



株式会社 安川電機

サーボドライブΣ-7 シリーズ 開発コード : Mercury

ソフトウェアアーキテクチャ仕様書

対象製品 :

Σ-7S モデル SGD7S-□□□□□□□□

Σ-7W モデル SGD7W-□□□□□□□□

本書は社内用です。本書を客先に提出することを禁じます。

| 改版・備考 | 配布先 | 部数 | 配布先 | 部数 | 担当部門 | モーションコントロール事業部 サーボドライブ技術部 | | |
|-------|-----|----|-----|----|------|------------------------------|----|----|
| | | 部 | | 部 | | 承認 | 照査 | 作成 |
| | | 部 | | 部 | | | | |
| | | 部 | | 部 | | | 岡 | |
| | | 部 | | 部 | | | | |
| | | 部 | | 部 | | | | |
| | | 部 | | 部 | | | | |
| | | 部 | | 部 | 図版 | 900-130-147 | | |

| 改 版 履 歴 | | | | |
|---------|------|-----|-------|------------|
| 版数 | 改版内容 | 改版頁 | 作成/承認 | 日 付 |
| 0.1 | 新規作成 | - | 岡 | 2013/02/04 |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

目次

| | | |
|--------|--|----|
| 1. | はじめに | 5 |
| 2. | 設計方針 | 6 |
| 2.1. | 基本方針 | 6 |
| 2.2. | ベースソフトウェア | 6 |
| 3. | ハードウェア制御 | 7 |
| 3.1. | ハードウェア基本構成 | 7 |
| 3.2. | JL-086 (SoC) スペック | 9 |
| 3.2.1. | 概略仕様 | 9 |
| 3.2.1. | ブロック図 | 10 |
| 3.3. | IF 基板 H/W 制御 | 11 |
| 3.4. | CB 基板 H/W 制御 | 13 |
| 3.5. | CC 基板 H/W 制御 | 13 |
| 4. | メモリ管理 | 14 |
| 4.1. | メモリマップ一覧 | 15 |
| 4.2. | 内蔵 RAM 仕様 | 16 |
| 4.3. | CPU キャッシュ | 16 |
| 4.4. | Serial Flash メモリ (2Mbyte) | 16 |
| 4.5. | EEPROM | 17 |
| 4.5.1. | EEPROM (1 軸目) | 18 |
| 4.5.2. | EEPROM (2 軸目) | 19 |
| 5. | 割り込み | 20 |
| 5.1. | 割り込み構成 | 20 |
| 5.2. | 割り込み同期 | 21 |
| 5.3. | M-III 125us 対応時のタスクタイミング | 22 |
| 5.4. | タスク同期合わせ | 22 |
| 6. | ソフトウェア構成 | 23 |
| 6.1. | ソフトウェアモジュール構成 | 24 |
| 6.1.1. | Common Library | 25 |
| 6.1.2. | Servo H/W Abstraction Layer | 25 |
| 6.1.3. | Servo Control Layer | 25 |
| 6.1.4. | System Application & Network Control Layer | 25 |
| 6.2. | サーボ制御ブロック図 | 27 |
| 6.3. | ソフトウェアアーキテクチャ | 28 |
| 6.3.1. | 多軸対応 | 28 |
| 6.3.2. | 機能のモジュール化 | 32 |
| 6.4. | マイクロプログラム | 34 |
| 7. | サーボレジスタ管理 | 35 |
| 7.1. | レジスタアクセスモデル | 35 |
| 7.2. | レジスタ登録 | 35 |
| 8. | ファームウェアダウンロード | 36 |

| | |
|-------------------------------|----|
| 9. データベース管理 | 37 |
| 10. エラー検出 | 38 |
| 10.1. サーボアラーム | 38 |
| 10.2. サーボワーニング | 38 |
| 10.3. CPU 異常検出 | 38 |
| 11. 新規開発・ネック技術一覧 | 39 |
| 12. 開発環境 | 40 |
| 12.1. ソフトウェアビルド環境 | 40 |
| 12.2. クロスコンパイル環境 | 40 |
| 12.3. マイクロプログラムビルド環境 | 40 |
| 12.4. プロジェクトフォルダ構成 | 41 |
| 12.5. コーディングルール | 42 |
| 13. 添付資料 | 44 |
| 13.1. 割り込み処理概要 | 44 |
| 13.2. タスク処理概要 | 44 |
| 13.3. ASIC-CPU 間データ転送 | 45 |
| 13.4. サーボ制御ブロック図 | 47 |
| 13.5. サーボ制御処理分担 | 48 |
| 14. ユーザアプリケーション I/F（暫定） | 50 |
| 14.1. 目的・用途 | 50 |
| 14.2. ユーザへの提供資料・ツール | 50 |
| 14.3. ユーザアプリフレームワーク | 51 |
| 14.3.1. ユーザタスク起動 | 51 |
| 14.3.2. フレームワーク概略 | 52 |
| 14.3.3. 実行タイミング | 53 |
| 14.4. ユーザアプリケーション IF | 54 |

1. はじめに

この文書では、サーボドライブ Σ -7 シリーズ (Mercury) のソフトウェア基本アーキテクチャについて記す。

本書の記載内容：

- ・ Σ -7 シリーズソフトウェア全体のアーキテクチャ
(タスク構成・タスク実行タイミング、メモリ構成、S/W モジュール構成等)
- ・ ハードウェアに対するソフト処理・機能の概要
- ・ 開発環境、処理系依存箇所の定義 (ソフトウェアの再利用ができない箇所の明確化(局所化))

【関連資料】

| 資料 | 資料名称 | 図番 |
|-------|--|----------------|
| 製品仕様書 | AC サーボドライブ Σ -7 シリーズサーボパック 製品仕様書 | 900-122-596 |
| | ASIC-PF TYPE-C10 リファレンス・インプリ機能仕様書 | NPPFTYPEC10V10 |
| | SS-Pro7 仕様書 | R09_20130206 |
| | Mercury サーボ SoC (JL-086)_サーボ IP モジュール仕様書 | RB1400401 |
| | | |

【言葉の定義】

| 言葉 | 定義 |
|--------|--|
| JL-086 | Σ -7S/ Σ -7W に搭載する SoC (System on Chip) の名称。ARM CortexR4F, サーボ IP (JL-076 の機能拡張版), M-III IP (JL-102 相当) を 1Chip にまとめたもの。 |
| モジュール | ある機能を実現するためのデータと処理の集まり。 |
| タスク | 割り込み処理の内容。ScanA タスクは ScanA 処理を指す。 |
| | |
| | |

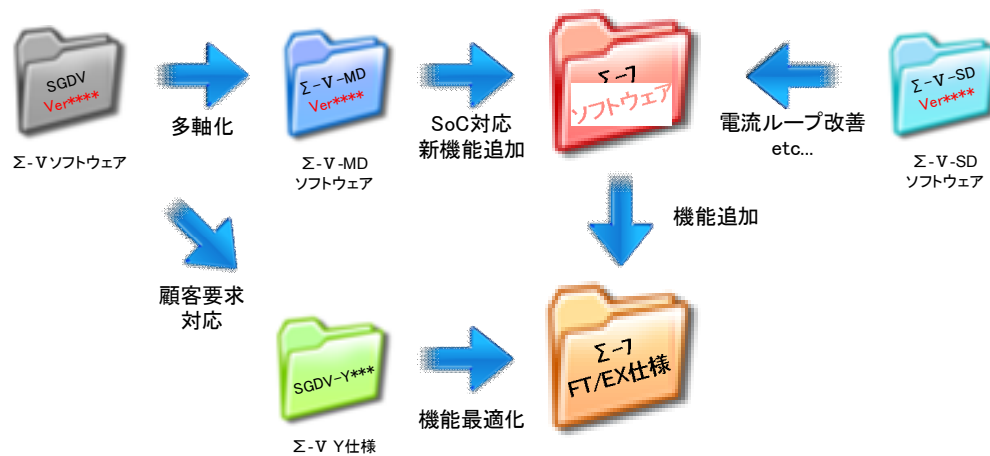
2. 設計方針

2.1. 基本方針

- ・基本的に Σ -Vの仕様、制御アルゴリズムを継承して開発する。
- ・1軸～複数軸まで対応できるソフトウェア構造にする。
- ・機能毎にコンポーネント化を図り、再利用が容易なモジュール構成とする。
(ハードウェア処理は、レイヤを分離し局所化することで、基板変更に対応される構造にする)

2.2. ベースソフトウェア

Σ -7シリーズのソフトウェアは、 Σ -Vのソフトウェアを多軸化した Σ -V-MDをベースとして開発する。また、工作機械用サーボとして開発された Σ -V-SDからも電流ループの改善策やI/F固有処理の共通化、工作機に特化した制御など、有用な機能を取り込む。さらに、 Σ -VのY仕様機能を機能単位で最適化したFT仕様やEX仕様の標準化を実施する。

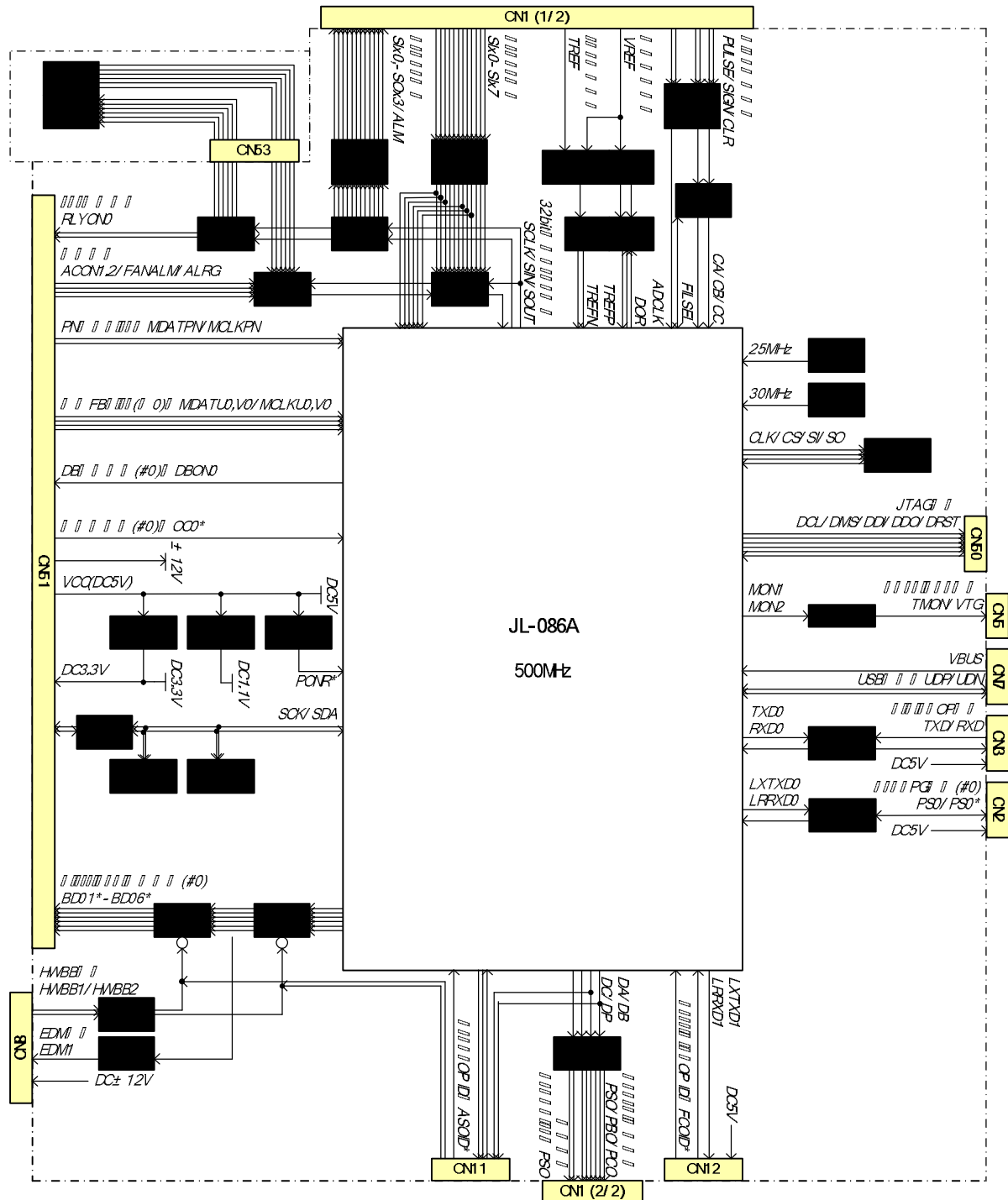


3. ハードウェア制御

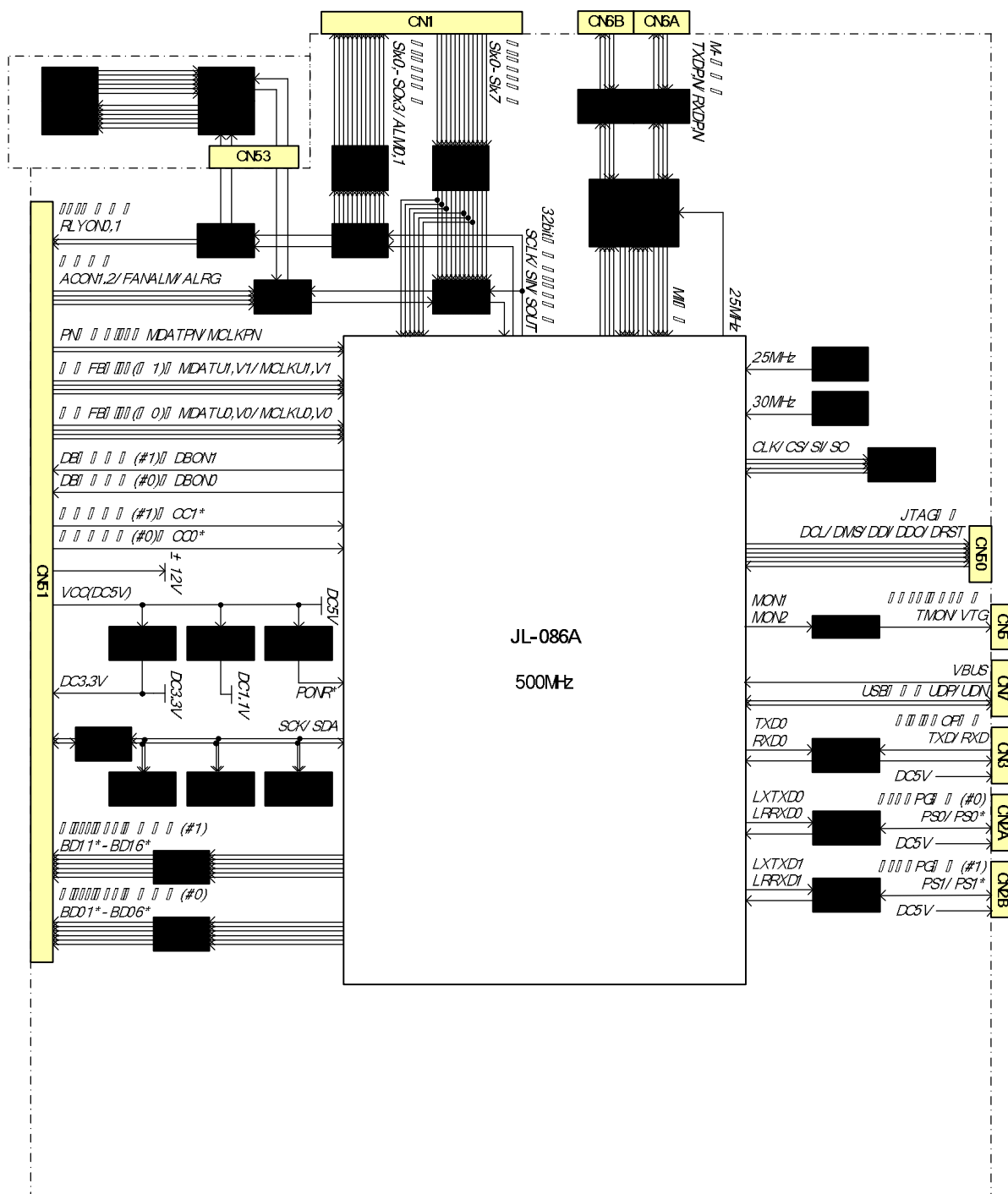
H/W に対するソフトウェア処理、機能の概要を以降に示す。

3.1. ハードウェア基本構成

代表機種として、 Σ -7S モデルのアナログ電圧・パルス列指令形、および Σ -7W モデルの H/W 構成概略ブロック図を以下に示す。



H/W ブロック図(Σ-7S モデル/アナログ電圧・パルス列指令形)



