

1 b = 100 fm^2 = 10^{-28} m^2

1 bar = 0.1 MPa = 10^5 Pa

1 Gal = 1 cm/s^2 = 10^{-2} m/s^2

1 Ci = 3.7 × 10^{10} Bq

1 R = 2.58 × 10^{-4} C/kg

1 rad = 1 cGy = 10^{-2} Gy

1 rem = 1 cSv = 10^{-2} Sv

（注）

1. 表 1－5 は「国際単位系」第 5 版，国際度量衡局 1985 年刊行による。ただし，1 eV および 1 u の値は CODATA の 1986 年推奨値によった。
2. 表 4 には海里，ノット，アール，ヘクタールも含まれているが日常の単位なのでここでは省略した。
3. bar は，JIS では流体の圧力を表わす場合に限り表 2 のカテゴリーに分類されている。
4. EC 閣僚理事会指令では bar，barn および「血圧の単位」mmHg を表 2 のカテゴリーに入れている。

換　算　表

力	N(=10^5 dyn)	kgf	lbf
	1	0.101972	0.224809
	9.80665	1	2.20462
	4.44822	0.453592	1

粘度　1 Pa·s(N·s/m^2) = 10 P(ポアズ)(g/(cm·s))

動粘度　1 m^2/s = 10^4 St(ストークス)(cm^2/s)

圧力	MPa(=10 bar)	kgf/cm^2	atm	mmHg(Torr)	lbf/in^2(psi)
	1	10.1972	9.86923	7.50062 × 10^3	145.038
	0.0980665	1	0.967841	735.559	14.2233
	0.101325	1.03323	1	760	14.6959
	1.33322 × 10^{-4}	1.35951 × 10^{-3}	1.31579 × 10^{-3}	1	1.93368 × 10^{-2}
	6.89476 × 10^{-3}	7.03070 × 10^{-2}	6.80460 × 10^{-2}	51.7149	1

エネルギー・仕事・熱量	J(=10^7 erg)	kgf·m	kW·h	cal(計量法)	Btu	ft·lbf	eV
	1	0.101972	2.77778 × 10^{-7}	0.238889	9.47813 × 10^{-4}	0.737562	6.24150 × 10^{18}
	9.80665	1	2.72407 × 10^{-6}	2.34270	9.29487 × 10^{-3}	7.23301	6.12082 × 10^{19}
	3.6 × 10^6	3.67098 × 10^5	1	8.59999 × 10^5	3412.13	2.65522 × 10^6	2.24694 × 10^{25}
	4.18605	0.426858	1.16279 × 10^{-6}	1	3.96759 × 10^{-3}	3.08747	2.61272 × 10^{19}
	1055.06	107.586	2.93072 × 10^{-4}	252.042	1	778.172	6.58515 × 10^{21}
	1.35582	0.138255	3.76616 × 10^{-7}	0.323890	1.28506 × 10^{-3}	1	8.46233 × 10^{18}
	1.60218 × 10^{-19}	1.63377 × 10^{-20}	4.45050 × 10^{-26}	3.82743 × 10^{-20}	1.51857 × 10^{-22}	1.18171 × 10^{-19}	1

1 cal = 4.18605 J (計量法)

= 4.184 J (熱化学)

= 4.1855 J (15 °C)

= 4.1868 J (国際蒸気表)

仕事率　1 PS (仏馬力)

= 75 kgf·m/s

= 735.499 W

放射能	Bq	Ci
	1	2.70270 × 10^{-11}
	3.7 × 10^{10}	1

吸収線量	Gy	rad
	1	100
	0.01	1

照射線量	C/kg	R
	1	3876
	2.58 × 10^{-4}	1

線量当量	Sv	rem
	1	100
	0.01	1

（86 年 12 月 26 日現在）

原研軟X線ビームライン(BL23SU)用挿入光源の制御系の開発