H/W ブロック図 (Σ-7W モデル/MECHATROLINK-III 通信指令形)

3.2. JL-086 (SoC) スペック

Σ-7 シリーズに搭載する JL-086 (SoC) のスペックを以下に示す。

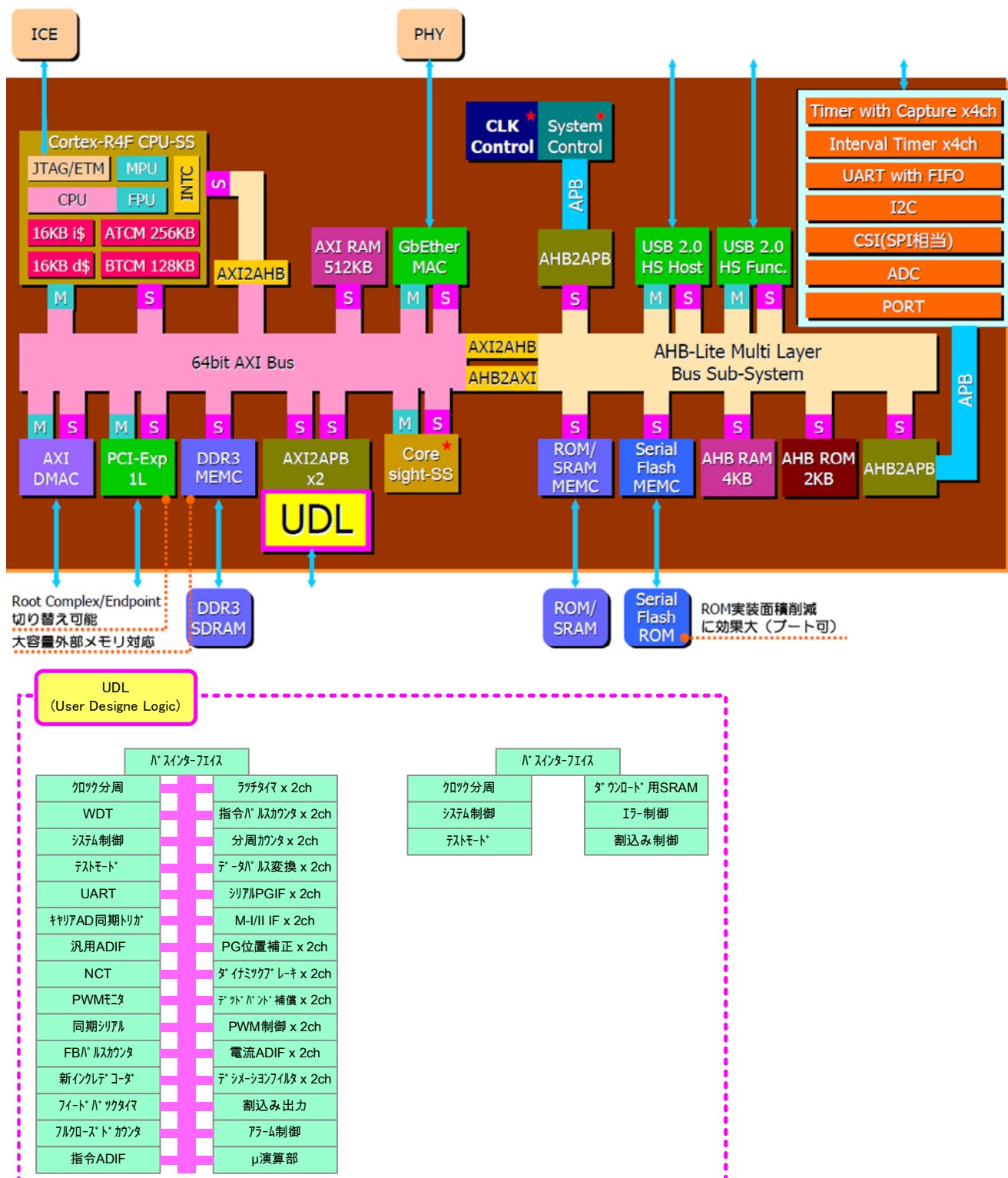
3.2.1. 概略仕様

下表に、JL-086 の概略仕様を示す。

| 項目 | 仕様 |
|------------------------------|--|
| CPU コア | Cortex-R4F Cache 命令 : 16KB/データ : 16KB, 500MHz or 250MHz, 下限電圧 1.05V , FPU, MPU 内蔵, ECC なし |
| オンチップ・デバッグ機能 | ETB : 4KB, トレース機能あり, トレース端子なし |
| TCM | ATCM : 192Kbyte 1-wait (※アクセス領域は 256Kbyte) BTCM : 96Kbyte no-wait (※アクセス領域は 128Kbyte) (BOTCM/B1TCM に分割) |
| 割り込みコントローラ | AHB 接続 INTC を CPU-SS として 64ch 内蔵 |
| バス仕様 | AXI 系 IP, AHB 系 IP をそれぞれ BUS-SS により, マルチレイヤ構成, 125MHz 固定 |
| 外部接続メモリ 1 | アドレス: 17bit, データ: 16bit, チップ・セレクト信号 2 本 AHB 接続 MEM-SS TYPE-SRAM 内蔵 (SRAM, Page ROM 対応, バス・ホールドなし) |
| 外部接続メモリ 2 | AXI 接続 DDR3 コントローラ内蔵 (512M 空間), DDR3-500 (リマップ機能あり) |
| シリアル Flash ROM メモリ・コントローラ | AHB 接続 MEM-SS TYPE-SROM 内蔵 (2Mbyte 空間) (ブート可能) |
| DMAC | DMA-SS TYPE-AXI 4 チャンネル内蔵 (バッファ 4 段) |
| PCI Express | PCI Express-SS (Gen1 1L×1, Endpoint/Root Complex 選択可能), AXI 接続 |
| Ether | GbEther-SS (10/100/1G Ether, RGMII or GMII 3.3V I/F MAC 外部接続), AXI 接続, バス・マスタ機能内蔵 |
| USB | USB-SS (PHY 内蔵 USB 2.0 Host×1, Function×1), AHB 接続, DMAC 内蔵, エンドポイント数: 4 |
| AXI RAM | 512Kbyte : RAM-SS TYPE-eSRAM AXI (AXI RAM) |
| AHB RAM | 4Kbyte : RAM-SS TYPE-eSRAM AHB (AHB RAM) |
| AHB ROM | 2Kbyte : RAM-SS TYPE-eROM AHB (AHB ROM) |
| キャプチャ対応タイマ | APB 接続 4ch タイマ・アレイ TAUJ (V850E2/MN4 と同一) |
| 汎用インターバルタイマ | APB-SS 4ch |
| I2C | APB-SS, 動作レート default 値 392.6kbps (9.23MHz 供給) 動作レート: 357kbps, 392.6kbps, 400kbps の 3 段階に切り替え可能 |
| CSI (SPI 相当) | APB-SS, FIFO 内蔵, MAX. 15.625Mbps (PCLK = 62.5MHz 時) PCLK=24MHz 供給で通信クロック 3MHz を実現 |
| UART | APB-SS, 19200bps, FIFO 内蔵, CTS, RTS なし |
| GPIO | 16 ポート |
| 外部割り込み | 4 本 (ノイズ・フィルタあり), INTC で有効信号を選択 |
| A/D コンバータ | APB-SS で 12 ビット×4 チャンネル内蔵 |
| UDL 接続 I/F | 割り込み 8 本, DMA I/F 2ch, バス・マスタ機能なし, Servo/Network 別スレーブの AXI2APB 接続 2 ポート |
| クロック、リセット | Main : 25MHz (内蔵 PLL で逡倍 500MHz 系クロック生成) USB + UDL : 30MHz (内蔵 PLL で逡倍 160MHz 系クロック生成) Ether (125MHz, 25MHz, 2.5MHz) と PCIe (差動 100MHz)。 リセットは, 端子からのリセット入力のみ。 リダンダンシ RAM 用 PONR 端子あり。 |
| Servo UDL | サーボ ASIC JL-076 互換機能搭載, 2 軸サーボ制御対応, 割り込み出力の同期機能対応 μプログラム動作周波数 125MHz, 命令ステップ数 10240Step, 演算レジスタ 1536word, 機能レジスタ 512word, 割り込み 4 本 |
| Network UDL | M-III ASIC JL-102 同等機能。 C1/C2 メッセージ処理, 1 チャンネル, 100Mbps |

3.2.1. ブロック図

以下に、JL-086 のブロック図を示す。



3.3. IF 基板 H/W 制御

下表に、H/W 信号・機能に対する、ソフトウェア側での処理内容・使用方法の概略を示す。

| No | HW 信号/ 機能ブロッ ク | Σ -V 又は Σ -V-MD からの変更点 | 関連 SW 機能（処理内容） |
|----|---|---|---|
| 1 | JL-086 (SoC) ARM R4F Servo UDL Network UDL | 新規 | <p>【ARM R4F】</p> <ul style="list-style-type: none"> ・最大 2 軸分のサーボ制御を行うメイン CPU。 ・エンコーダ通信、位置/速度ループ演算、異常検出などを担当。 ・I2C I/F、USB I/F、UART I/F による通信制御。 <p>【Servo UDL (サーボ IP)】</p> <ul style="list-style-type: none"> ・割り込み信号の生成を行う。PWM、通信、エンコーダ、各割り込み処理の完全同期を実現。 ・μ プログラムによる電流ループ演算を担当。 ・M-II I/F による上位コントローラとの通信を行う。 <p>【Network UDL (M-III IP)】</p> <ul style="list-style-type: none"> ・M-III I/F による上位コントローラとの通信を行う。 ・μ プログラムを入れ替えることにより、マルチスレーブとして 通信可能。 ・コアへのアクセスには、MMA のアクセスドライバを使用。 |
| 2 | Serial Flash (2Mbyte) | 新規 | <ul style="list-style-type: none"> ・サーボファームウェア格納用 Flash ROM。 ・電源投入後、Flash からプログラムコードが TCM や AXI、AHB といった RAM へ展開される。 ・初回の Flash 書き込みは、ICE を接続するにより実施する。 ・空き領域を利用し、データ保持用不揮発メモリとしても使用する。 |
| 3 | 高速 RAM ATCM BOTCM B1TCM | 新規 | <p>【ATCM】 192Kbyte</p> <ul style="list-style-type: none"> ・B1TCM と合わせ、プログラム領域として使用。 ・INTSYNC, SCANA, SCANB の一部を配置する。 <p>【BOTCM】 64Kbyte</p> <ul style="list-style-type: none"> ・データ領域として使用。 <p>【B1TCM】 32Kbyte</p> <ul style="list-style-type: none"> ・ATCM と合わせ、プログラム領域として使用。 ・IntSync, ScanA, ScanB の一部を配置する。 |
| 4 | RAM AXI AHB | 新規 | <p>【AXI】 512Kbyte</p> <ul style="list-style-type: none"> ・キャッシュ有効。 ・ScanB の一部、ScanC、ROUND 処理を配置する。 <p>【AHB】 4Kbyte</p> <ul style="list-style-type: none"> ・USB 用ワーク・メモリ用。 |
| 5 | 割り込み入力 同期回路 | 新規 | <ul style="list-style-type: none"> ・Servo UDL 機能。 ・PWM、通信、エンコーダ、各割り込み処理の完全同期が可能。 ・Servo UDL のレジスタにて各割り込み信号のディレイを設定することで、無駄時間を最小化することが可能。 |
| 6 | EEPROM (I2C) (4Mbit) | 新規 | <ul style="list-style-type: none"> ・全軸分のユーザパラメータ、システムパラメータ、およびユニット共通パラメータを格納する ・ARM R4F コアのペリフェラルで用意されている I2C I/F を介してアクセスする。スレーブアドレスは、以下の通り。 <p>1 軸目 : 1010_000 2 軸目 : 1010_001</p> |

| No | HW 信号/ 機能ブロッ ク | Σ -V 又は Σ -V-MD からの変更点 | 関連 SW 機能（処理内容） |
|----|---|---|--|
| 7 | パネルオペレー タ | 新規 | 【A/P 用】 <ul style="list-style-type: none"> ・サーボ状態表示用 7seg 5 桁。 ・プッシュ SW 4 点。 ・Servo UDL の同期シリアル I/F により LED 制御、および SW 入力検出を行う。 |
| | | | 【network 用】 <ul style="list-style-type: none"> ・通信状態監視用 LED。 ・サーボ状態表示用 7seg 1 桁。 ・R-SW、D-SW による通信局アドレス、バイト数設定。 ・Servo UDL の同期シリアル I/F により LED 制御、および SW 状態検出を行う。 |
| 8 | USB | 新規 | <ul style="list-style-type: none"> ・ツールとの USB 通信用 I/F ・USB ドライバは ARM R4F コアのペリフェラルを使用する。 ・MEMOBUS 処理は、サーボ側 S/W にて実現。 |
| 9 | WDT | 新規 | <ul style="list-style-type: none"> ・CPU WDT 検出用。 ・WD 異常の場合、CPU 側に割り込みが通知され、例外割り込みが発生する（システム停止）。 |
| 10 | 温度センサ (I2C) | 新規 | <ul style="list-style-type: none"> ・サーボバック内気温度検出用。 ・S/W 処理にて内気温度を監視し、設置環境モニタ機能にてユーザへ報告する。また、閾値温度を超えた場合には、アラーム処理を行う。 ・ARM R4F コアのペリフェラルで用意されている I2C I/F を介してアクセスする。スレーブアドレスは、以下の通り。 制御部 : 1001_000 パワー部 : 1001_001 |
| 11 | UART | 新規 | <ul style="list-style-type: none"> ・デジタルオペレータ接続用。 ・ARM R4F コアのペリフェラルを使用する。 |
| 12 | HWBB 信号 | Σ -V と同様 | <ul style="list-style-type: none"> ・HWBB 機能用。 ・Servo UDL の同期シリアル I/F により入力状態を取得。 |
| 13 | アナログ/パルス : S00-S05 Network : S00-S02 | Σ -V と同様 | <ul style="list-style-type: none"> ・汎用 I/O 出力用。 ・Servo UDL の同期シリアル I/F により信号を出力。 ・アナログ/パルス形にてアラームコード出力を割り付け可とする。 |
| 14 | アナログ/パルス : SI0-SI8 Network : SI0-SI7 | Σ -V と同様 | <ul style="list-style-type: none"> ・汎用 I/O 入力用。 ・Servo UDL の同期シリアル I/F により入力状態を取得。 ・アナログ/パルス形にて SEN 信号入力を割り付け可とする。 |
| 15 | エンコーダ シリアル I/F | Σ -V と同様 | <ul style="list-style-type: none"> ・エンコーダ通信用。 ・Servo UDL のシリアルエンコーダ I/F を使用する。 ・エンコーダ伝送周波数 8MHz、伝送周期 31.25us、分解能 24bit 対応。（現状のシリアルコンバータ、旧モータとは 4MHz にて通信を行う。） |

3.4. CB 基板 H/W 制御

下表に、H/W 信号・機能に対する、ソフトウェア側での処理内容・使用方法の概略を示す。

| No | HW 信号/ 機能ブロッ ク | Σ -V 又は Σ -V-MD からの変更点 | 関連 SW 機能（処理内容） |
|----|----------------------|---|--|
| 1 | 電流検出 AD | Σ -V と同様 | ・ U 相、V 相の電流検出用。 |
| 2 | DB 制御 (DBON) | Σ -V と同様 | ・ DB 制御用。 ・ Servo UDL の同期シリアル I/F により DB 信号制御を行う。 |
| 3 | PWM 制御信号 | Σ -V と同様 | ・ PWM 制御用。 ・ Servo UDL の PWM 機能により実現。 |
| 4 | ヒートシンク 温度検出 (OH) | Σ -V と同様 | ・ OH アラーム検出用。 ・ OH 信号入力を S/W で監視し、アラームを検出する。 (SGDV では 0V 端子は未使用で、OH アラームは、サーミスタの 温度を監視) |

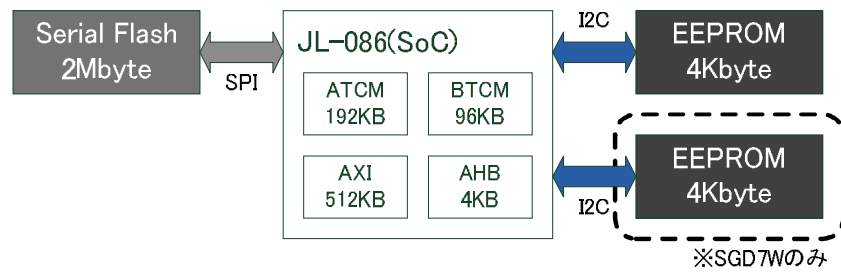
3.5. CC 基板 H/W 制御

下表に、H/W 信号・機能に対する、ソフトウェア側での処理内容・使用方法の概略を示す。

| No | HW 信号/ 機能ブロック | Σ -V 又は Σ -V-MD からの変更点 | 関連 SW 機能（処理内容） |
|----|----------------------|--|--|
| 1 | RLYON (突防リレーON) | Σ -V と同様 | ・ 突防リレー制御用。 ・ Servo UDL の同期シリアル I/F により RLYON 信号制御を 行う。 |
| 2 | RGON (回生 Tr ON) | Σ -V と同様 | ・ 回生トランジスタ制御用。 ・ Servo UDL の同期シリアル I/F により REGON 信号制御を 行う。 |
| 3 | ACON1 (主回路 AC 検出) | Σ -V と同様 | ・ ACON（主回路電源入力）状態検出用。 ・ Servo UDL の同期シリアル I/F により ACON1 信号の検出を 行う。 |
| 4 | ACON2 (主回路 AC 検出) | Σ -V と同様 | ・ ACON（主回路電源入力）状態検出用。 ・ Servo UDL の同期シリアル I/F により ACON2 信号の検出を 行う。 |
| 5 | ALRG (回生異常) | Σ -V と同様 | ・ 回生異常検出用。 ・ Servo UDL の同期シリアル I/F により ALRG 信号の検出を行 う。 |
| 5 | FANALM (FAN 停止) | Σ -V と同様 | ・ ファン停止異常検出用。 ・ Servo UDL の同期シリアル I/F により FANALM 信号の検出を 行う。 ・ 信号検出からアラーム検出までを 300ms とし、ファンワーニ ング を追加する。 |
| 6 | PN 間電圧検出 | Servo UDL の指令 AD I/F (Σ - Δ 変換) を 介して入力 | ・ UV/0V 検出、回生制御、電圧補正、弱め界磁制御用。 ・ AC240V 対応に伴い、UV, 0V、回生レベルを変更。 |

4. メモリ管理

Σ-7S/Σ-7W モデルにおける EEPROM、Flash といった外部メモリと TCM、AXI、AHB といった内部メモリの使用方法、メモリ配置について示す。



4.1. メモリマップ一覧

以下にメモリマップ一覧を示す。

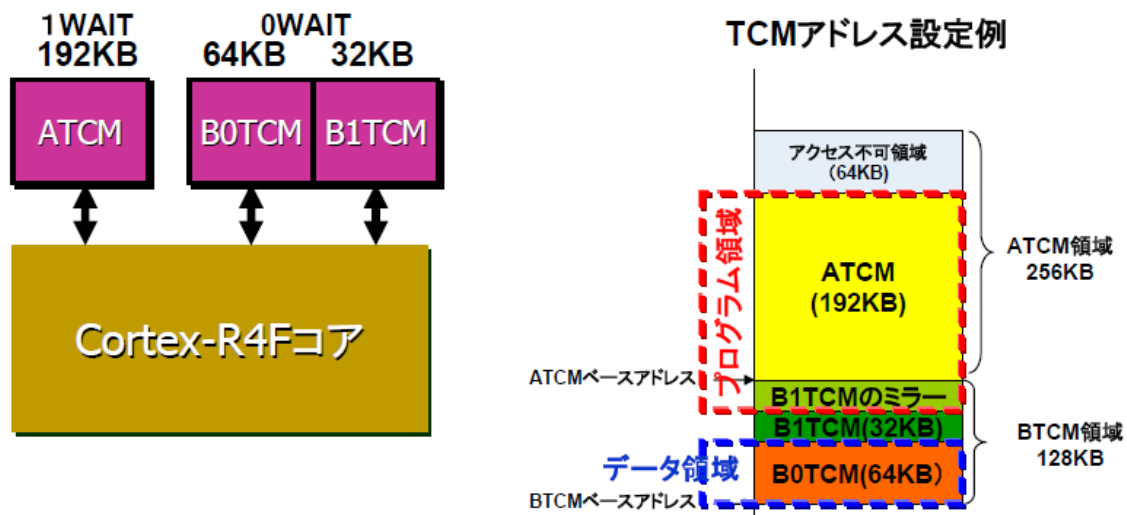
| | | | | |
|-------------|--------------------|--|----------|-------------|
| | BTCM (128 KB) | | () | C000 1500 H |
| F000 0000 H | FLASH (128 MB) | | ASIC / 2 | C000 1420 H |
| E800 0000 H | AHB (128 MB) | | () | C000 1400 H |
| E000 0000 H | SRAM CSZ0 (128 MB) | | ASIC / 2 | C000 1000 H |
| | UDL (512 MB) | | () | C000 0500 H |
| C000 0000 H | | | ASIC / 2 | C000 0420 H |
| 8000 0000 H | () | | ASIC / 1 | C000 0400 H |
| 7000 0000 H | PCIe (128 MB) | | ASIC / 1 | C000 0000 H |
| 6800 0000 H | AXI (128 MB) | | | |
| 6000 0000 H | () | | | |
| | DDR3 (512 MB) | | | |
| 4000 0000 H | | | | |
| 3008 0000 H | () | | | |
| 3004 0000 H | ATCM (128 KB) | | | |
| 3002 0000 H | BTCM (128 KB) | | | |
| 1000 0000 H | () | | | |
| 0800 0000 H | SRAM CSZ1 (128 MB) | | | |
| 0000 0000 H | SRAM CSZ0 (128 MB) | | | |

| 開始アドレス | 終了アドレス | サイズ | メモリ空間 |
|-------------|-------------|-----------|-------------------|
| FFFF 0000 h | FFFF FFFF h | | |
| F000 0000 h | FFFE FFFF h | 128 MByte | シリアル ROM 領域 |
| EF05 0000 h | FFFF FFFF h | | ??? AHB 系レジスタ領域 |
| EF04 0000 h | EF04 1FFF h | 8 KByte | USB (Function) 領域 |
| EF02 0000 h | EF03 FFFF h | 128 KByte | USB (Host) 領域 |
| E800 0000 h | E800 0FFF h | 4 KByte | AHB-RAM 領域 |
| E000 0000 h | E7FF FFFF h | 128 MByte | SRAM_CSZ0 ミラー領域 |
| D000 0000 h | DFFF FFFF h | 256 MByte | UDL (通信 ASIC) 領域 |
| C000 0000 h | CFFF FFFF h | 256 MByte | UDL (サーボ ASIC) 領域 |
| 8000 0000 h | BFFF FFFF h | | (予約) |
| 7000 0000 h | 7FFF FFFF h | 256 MByte | PCIe ウィンドウ領域 |
| 6FEF 0000 h | 6FFF FFFF h | | ??? AXI 系レジスタ領域 |
| 6FEE 0000 h | 6FEE FFFF h | 64 KByte | DMA 領域 |
| 6808 0000 h | 6FED FFFF h | | ??? |
| 6800 0000 h | 6807 FFFF h | 512 KByte | AXI-RAM 領域 |
| 6000 0000 h | 67FF FFFF h | | (予約) |
| 4000 0000 h | 5FFF FFFF h | 512 MByte | DDR3 領域 |
| 3008 0000 h | 3FFF FFFF h | | (予約) |
| 3004 0000 h | 3007 FFFF h | 256 KByte | ATCM 領域 |
| 3002 0000 h | 3003 FFFF h | 128 KByte | BTCM 領域 |
| 1000 0000 h | 3001 FFFF h | | (予約) |
| 0800 0000 h | 0FFF FFFF h | 128 MByte | SRAM_CSZ1 領域 |
| 0000 0000 h | 07FF FFFF h | 128 MByte | SRAM_CSZ0 領域 |

4.2. 内蔵 RAM 仕様

以下に JL-086 内蔵の RAM 仕様を示す。

| アドレス | 名称 | 容量 | バス周波数 | ウェイト | 用途 |
|------------------------------|------|----------|--------|--------|----------------------|
| 3004 0000 h ~ 3006 EE00 h | ATCM | 192Kbyte | 500MHz | 1 ウェイト | プログラム領域を配置 |
| 3002 0000 h ~ 3003 7700 h | BTCM | 96Kbyte | 500MHz | ノーウェイト | プログラム、データ領域を配置。 |
| 6800 0000 h ~ 6807 D000 h | AXI | 512Kbyte | 125MHz | ノーウェイト | TCM に配置しきれなかった処理を配置。 |
| E800 0000 h ~ E800 0FA0 h | AHB | 4Kbyte | 125MHz | ノーウェイト | USB 用ワーク・メモリ |



4.3. CPU キャッシュ

JL-086 は、以下のようにキャッシュが搭載される。

Σ-7S/Σ-7W モデルでは、AXI RAM エリアに対して、キャッシュを有効にして使用する。

(キャッシュ設定は、BSP のスタートアップ処理で設定。)

| キャッシュ | サイズ | 説明 |
|---------|---------|----------------------------|
| i Cache | 16Kbyte | 命令用キャッシュ ※キャッシュロックは不可。 |
| d Cache | 16Kbyte | データ用キャッシュ ※キャッシュロックは不可。 |

4.4. Serial Flash メモリ (2Mbyte)

| アドレス | サイズ | エリア内容 | 備考 |
|------------------------------|----------|-----------------------------|----------------------------------|
| F016 0000 h ~ F01F FFFF h | 512Kbyte | データメモリ | アラームトレース結果等のデータ格納用。 |
| F00B 0000 h ~ F015 FFFF h | 750Kbyte | サーボシステム ファームウェア(ミラ ー) | システムダウンロード失敗時の復旧用ファームウ ェア格納用。 |
| F000 0000 h ~ F00A FFFF h | 750Kbyte | サーボシステム ファームウェア | ブート用サーボファームウェア格納用。 |

4.5. EEPROM

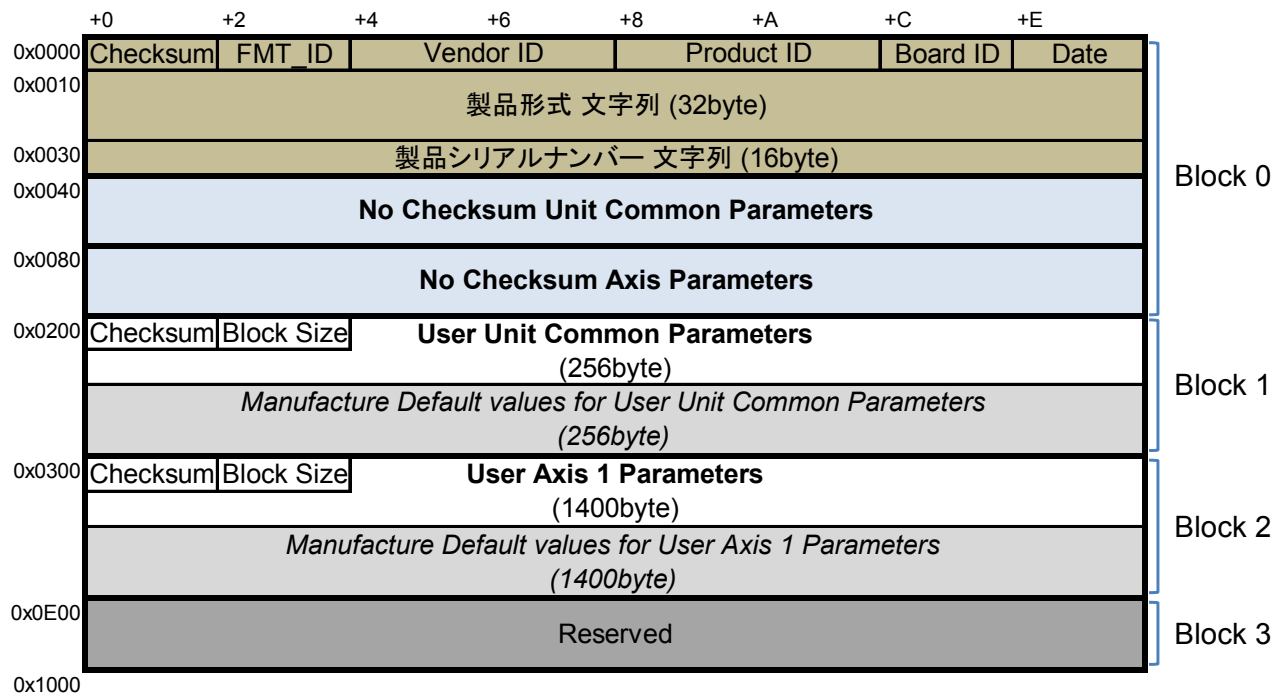
Σ-7S モデルでは一つ、Σ-7W モデルでは二つの EEPROM が搭載される。基本的には軸毎のパラメータはそれぞれの軸用の EEPROM に格納し、ユニットで共通となるパラメータは 1 軸目の EEPROM に格納する。

下表に、それぞれの EEPROM へ格納するパラメータデータの概略を示す。

| 軸番号 | I2C スレーブアドレス | 容量 | EEPROM アクセス | EEPROM で保存するデータ内容 |
|------|-----------------|--------|-----------------------|---|
| 1 軸目 | 0101_000 | 4Kbyte | ARM R4F コア I2C I/F | 【ユニット共通パラメータ】 <ul style="list-style-type: none"> ・ 製品形式 ・ 製品シリアルナンバー ・ 電圧 (システムパラメータ値) ・ PN 電圧検出 A/D 調整値 (システムパラメータ値) ・ UV, OV, 回生電圧レベル (システムパラメータ値) ・ 回生抵抗容量 (内蔵・外付け) ・ 回生抵抗値 (内蔵・外付け) ・ 瞬停保持時間 ・ アナログモニタ A/D 調整値 ・ 内気温度アラームレベル, ワーニングレベル (制御部用, パワー部用) <hr/> 【軸毎パラメータ】 <ul style="list-style-type: none"> ・ 軸毎のシステムパラメータ値 PnExx (1 軸目用) ・ 軸毎のユーザパラメータ Pn000~Pnxxx (1 軸目用) |
| 2 軸目 | 0101_001 | 4Kbyte | ARM R4F コア I2C I/F | 【軸毎パラメータ】 <ul style="list-style-type: none"> ・ 軸毎のシステムパラメータ値 PnExx (2 軸目用) ・ 軸毎のユーザパラメータ Pn000~Pnxxx (2 軸目用) |

4.5.1. EEPROM(1 軸目)

1 軸目用 EEPROM データのメモリマップを以下に示す。



| シンボル | 説明 | 備考 |
|--|--|-----------|
| FMT_ID | EEPROM フォーマット ID 上記のフォーマットが変更された場合に ID をインクリメントしていく (この値を変える場合は、ファームウェアも一緒に変わる) | |
| Chksum | ブロック毎のチェックサム(ブロック内の全ての 16bit データの和が 0 となる値) ※ただし、Block0 については、0x0000~0x0040 までの sum 値 | |
| Vendor ID | ベンダ ID | |
| Product ID | 製品 ID(レジスタ仕様書記載の機種 ID を格納する) | |
| Board ID | 基板 ID(レジスタ仕様書記載のモジュール ID を格納する) | 予約 |
| Date | 製造年月 | |
| 製品形式 | 製品板形式文字列を格納する | 出荷試験機にて設定 |
| 製品シリアル No | 製品シリアル No 文字列を格納する | |
| No Checksum Unit Common Parameters | ユニット共通データのうち、タイムスタンプなどサムチェックを行わないパラメータを格納するエリア。 | (1) 参照 |
| No Checksum Axis Parameters | アラーム履歴、アラームタイムスタンプ、OL 発生状態、軸名称等 サムチェックを行わないデータを格納するエリア。 | (2) 参照 |
| User Unit Common Parameters | ユニット共通パラメータ (Pnxxx) 保存領域 | |
| Manufacture Default values for User Unit Common Parameters | ユニット共通パラメータ (Pnxxx) の工場出荷時設定保存領域 | |
| User Axis n Parameters | 1 軸目のユーザパラメータ (Pnxxx) 保存領域 | |
| Manufacture Default values for User Axis Parameters | 1 軸目のユーザパラメータ (Pnxxx) 工場出荷時設定保存領域 | |

(1) No Checksum Unit Common Parameters

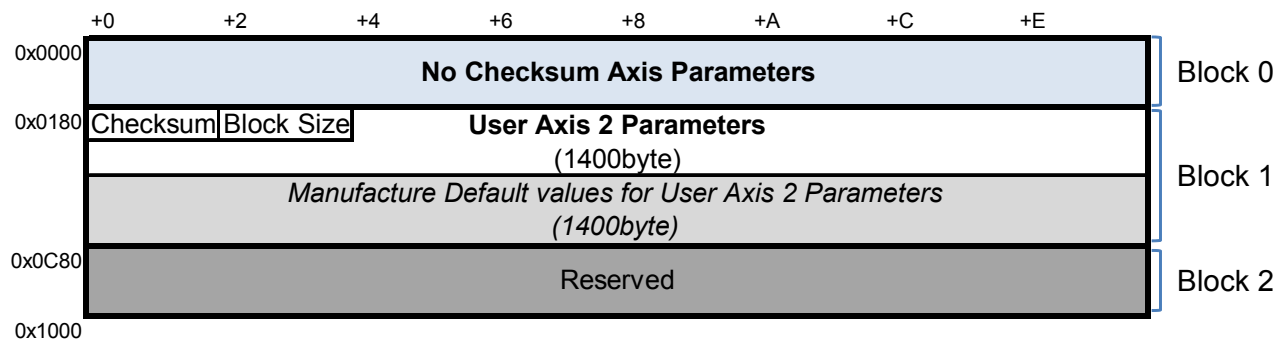
| EEPROM アドレス | 内容 | 型(サイズ) |
|-----------------|--------------|-----------|
| 0x0040 - 0x0043 | タイムスタンプ | ULONG |
| 0x0044 - 0x0045 | オプションカード接続履歴 | USHORT |
| 0x0046 - 0x004B | オプションカード ID | USHORT[3] |
| 0x004C - 0x0055 | BOOT 情報 | USHORT[5] |
| 0x0056 - 0x007F | Reserved | UCHAR[42] |

(2) No Checksum Axis Parameters

| EEPROM アドレス | 内容 | 型(サイズ) |
|-----------------|----------------------------|------------|
| 0x0080 - 0x0081 | モータ・アンプ過負荷情報 | USHORT |
| 0x0082 - 0x0083 | アラーム履歴発生個数、現在 Index | USHORT |
| 0x0084 - 0x0097 | アラーム履歴コード | USHORT[10] |
| 0x0098 - 0x00BF | アラーム履歴タイムスタンプ | ULONG[10] |
| 0x00C0 - 0x00CF | エンジニアリング情報(軸名称/シリアル No) | UCHAR[16] |
| 0x00D0 - 0x00D1 | 前回接続したモータ種別(回転: 0, リニア: 1) | USHORT |
| 0x00D2 - 0x00E1 | 前回接続したモータのシリアル No | UCHAR[16] |
| 0x00E2 - 0x0121 | アラームラッチデータ | USHORT[32] |
| 0x0122 - 0x018F | デジタルオペレータ校構成面情報 | USHORT[55] |
| 0x0190 - 0x01BF | アラーム発生時の通信コマンドデータ | USHORT[24] |
| 0x01C0 - 0x01EF | アラーム発生時の通信レスポンスデータ | USHORT[24] |
| 0x01F0 - 0x01FF | Reserved | UCHAR[16] |

4.5.2. EEPROM (2 軸目)

2 軸目用 EEPROM データのメモリマップを以下に示す。



| シンボル | 説明 | 備考 |
|---|--|-----------|
| No Checksum Axis Parameters | アラーム履歴、アラームタイムスタンプ、OL 発生状態、軸名称等 サムチェックを行わないデータを格納するエリア。 | 前節 (2) 参照 |
| User Axis n Parameters | 2 軸目のユーザパラメータ (Pnxxx) 保存領域 | |
| Manufacture Default values for User Axis Parameters | 2 軸目のユーザパラメータ (Pnxxx) 工場出荷時設定保存領域 | |

5. 割り込み

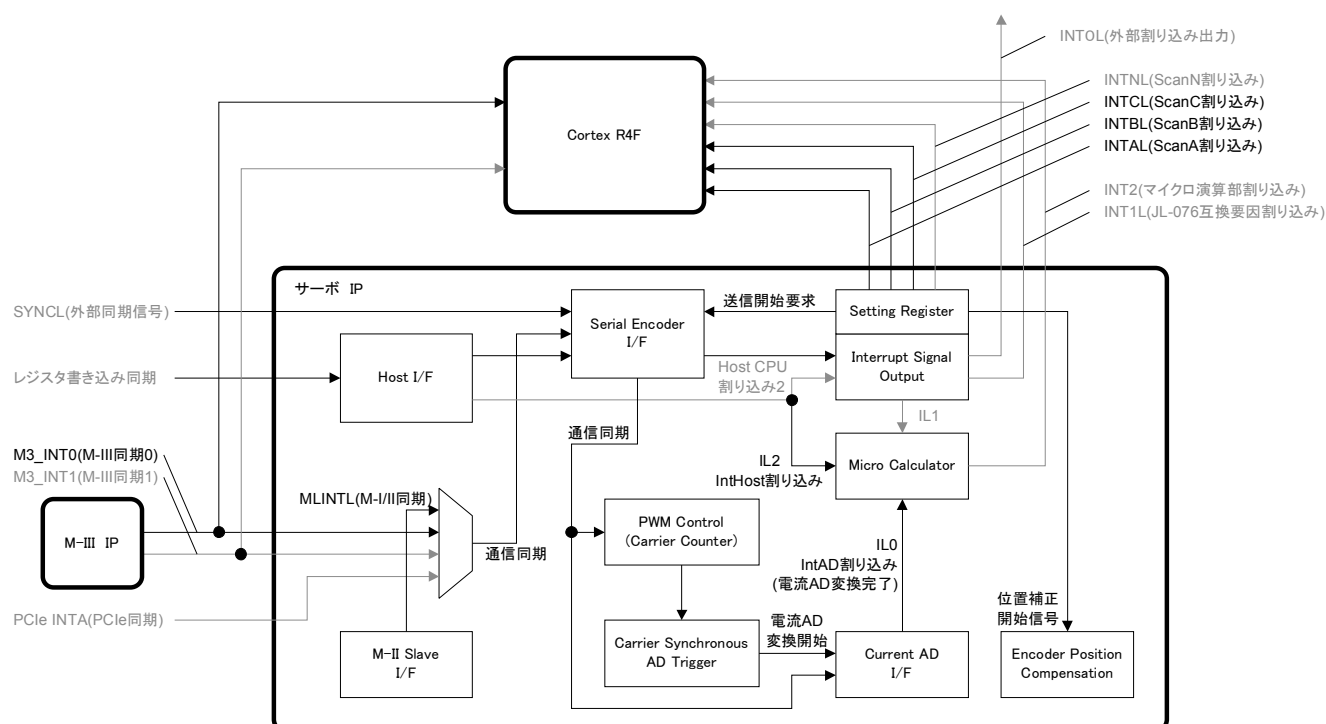
5.1. 割り込み構成

Σ-7S/Σ-7W モデルにおける割り込み構成を下表に示す。

| 優先度 | 割り込み名称 | 処理内容 | 備考 |
|-----|---------|--------------------------|------------------------|
| 0 | IntSync | 上位通信状態検出、トルグル切り替え | Cortex R4F コア 割り込み |
| 2 | ScanA | サーボ制御用高速スキャン処理 | |
| 4 | ScanB | サーボ制御用低速スキャン処理、指令解析・応答生成 | |
| 6 | ScanC | サーボ制御用シーケンス処理、異常検出処理 | |
| 0 | IntHost | トルク指令フィルタ処理 | サーボ IP μ プログラム割り込み |
| 1 | IntAD | 電流制御処理 | |

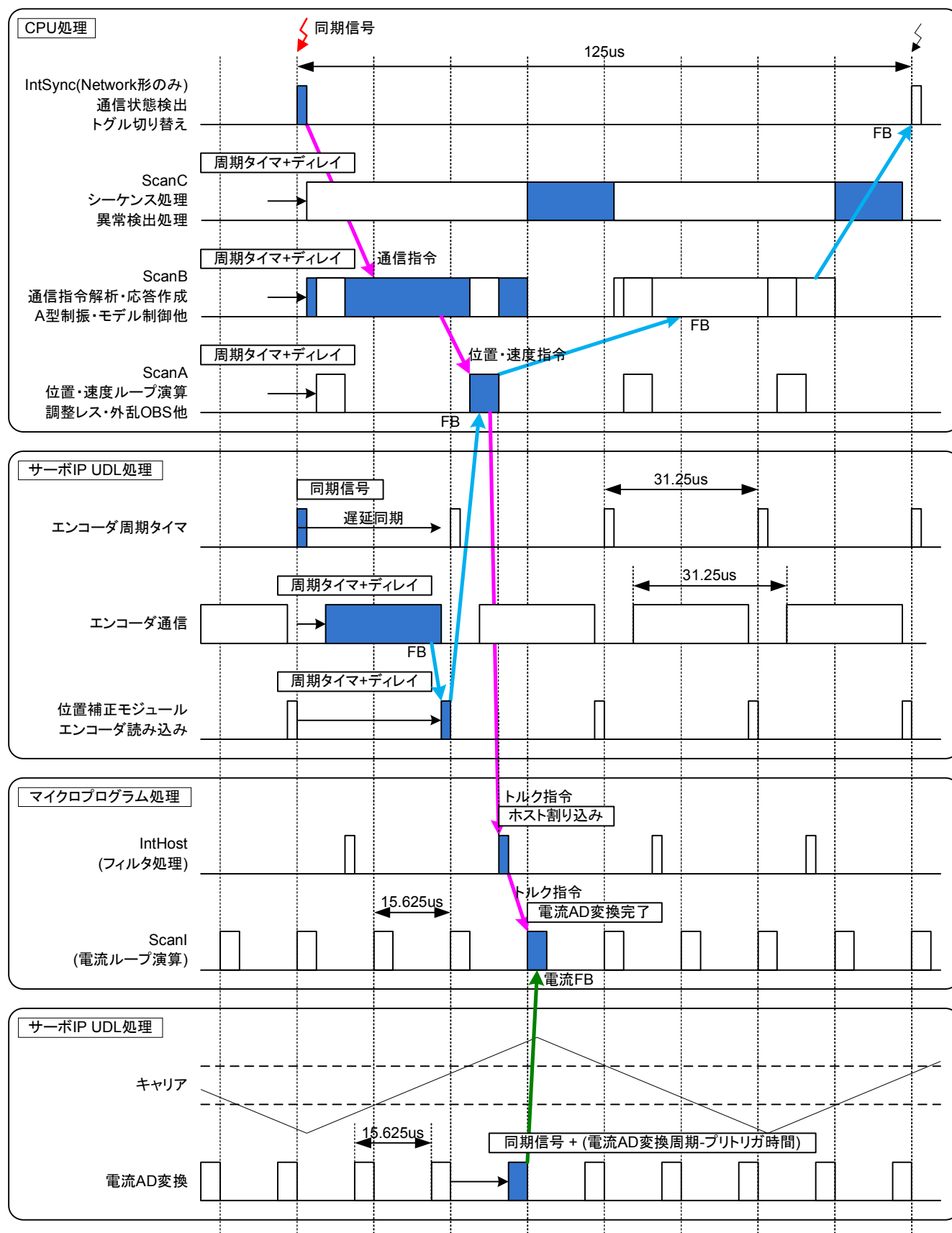
※「優先度」は、数字が小さいほど優先度が高いことを示す。

Σ-7S/Σ-7W モデルでは、M-III IP による同期割り込み以外の割り込みは全て JL-086 サーボ IP による割り込み出力機能を使用する。以下に JL-086 の割り込み出力のブロック図を示す。黒色の信号が使用する割り込み信号を表す。



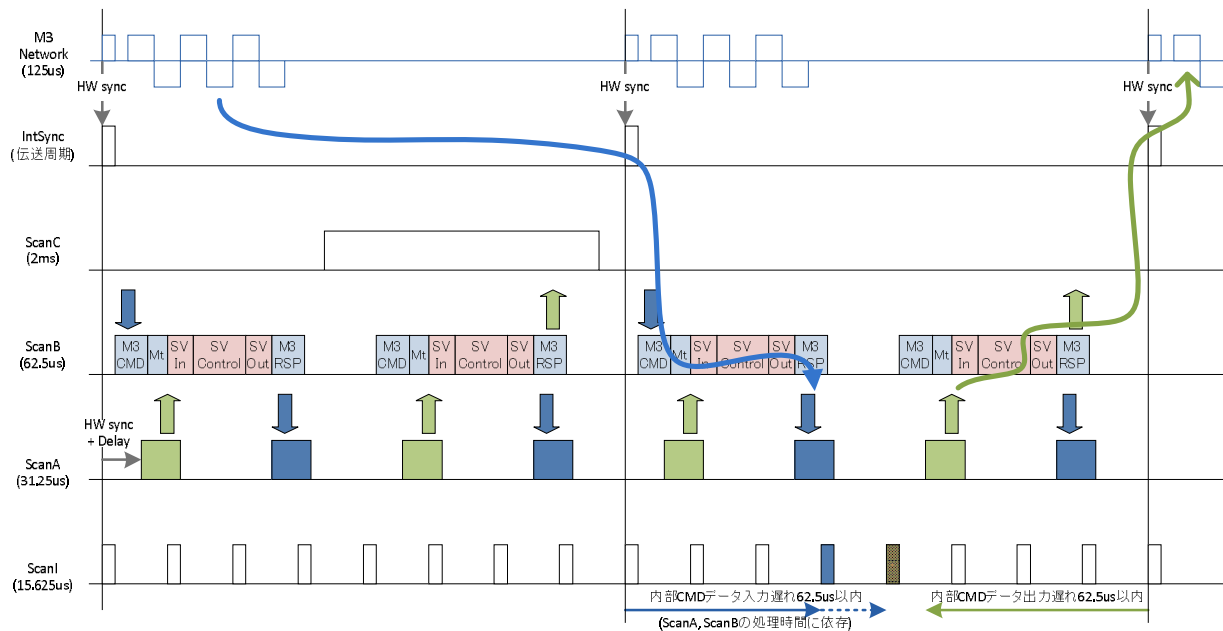
5.2. 割り込み同期

JL-086 では特に割り込み出力機能が強化されており、各割り込みの同期機能や遅延時間設定といった機能を有している。さらに、同期信号(レジスタ書き込み同期、外部同期、通信同期)によるエンコーダ周期タイマ、キャリア、電流 AD 変換の同期が可能である。これらを適宜組み合わせることにより、内部の遅れ時間を最小化した割り込み生成を実現する。**A/P 形は H/W による同期信号入力で OK?**



5.3. M-III 125us 対応時のタスクタイミング

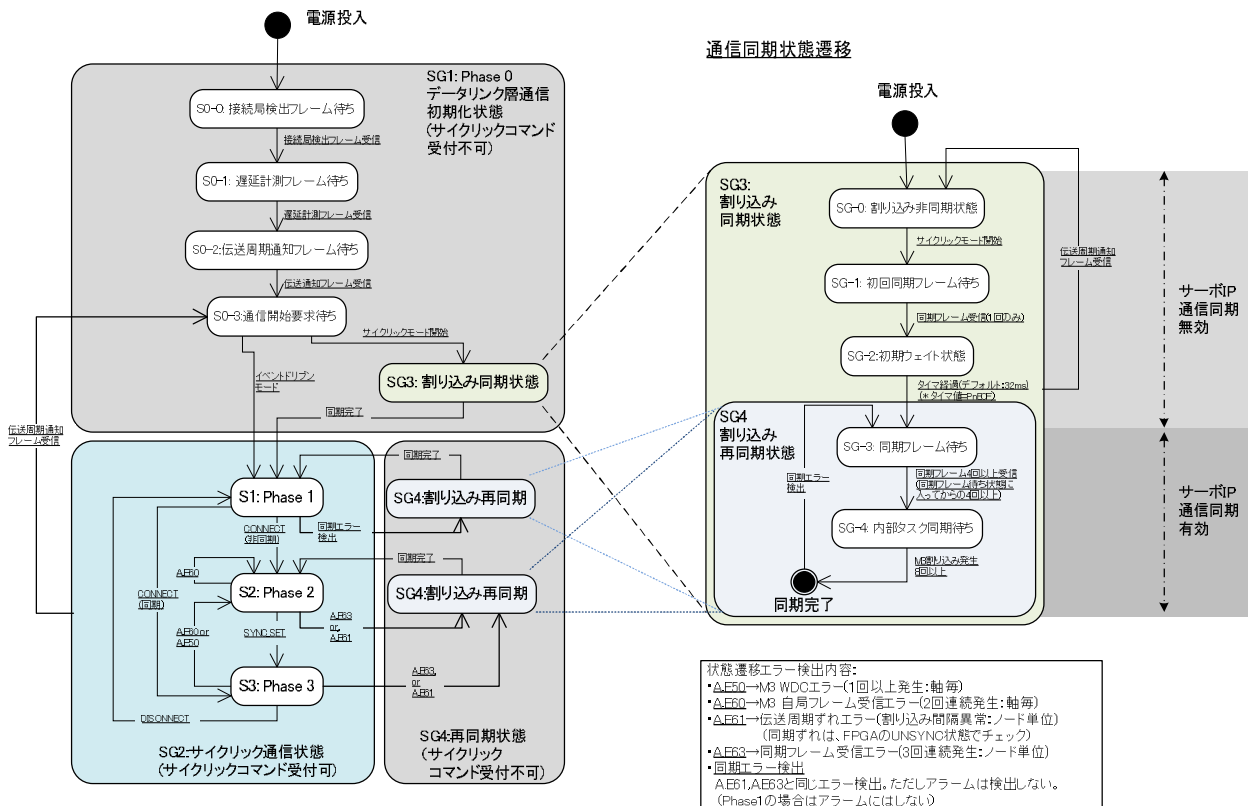
下図に、M-III 伝送周期 125us 時のタスク起動タイミング、およびタスク間のデータフローを示す。



5.4. タスク同期合わせ

下図に、M-III 時の通信状態遷移、及びタスク同期処理の状態遷移を示す。要・不要の判断必要。

MECHATROLINK通信状態遷移
























































処理概要:

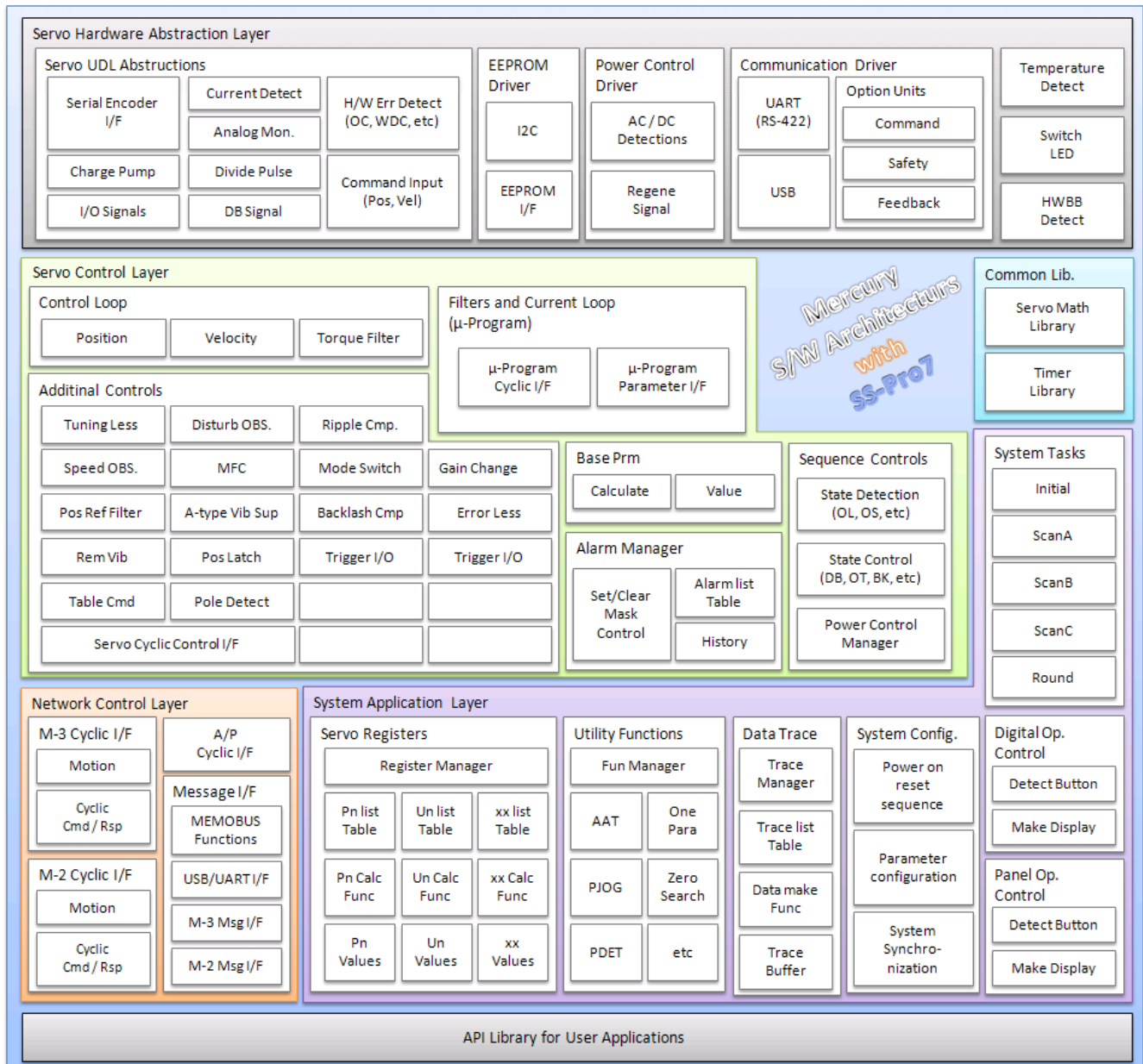
- (1) サイクリック開始後、最初のグローバルフレームを受信してから、数 10ms 以上待ち、M3 割り込みが落ち着いてから、サーボ IP による同期を有効にする。待ち時間はシステムパラメータ (default: 32ms) にて設定する。
- (2) 一度、割り込み同期を行った後でも、以下の場合は、割り込み再同期をかける。
 - ・伝送周期フレームを再受信した場合(同期確立後、マスタがリセットした場合)
 - ・同期フレームの受信が連続 3 回失敗 (A. E63 検出) した場合(しばらく同期フレームが抜けると、タイミングがずれるため)
 - ・同期異常を検出した場合(→A. E61 検出時)

6. ソフトウェア構成

6.1. プロジェクト構成

| | |
|---|---|
|  Mercury_Project_vF*** | Mercury project top directory |
|  src | Servo control sources |
|  Bsp | Board Support Packages (start up, drivers, vectors, etc...) |
|  Std_Sources | Products common directory (Standard servo control sources) |
|  BaseControls | ScanA / ScanB control modules |
|  BaseSequences | ScanC control modules |
|  HAL | Hardware Access Library (Hardware Abstraction Layer) |
|  R4FIP | JL-086 (SoC) ARM Cortex R4F IP Core access library |
|  ServoIP | JL-086 (SoC) Servo IP Core and ASIP access library |
|  M3IP | JL-086 (SoC) M-III IP Core access library |
|  MicroProgram | JL-086 (SoC) ASIP control sources |
|  Communications | Interface Modules to external devices |
|  AP | Analog / Pulse reference I/F modules |
|  M2 | MECHATROLINK-II communication I/F modules |
|  M3 | MECHATROLINK-III communication I/F modules |
|  DPM | Dualport memory communication I/F modules |
|  MemoBus | Memobus communication modules (M-II/M-III message, Operator, PC, etc...) |
|  Systems | System control modules |
|  Config | System configuration modules (Initialize, Global data definitions, etc) |
|  DataTrace | Data trace execution modules |
|  Registers | Variety of internal register definitions and access modules |
|  UtilityFuns | Utility Functions execution modules |
|  YE_Products | Product specific directory (Product specific source, execute files, Parameter |
|  NormalSpec | Normal specific products directory (Σ -7S/ Σ -7W/ Σ -7C/ Σ -7MD) |
|  Outputs | Output files for standard specific (execute files, Parameter files) |
|  SGD7S | Output files for SGD7S |
|  Bin | Execute files for Each I/F type |
|  Prm | Parameter files for Each I/F type |
|  SGD7W | Output files for SGD7W |
| . . . | |
|  Sources | Standard specific sources |
|  DataTrace | Data trace calculation modules |
|  Registers | Variety of register data tables |
|  Y-Spec | Y-specific products directory (Σ -7S/ Σ -7W/ Σ -7C/ Σ -7MD) |
|  Y100 | Y100 specific directory |
|  Outputs | Output files for Y100 specific (execute files, Parameter files) |
|  Sources | Y100 specific sources |
|  FT-Spec | FT-specific products directory (Σ -7S/ Σ -7W/ Σ -7C/ Σ -7MD) |
|  FT001 | FT001 specific directory |
| . . . | |
|  DS-5_Project | Project directory for DS-5 |
|  Workspace | Work space with respective products (Main project) |
|  NormalSpec | Project for Mercury Normal specific products (Σ -7S/ Σ -7W/ Σ -7C/ Σ -7MD) |
|  SGD7S_AP | Sub project for SGD7S analog / pulse I/F type |
|  SGD7S_M2 | Sub project for SGD7S M-II I/F type |
|  SGD7S_M3 | Sub project for SGD7S M-III I/F type |
|  SGD7S_CM | Sub project for SGD7S Command option I/F type |
|  SGD7S_CoE | Sub project for SGD7S EtherCAT I/F type |
|  SGD7W_M3 | Sub project for SGD7W M-III I/F type |
|  Y-Spec | Project for Mercury Y-specific products (Σ -7S/ Σ -7W/ Σ -7C/ Σ -7MD) |
|  Y100 | Project for Y100 specification |
|  SGD7S_AP | Sub project for SGD7S-Y100 analog / pulse I/F type |
|  SGD7S_M2 | Sub project for SGD7S-Y100 M-II I/F type |
|  FT-Spec | Project for Mercury FT-specific products (Σ -7S/ Σ -7W/ Σ -7C/ Σ -7MD) |
|  FT001 | Project for FT001 specification |
| . . . | |

6.2. ソフトウェアモジュール構成



| レイヤ | 概要 | 補足 | |
|-----------------------------------|----------------------|--|---|
| Common Lib. | 基本共通ライブラリ | 全てのモジュールが共通で使用するライブラリ サーボ演算ライブラリ、タイマ等のAPIをラッピングした I/Fライブラリから構成される。 | <div style="display: flex; align-items: center;"> <div style="flex: 1; border-left: 1px solid black; border-right: 1px solid black; position: relative;"> <div style="position: absolute; top: 0; right: 0; width: 10px; height: 10px; background: white; border: 1px solid black;"></div> <div style="position: absolute; bottom: 0; right: 0; width: 10px; height: 10px; background: white; border: 1px solid black;"></div> </div> <div style="flex: 1; text-align: center;"> <div style="margin-bottom: 10px;">↑</div> <div style="margin-top: 10px;">↓</div> </div> </div> |
| Servo Hardware Abstraction Layer | ハードウェアアクセスドライバレイヤ | H/Wアクセスを抽象化したレイヤ H/Wが変更された場合は、全てこのレイヤ内で対応する。 | |
| Servo Control Layer | サーボ制御コアレイヤ | サーボ制御のメイン処理を行うレイヤ | |
| Network Control Layer | 通信制御レイヤ | M3, USB, UART 処理を行うブロック | |
| System Application Layer | サーボ製品固有処理レイヤ | 各サーボ製品固有のシステム処理を行うレイヤ。 | |
| API Library for User Applications | ユーザアプリケーション用APIライブラリ | ユーザアプリケーションとのI/Fライブラリ 将来用 | 上位 |

制約：下位層から上位層に対するアクセス（関数呼び出し、変数アクセス）は、絶対に行わないこと

6.2.1. Common Library

| パッケージ | 略称 | 概要 | 備考 |
|--------------------|------|------------------------|----|
| Servo Math Library | Mlib | サーボ演算ライブラリ | |
| Timer Library | Klib | タイマライブラリ ウェイト、時刻測定用 | |

6.2.2. Servo H/W Abstraction Layer

| パッケージ | 略称 | 概要 | 備考 |
|------------------------|---------|---|------------------------------------|
| Servo UDL Abstractions | SHALdrv | サーボ ASIC 操作を抽象化した、サーボ制御用の HW ドライバ シリアルエンコーダ送受信、電流検出、チャージポンプ、I/O 入出力、 分周出力、パルス指令入力、指令 AD、DB 制御、アナログモニタ、 H/W 異常検出、JL-086 サーボ IP セットアップ用の IF を備える | I/O 入出力、アナログモニタはユニット共通 それ以外は軸依存 |
| EEPROM driver | EEPdrv | EEPROM にアクセスするためのドライバ I2C I/F に対応 | ユニット共通 |
| Power Control driver | SHALdrv | パワー部 H/W アクセス用のドライバ 主回路 AC 状態、DC 電圧読み込み、回生 Tr の ON/OFF 等 | ユニット共通 |
| Temperature Detect | SHALdrv | 制御基板、パワー基板に搭載された温度センサ用のアクセスドライバ | ユニット共通 |
| SW, LED | SHALdrv | RSW、DSW の読み込み処理、LED 制御用のドライバ | ユニット共通 |
| HWBB | SHALdrv | HWBB 状態検出、HWBB 異常検出 (SIL3 対応) 用のドライバ | ユニット共通 |
| Communication Driver | COMdrv | USB、UART、 オプションユニットアクセス用ドライバ | ユニット共通 |

6.2.3. Servo Control Layer

| パッケージ | 略称 | 概要 | 備考 |
|--|------|--|------------------|
| Filters and Current Loop (μ -Program) | | uP Cyclic (Prm) Interface μ プログラム (フィルタ処理、電流ループ) とのデータ交換 IF 処理ブロック | |
| Control Loop | | 位置/速度ループ・トルク指令フィルタ処理 | |
| Additional Controls | | ScanA、ScanB 処理で実行される制御ループ以外のサーボ制御処理 制御機能毎にパッケージングする | |
| Servo Cyclic Control I/F | | A/P、MECHATROLINK、指令オプション用 FG 処理部との I/F ブロック SGDV の DPM 仕様とほぼ同等の I/F とする | |
| Sequence Controls | | 電源関連制御処理 ベースブロック、DB 停止、OT 停止、BK 出力等のドライブ状態制御 OL、OS 等のドライブ状態監視処理 | 電源関連制御処理は、ユニット共通 |
| Base Prams | Bprm | Servo Control Layer の各モジュールが共通で使用する基本パラメータデータ | |
| Alarm Manager | ALM | アラーム、ワーニング状態管理、アラーム履歴管理を行うモジュール | |

6.2.4. System Application & Network Control Layer

| パッケージ | 概要 | 備考 |
|--------------------------------|---|------------------|
| M3 cyclic I/F M2 cyclic I/F | M3/M2 通信処理部 コマンド解析、レスポンス作成、モーション生成を実施する | |
| A/P cyclic I/F | A/P 指令取得、フィルタリング、モーション生成を実施する | |
| Message I/F | UART、USB、M3/M2 メッセージ処理を行うブロック | ユニット共通 |
| Servo Regiseters | レジスタアクセス処理を行うブロック (Pn/Un/Fn 等のレジスタ操作を集約) | 特定レジスタのみユニット共通 |
| Utility Functions | Fn 機能の実行を行うブロック | 特定 Fn 機能のみユニット共通 |
| Data Trace | データトレースを実施する | ユニット共通 |
| System Configurations | システムのコンフィギュレーション処理を行うブロック 電源投入時のイニシャルシーケンスや、パラメータ再計算、システ | パラメータ再計算以外、ユニ |

| | | |
|---------------------|--|--------|
| | ムの 同期合わせを実施する | ット共通 |
| System Tasks | メインタスク処理 メインタスクから各パッケージのランタイム処理を呼び出して実行する | ユニット共通 |
| Digital Op. Control | デジタルオペレータ用画面生成、キー入力状態取得 | ユニット共通 |
| Panel Op. Control | パネルオペレータ用画面生成、キー入力状態取得 | ユニット共通 |

6.3. サーボ制御ブロック図

6.4. ソフトウェアアーキテクチャ

Σ-7 シリーズのソフトウェアアーキテクチャでは、多軸対応および機能のモジュール化を行う。それぞれの概念を以下に示す。

6.4.1. 多軸対応

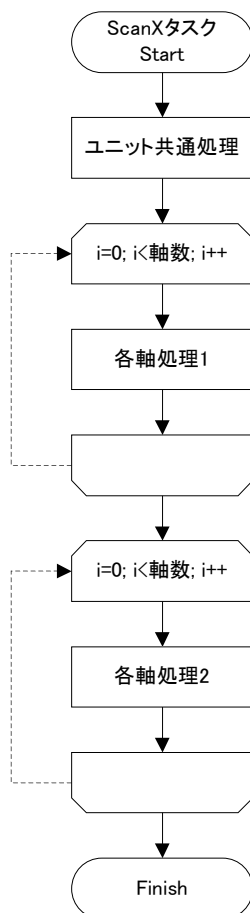
多軸制御に対応するためには、処理の多軸化とデータの多軸化が必要となる。それぞれの対応方法を以下に示す。

1) 処理の多軸化

処理の多軸化のための方策として、以下 2 点を実施する。

- ユニット共通処理の選択と集中
- 各軸処理の「for 文」化

“ユニット共通処理の選択と集中”とは、電源関連処理などのユニット共通処理を選別し、各割り込みタスクの先頭に配置することである。また、“各軸処理の「for 文」化”とは、1 軸ごとに必要となる処理のコールを for ループで繰り返し処理させることである。以下に多軸化後のタスク処理フローを示す。



ユニット共通処理の選択と集中

ユニット共通処理はタスク先頭に集中させる。

※forループに含めると「0軸目の場合」といった分岐が必要となり、現在処理中の軸を意識したソフトウェアとなるため、ふさわしくない。

各軸処理の「for文」化

各軸処理は、Input、Main、Outputといった単位で切り分けを行い、それぞれをforループで回すような作りとする。

2) データの多軸化

単 CPU による多軸サーボソフトウェアのみをターゲットとした場合、多軸処理方式は軸変数の管理方式により決まる。以下に Σ -V-SD 開発時に検討を行った軸変数の管理方式の一覧を抜粋する。

| No | 方式 | 内 容 | 採用製品/CPU |
|----|---------------------------|---|---|
| 1 | ベースレジスタ方式 (単 CPU 多軸制御) | CPU のベースレジスタ(グローバルポインタ gp)を軸毎に切り替える 【メリット】 <ul style="list-style-type: none"> ・ 単軸と多軸ソフトウェアの共通化が最大 (1 軸分の変数定義で多軸化が可能) ・ 多軸ソフトウェアのオーバーヘッドが最少 【デメリット】 <ul style="list-style-type: none"> ・ 定数テーブル変数ポインタの補正処理が必要 ・ CPU 固有機能を使用する。 ・ 変数定義が 1 軸分しかなく、多軸処理としての可読性は低い。 | Σ -V-SD V850 ME3 (Renesas) |
| 2 | ベースポインタ方式 (単 CPU 多軸制御) | ベースポインタを軸毎に切り替える 【メリット】 <ul style="list-style-type: none"> ・ 多軸ソフトウェア用に変数の定義変更が必要(変数は配列として定義) ・ 単軸と多軸ソフトウェアの共通化が可能 (単軸ソフトウェアは制御軸数を"1"とすることで実現可能) ・ CPU 依存の機能を使用しないため、汎用性が高い 【デメリット】 <ul style="list-style-type: none"> ・ 多軸ソフトウェアではポインタアクセスオーバーヘッド有り ・ 定数テーブル変数ポインタの補正処理が必要 | Σ -V-MD PQ3 (PowerPC) |
| 3 | 仮想記憶方式 (単 CPU 多軸制御) | CPU の仮想記憶機能を使用して、実メモリエリアを軸毎に切り替える 【メリット】 <ul style="list-style-type: none"> ・ 単軸と多軸ソフトウェアの共通化が最大 ・ 多軸ソフトウェアのオーバーヘッドが最少 ・ 定数テーブル変数ポインタの補正処理は不要 【デメリット】 <ul style="list-style-type: none"> ・ CPU 固有機能を使用する。 | - |
| 4 | マルチコア方式 (軸制御/Core) | 軸毎に CPU コアを割り当てる 【メリット】 <ul style="list-style-type: none"> ・ 単軸と同じソフトウェアが使用できる ・ 定数テーブル変数ポインタの補正処理は不要 【デメリット】 <ul style="list-style-type: none"> ・ 外部バス競合によるオーバーヘッド有り(微小) ・ マルチコア CPU 以外選択できない | - |

参照元：【900-066-604】 multiaxisdrive_multidrivessoftdeignseet

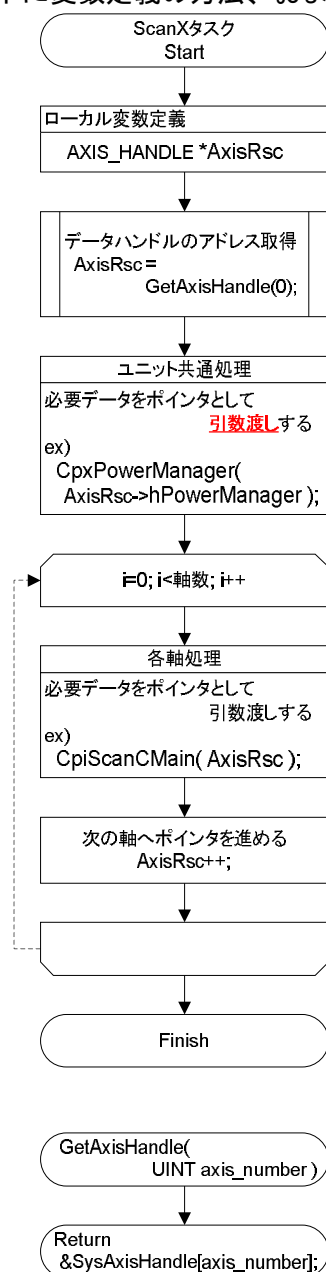
上の表から分かる通り、ベースポインタ方式以外は CPU 固有機能を使用する。 Σ -V 世代までは CPU は基本的に V850 系列を使用しており、ある意味 CPU 性能を最大限に活かすようなソフトウェアアーキテクチャを構築してきた。

しかしながら、最近では Σ -V-MD をはじめ、 Σ -S、富士機械殿向け SGDZ-BS63 など V850 系列ではない CPU を搭載した製品が増加傾向にあり、CPU 固有機能を使用するようなアーキテクチャは敬遠される状況である。特に標準製品である Σ -7 シリーズは、各種カスタマイズ製品のベースとなる製品であるため、なおさらである。

また、1CPU 多軸においては、軸間協調を行いやすいといったメリットもある。この点を考慮すると、変数データの軸切り替えは出来る限り容易に行えることが理想である。

以上を踏まえ、 Σ -7 シリーズにおいては、汎用性と可読性を重視し、ベースポインタ方式を採用する。

以下に変数定義の方法、および多軸処理時のフローを示す。



グローバル変数定義

各軸の制御を司るデータハンドルを軸数分配列として定義する。
 AXIS_HANDLE 各軸制御用データハンドル
 MAX_AXIS_NUMBER 制御軸数

AXIS_HANDLEの各要素はモジュール処理用データへのポインタで構成される。軸ごとにそれぞれの軸データのアドレスが格納される(実態は軸数分存在する)。

ユニット共通変数は、同一のアドレスが設定される(実態はユニットで1つ)。

AXIS_HANDLE SysAxisHandle[MAX_AXIS_NUMBER];

```

typedef struct AXIS_HANDLE {
    LONG AxisNo; /* 軸番号(=MB基板スロット番号) */
    LONG AxisAddress; /* 軸アドレス(論理軸番号) */

    /* ベースサーボ制御用データ(ScanA) */
    BASE_LOOPCTRLS *BaseLoops; /* 制御ループ演算用データ */
    CTRL_LOOP_OUT *CtrlLoopOut; /* ScanA出力データ */

    /* ベースサーボ制御用データ(ScanB) */
    BASE_CTRL_HNDL *BaseCtrlData; /* 基本制御用共有データ */
    SERVO_CONTROL_IF *SvCtrlIf; /* M3 I/Fデータ */
    BASE_CTRL_OUT *BaseCtrlOut; /* ScanB出力データ */
    DETVIB *DetVib; /* 振動検出用変数 */
    USHORT *TableRefBuffer; /* テーブル運転指令パツファへのポインタ */

    MENCV *MencV; /* モータエンコーダデータへのポインタ */
    MENCV *FencV; /* フルクエンコーダデータ */
    ASICS *SvAsicRegs; /* ASICアドレステーブルへのポインタ */
    SHAL_LTSIGNALS *LatchSignals; /* EXTラッチ信号用 */

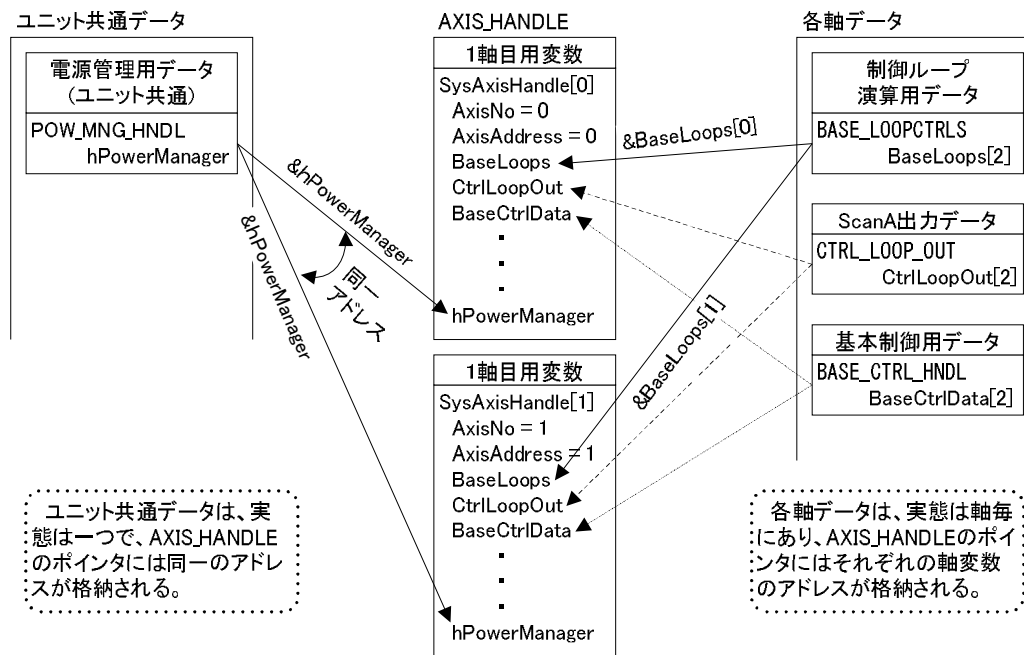
    MOTSPDMAFIL *MotSpdMafil; /* モータ速度検出用移動平均演算用変数 */

    PDET_ABSCONTROL *PdetAbsSeq; /* ABSスケール磁極検出用変数 */
    MPFIND *MpFind; /* 磁極検出動作用変数 */
    PUMP_CTR_STR *pump_if_ptr; /* PUMP ON用データへのポインタ */
    SETTLINGTIME *SettlingTime; /* 整定時間計測関連処理用構造体 */

    /* サーボシーケンス制御用データ(ScanC) */
    SEQ_CTRL_HNDL *SeqCtrlData; /* シーケンス処理用共有データ */
    SEQ_CTRL_OUT *SeqCtrlOut; /* ScanC出力データ */
    BE_SEQ_HNDL *BeSeqData; /* Base Enable Sequence Handle */
    CHECK_ALARM *CheckAlarm; /* アラーム検出処理用構造体 */
    DET_HWBB_STS *DetHwbbSts; /* HWBB状態検出用データ */
    REMVIBFREQ *RemVibFreq; /* 残留振動周波数モニタ用データ */
    POW_MNG_HNDL *hPowerManager; /* 電源管理用データ(ユニット共通) */
    .
    .
    .
} AXIS_HANDLE;
  
```

上記のフローに示す通り、各モジュールで使用するデータは引数渡しとなる。この点が既存のΣ-Vまでのアーキテクチャと大きく異なる点である。

最後に AXIS_HANDLE 構造体のイメージ図を以下に示す。



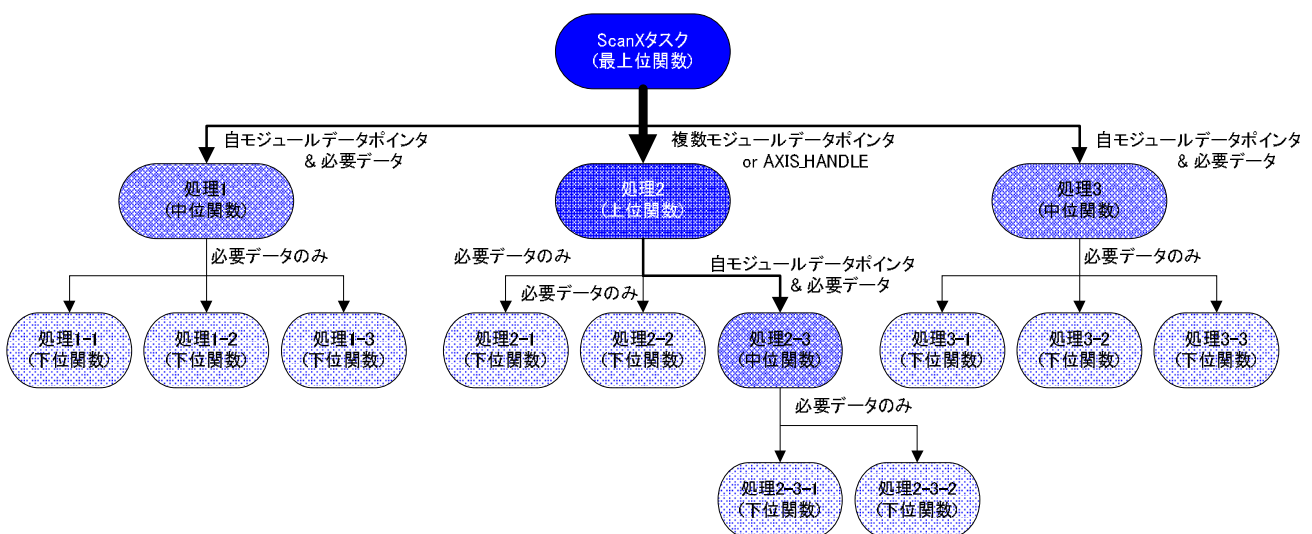
6.4.2. 機能のモジュール化

ソフトウェアの移植性を向上するため、各機能のモジュール化をさらに進める。Σ-V のソフトウェアにおいて、各機能の分割についてはかなり実施されている。ただし、各モジュールデータへのアクセスやスキャン間グローバル変数アクセス、引数・戻り値なし関数の多用など、完全とは言い難い。そこで、Σ-7 シリーズのアーキテクチャにおいては、機能のモジュール化をもう一段進め、より移植性の高い機能モジュールを作り込む。

機能のモジュール化を進めるに当たり、以下 2 点を実施する。

- 処理とデータを階層化する。
- モジュール間結合度を弱くする。

まず、処理とデータの階層化について、概念図を以下に示す。



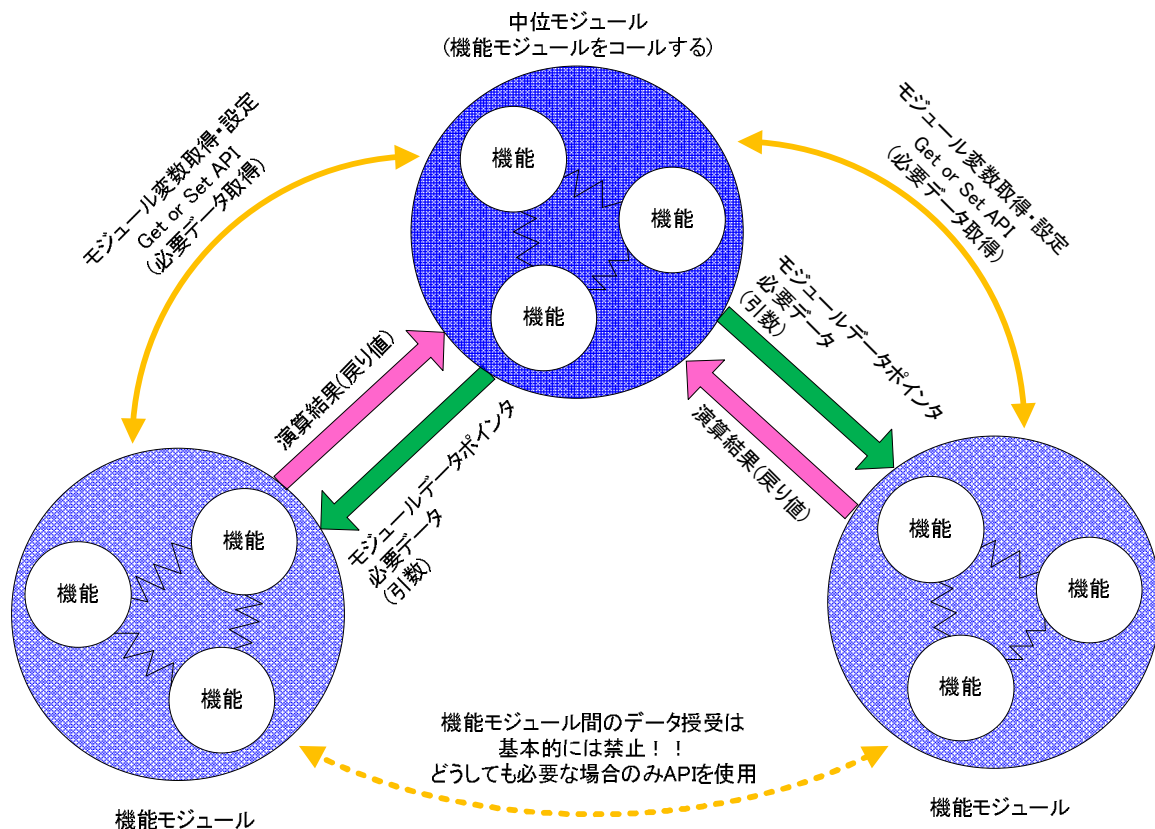
上図に示す通り、それぞれの関数レベルに階層を設け、階層に応じてアクセス可能な変数を制限する。Σ-V では関数の階層化までは行われていたものの、下位の関数までもが自由にグローバル変数の読み書きを行っていた。Σ-7 ではほぼ全ての関数に引数を設け、使用するデータは呼び出し側から受け取る仕組みとする。処理を行うに当たって必要となる別モジュールの情報などは、同様に引数データとして渡すものとする。これにより、関数内でアクセス可能なデータは必要最小限となり、またどの情報を用いるのかが分かりやすくなるため、可読性も向上する。当然下位の関数になればなるほど必要となる情報は少なくてすむはずであるため、関数の階層同様データも階層化される。

次に、モジュール間結合度について説明する。結合度とは、モジュール同士がどれだけ独立に、または関連して作られているかの度合を示す尺度であり、結合度が低い(弱い)ほど望ましいとされている。ただし、完全に独立したモジュールのみでソフトウェアを構成することはほぼ不可能であるため、その指針を設け、可能な限り結合度の低いソフトウェアの構築を目指す。

具体的には、以下4点をルール化し、これに基づいて設計を行う。

- 各機能モジュールは、引数として自モジュール変数のポインタと必要となる他モジュールの情報を受け取り、演算結果を戻り値として返す。
- 必要となるデータが多く、引数が増大する場合には、その処理が複雑である可能性があるため、分割を検討する。それでも引数が6以上となる場合には、必要なデータを集めた構造体を定義し、それを引数として渡す。
- モジュール変数へのアクセス (Get/Set) は、API を用いて行うものとする。API は各モジュールにてインライン関数として準備する。
- モジュール変数へのアクセスは、基本的には上位モジュールもしくは中位モジュールにて行い、機能モジュール同士でのデータのやり取りは禁止する。ただし、処理の構成上どうしても必要な場合に限り、API を使用可能とする。

以下に、モジュール間の関係性を図示する。



6.5. マイクロプログラム

JL-086 サーボ IP のマイクロプログラムは、C 言語による開発が可能である。

サーボ IP マイクロプログラム側の変数(レジスタ)アドレスと CortexR4F ソフトウェア側の変数アドレスを一致させる必要がある。ソースコードにて定義を行うとアドレスをずらす場合など手間が大きくなるため、サーボ IP マイクロプログラム用、CortexR4F ソフトウェア用それぞれの変数定義ヘッダーファイルを作成するエクセルマクロを準備し、レジスタ定義は全てエクセルにて管理する。

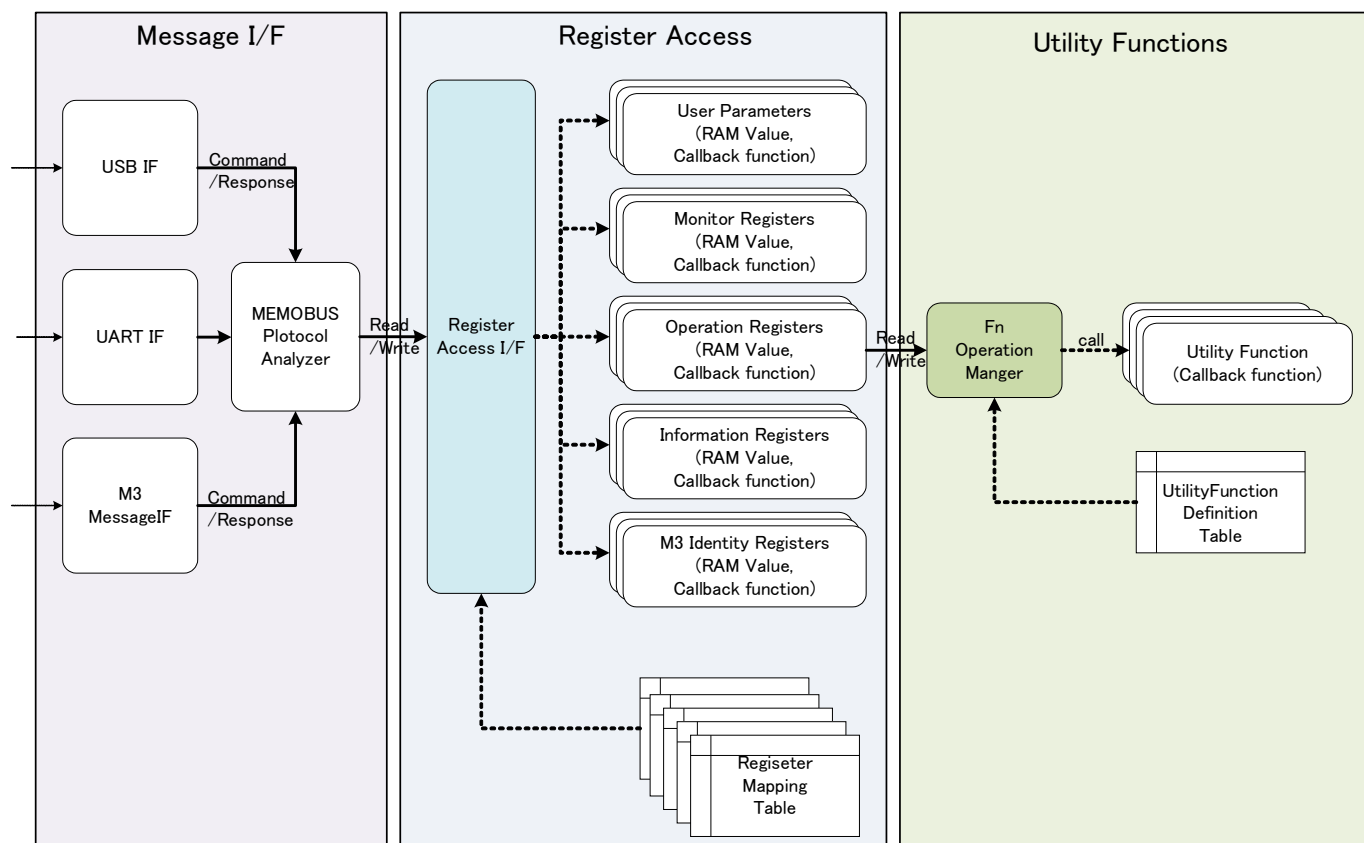
CortexR4F ソフトウェア同様、マイクロプログラムも多軸化に対応する。方針は 6.3.1 節に記載したものと同様である。

7. サーボレジスタ管理

7.1. レジスタアクセスモデル

ユーザパラメータを含む、サーボのレジスタアクセスモデルを下記に示す。

- (1) メッセージ I/F からのサーボレジスタへのアクセスは、“Register Access I/F” を介して実施する。
- (2) “Register Access I/F” は、あらかじめ登録された“Register Mapping Table”を参照し、テーブルに登録されたレジスタデータの格納先や、パラメータアクセス時のコールバック処理関数等に従って、レジスタのリード、ライト処理が実施される。



7.2. レジスタ登録

レジスタへの登録は、下記手順により実施する。

| 手順 | 概要 | 詳細 |
|----|------------------------------|---|
| 1 | “Register Mapping Table” の作成 | Excel を使用して、レジスタを登録し、テーブルソースを作成する。 |
| 2 | 登録した Register 処理の実装 | 以下を実装する。 ・ レジスタデータの格納先の実装 ・ レジスタリード・ライト時のコールバック処理関数 |
| 3 | ソースのビルド | |

8. ファームウェアダウンロード

ソフトウェアの暗号化

書き込み失敗、書き込み中電源断対策

SigmaWin+対応

T. B. D.

9. データベース管理

Σ-7 シリーズにおいては、パラメータやモニタ、トレース、アラームといったデータベース情報を、ソフトウェア、エンジニアリングツール、マニュアルにて共有する仕組みを構築する。具体的には、共通のマスターデータベースを1つ準備し、そこからそれぞれに必要な情報をピックアップしてスレーブデータベースを生成するといった仕組みを検討中。

10. エラー検出

T. B. D.

10.1. サーボアラーム

T. B. D.

10.2. サーボワーニング

T. B. D.

10.3. CPU 異常検出

T. B. D.

11. 新規開発・ネック技術一覧

| No | 分類 | 詳細項目 | 技術課題、新規開発内容 | 対応内容 |
|----|------------------|-------------------|---|--|
| 1 | フルク構成 | フルク接続 I/F | エンコーダ接続コネクタが FBA 搭載分しかない。 | ②空きスロットをフルクとして使用し、(何処をフルクとするかはパラメータ) フルク時は、 <u>位置ループを CPU 側 (ScanB) で処理し、マイクロ側で速度ループを実施する。</u> ※この形態の場合は、完全フルクは未サポートとする。 |
| 2 | Ethernet | MAC アドレス, IP アドレス | MAC アドレスの管理 | MP シリーズと同様に、製品シリアル番号と合わせて管理。製品出荷時に試験機から、製品シリアル番号等とともに、EEPROM に MAC アドレス、IP アドレスを書き込む。(詳細は、出荷試験仕様書に記載する) |
| 3 | | 割り込み | 割り込みが多発した場合に、サーボ制御処理への影響 | SigmaWin でトレースしながら、処理時間を評価し、問題となるようなら、Ethernet ドライバを変更 (チューニング) する。 → ScanB への影響はみられないため、デフォルトのまま使用する。 |
| 5 | CPU (PowerPC) | 処理性能 | ScanB: 125us で何軸まわせるか | JL-102 や JL-077 への転送を DMA 化し、CPU 処理負荷を減らす。また、ScanB で使用する命令コード、RAM を常に L2 キャッシュ上にロックするようにする。 |
| 6 | | 同期割り込みジッタ | RTOS の割り込みレイテンシの問題で、定周期でたたく JL-077 のレジスタ同期にぶれが生じないか | INTEGRITY の割り込みジッタを評価。ジッタが大きいようであれば (2us 以上)、FPGA 上に同期機能を実装するようにする。 → 同期ジッタ < 2us のため OK. |
| 7 | JL-077 μ I/F | ラッチ機能 | JL-077 のラッチ機能を使用 | JL-077 のラッチタイマ機能を使用して外部信号ラッチを実装 |
| 8 | | ScanA 処理 | SCAN A 処理のマイクロ化 | |
| 9 | ユーザアプリ | I/F | ユーザアプリ環境の構築 | 順次検討 |
| 10 | | 開発環境 | | |
| 11 | | アプリロード方法 | | |
| 12 | | アプリデータモニタ | | |
| 13 | | アプリ処理異常検出 | | |
| 14 | 保護機能 | 温度センサによる OH 検出 | | 検討中 |
| 17 | | コンバータ過負荷検出 | | 検討中 |
| 18 | PCIe 対応 | パラメータ | IFMP (現状の SVC) とサーボとの共通のパラメータの決定 | 製品仕様書 (PCIe 版) 記載の通り。 |
| 19 | | サーボ処理部との I/F | サーボ処理部との I/F の決定 | 現状の M3 とサーボ部の I/F 仕様とする。(別途内部 I/F 仕様書を作成) |
| 20 | SigmaWin+ | 多軸トレース | トレース I/F の拡張 | レジスタ仕様書で定義 |
| 21 | | モータ-エンコーダ誤配線診断 | | 検討中 |
| 22 | ファームウェアダウンロード | Ethernet/M3 | FTP, HTTP プロトコルを使用 | MP-Platform |

12. 開発環境

12.1. ソフトウェアビルド環境

| 項目 | ツール名 |
|----------|-------------------------|
| コンパイラ | ARM DS-5 (Version 5.xx) |
| デバッガ | microVIEW-PLUS |
| JTAG ICE | advice LUNA |

以下、コンパイラ設定を示す。(デフォルト設定から変更する項目のみ記載)

| 項目 | 設定 | 内容 |
|-----------------|--------------------------|---|
| 使用ライブラリ | socket net fpgaapi | Ethernet 通信用ソケットライブラリ (INTEGRITY 標準) Ethernet 通信用プロトコルスタック (INTEGRITY 標準) IF 基板同期用アクセスライブラリ (MP-Platform) |
| 最適化レベル | Optimize for General Use | Enables all the optimizations which improve both performance and size, or which improve performance without dramatically increasing size. |
| 構造体 最小アライメント | 4byte | Controls the minimal alignment of all objects of types struct, class, and union. |
| Char タイプの符号 | Signed | Specifies the signedness of the char type |
| ポインタタイプの符号 | Unsigned | Specifies the signedness of the pointer type |

12.2. クロスコンパイル環境

Σ-7 シリーズの開発においては、以下を目的としてクロスコンパイル環境を構築する。

- 開発環境への依存度を低くする。
- より厳密にコンパイルエラーをチェックする。
- 単体試験の効率を向上する。
- 実機レスでの開発環境を準備する。

具体的には、ARM DS-5 の他に、Microsoft Visual Studio 2010 による開発環境を構築する。基本の処理は共通とし、C 言語化したマイクロプログラムも組み込む。割り込み処理とシミュレータ機能のみ専用処理として準備することで、簡易シミュレータとして単体試験にも対応可能とする。

12.3. マイクロプログラムビルド環境

| 項目 | ツール名 |
|-------|---------------|
| コンパイラ | IP Programmer |
| デバッガ | IP Debugger |

T. B. D.

12.4. プロジェクトフォルダ構成

T. B. D.

12.5. コーディングルール

INGRAM では、下表のコーディングルールに従い、コーディングを実施する。

| | |
|----|--|
| R1 | コーディングスタイル <ul style="list-style-type: none"> ・ANSI スタイルを採用する ・字下げには 4文字タブ を使用する。半角スペースは使用しない。 ・全角スペースは不可。 ・コメントは /* ○○○ */ の形とする。 ・半角カタカナは使用不可。 ・goto、continue を使用しない。 ・コメント内容は基本的に SGD V を踏襲し、必要に応じて変更する。 ・処理が 1 行だけの場合でも、if 文など必ず中括弧を使用(ブロック化)する。 Ex) if(x == 0) { y = 0; } |
| R2 | ヘッダファイル <ul style="list-style-type: none"> ・ヘッダファイルの先頭と最後に再読み込み防止用の定義を記述する。 ⇒ "#ifndef", "#define", "#endif" を入れる。 ・ヘッダファイルの記述内容は以下の通りとする。 <ol style="list-style-type: none"> (1) ファイルヘッダコメント (2) インクルード (3) 関数のプロトタイプ宣言 (4) #define 定義 (5) 構造体定義 (6) 外部変数宣言 (←const 型の定数、又は、グローバル変数を定義している一部のファイルのみ定義すること。通常は、外部変数宣言はしないこと。) (7) 関数プロトタイプ宣言 (←モジュール間 I/F 関数の定義) |
| R3 | ソースファイル <ul style="list-style-type: none"> ・ソースファイルの記述内容は以下の通りとする。 <ol style="list-style-type: none"> (1) ファイルヘッダコメント (2) インクルード (3) static 変数、 (4) static 関数宣言 (5) 関数定義 ※自ファイル専用の#define 定義は自ファイル専用ヘッダファイルに記述 ・他モジュールの関数を使用する場合には、プロトタイプ宣言のあるヘッダファイルをインクルードして使用する。 |
| R4 | モジュール構成 <ul style="list-style-type: none"> ・機能モジュール構成は基本設計書の構成に従う。 ・機能モジュール(=クラス)毎に関数(メソッド)、構造体(クラスメンバ)を定義する。 ・各機能モジュール内の I/F 関数の構成は、下記の例を基本とする。 <ol style="list-style-type: none"> (1) 電源投入時初期化処理関数 ・ ・ ・ xxxxInitialize() (2) パラメータ再計算時初期化処理関数 ・ ・ ・ xxxxCalculateInitPrmYYYYY() (3) パラメータ(レジスタ)ライト時、オンライン計算処理関数 ・ ・ ・ xxxxCalculatePrmYYYYY() (4) 定周期ランタイム処理関数 ・ ・ ・ xxxxRuntimeService() (5) モジュール毎の I/F 関数 ・ ・ ・ xxxxXxxxYyyyyy() ※あてはまらないモジュールもあるので、設計段階で I/F を必ず検討すること。 |
| R5 | ネーミング <ul style="list-style-type: none"> ・関数名 クラス名 - 動詞 - 補語 Ex) PcmDetectDcVolatage, PcmInitialize, PcmCalculate*** ↑ Power Control Manager ・構造体タグ名 クラス名 - _ - **** Ex) PCM_POWER_MANAGER ・変数名 <ul style="list-style-type: none"> ・異なる役割で同一の変数名を使用しない。 ・引数やローカル変数として構造体メンバを指定する場合には、同一の名称とする。 ・できる限り SGD V を踏襲する(あまりにもおかしいものは変更する)。 |
| R6 | 型宣言 <ul style="list-style-type: none"> ・char, short, long など基本の型を使用しない。 ・CPU 変更時等の移植性を考慮し、Basedef.h で定義した CHAR/SHORT/LONG/UCCHAR/USHORT/ULONG や、INT8/INT16/INT32/UINT8/UINT16/UINT32 等を使用する。 |
| R7 | エンディアン |

| | |
|-----|---|
| | <ul style="list-style-type: none"> ・ ビッグエンディアン、リトルエンディアンどちらにも対応可能とする。 ・ 必要に応じて、<code>#ifdef __BIG_ENDIAN__</code> , <code>__LITTLE_ENDIAN__</code> を使用して処理を分ける。 |
| R8 | <p>演算の優先順位, 比較対象の明確化</p> <ul style="list-style-type: none"> ・ <u>if 文内で演算や代入はしない。</u> Ex) <code>if(++x < y)</code> ⇒ <code>x++;</code> <code>if(x < y)</code> Ex) <code>if((x=yyyy()) < y)</code> ⇒ <code>x=yyyy();</code> <code>if(x < y)</code> ・ 比較対象を必ず明記する。 Ex) <code>if(x) ⇒ if(x != FALSE)</code> ↑ 厳禁 |
| R9 | <p>定数</p> <ul style="list-style-type: none"> ・ 定数を使用する場合、型を明確にする。 Ex) <code>#define MAXSIZE 0x8000UL /* unsigned long */</code> <code>#define MAXSIZE 0x8000L /* signed long */</code> <code>#define MAXSIZE 0x1000U /* unsigned */</code> |
| R10 | <p>マクロの使用</p> <ul style="list-style-type: none"> ・ マジックナンバー使用禁止。マクロ定義すること。 ・ <u>マクロ関数は使用禁止。</u>特に複数行に渡るマクロ関数は厳禁。 ・ 必要な場合には、マクロ関数ではなくインライン関数とする。 ・ <u>関連する定数定義は define ではなく、enum を使用し定義する。</u> |
| R11 | <p>変数宣言</p> <ul style="list-style-type: none"> ・ 宣言文で変数初期化を行わない。 Ex) <code>LONG i = 0;</code> ⇒ <code>LONG i;</code> <code>i = 0;</code> ・ グローバル変数は1つのファイルにまとめて定義する。 ・ 各ファイルに定義するのは、static 変数のみ。 |
| R12 | <p>関数</p> <ul style="list-style-type: none"> ・ <u>関数内でのグローバル変数アクセスは禁止</u>とする。関数 IF を検討し、全て引数で渡すように設計する。 ただし、同一ファイル内の static 変数のアクセスは可とする。 ・ 関数の引数の数は出来る限り4つ以内。 ・ 5つ以上となる場合は、構造体でデータを纏めるか、処理を分割できないかなど検討する。 ・ 高速スキャンで呼ばれる関数で、構造体を引数とする場合は、必ずポインタ渡しとする。 |

13. 添付資料

13.1. 割り込み処理概要

| 名 称 | | 優先 順位 | 機 能 説 明 | 割り込みハンドラ処理内容 |
|-------|------------|----------|---|--|
| IRQ0 | PFAIL, WDG | 15 | IF 基板ウォッチドッグタイムアウト (MP_Platform 内で割り込処理が実装される) | システム停止処理 (LED を点滅させ、無限ループ) |
| IRQ1 | M3INT | 14 | M-III 通信周期割り込み (JL-102 の INT1 割り込み) | 割り込みハンドラ処理なし。(FPGA にて HINT3000 割り込みと同期されるため、 HINT3000 割り込みで全て処理する) |
| IRQ2 | unused | – | 未使用 | |
| IRQ3 | MBINT | 11 | MB 基板からの割り込み (未使用) | |
| IRQ4 | OPSYNC2 | 10 | OPSYNC2 同期信号割込 (62.5us) (未使用) | |
| IRQ5 | OPSYNC | 9 | OPSYNC 同期信号割込 (0.5ms) (未使用) | |
| IRQ6 | MPSYNC | 8 | MPSYNC 同期信号割込 (6ms) (未使用) | |
| IRQ7 | HINT3000 | 6 | ScanB 割り込み (同期回路上で M3INT 割り込み信号と同期) | <ul style="list-style-type: none"> ・FPGA 割り込み要因のクリア ・ScanB タスク起床 ・JL-077 全軸分の ESYNC レジスタキック (エンコーダ外部同期の実行) ・M3 トグル切り替え |
| IRQ8 | LINT | 5 | L スキャン割り込み (未使用) | |
| IRQ9 | M3_INT0 | 1 | JL-102 の/INT0 割り込み (将来用、未使用) | |
| IRQ10 | unused | – | | |
| IRQ11 | unused | – | | |

*各割り込みの優先順位は割り込み優先レベル値 (15~0) で表され、レベル 15 が最高で、レベル 1 が最低。

レベル 0 に設定すると、その割り込みはマスクされ、割り込み要求は無視される。

13.2. タスク処理概要

| タスク | 説明 | 処理内容 |
|---------|--------------------|---|
| Initial | サーボシステム制御部イニシャルタスク | 電源投入時イニシャル処理 |
| ScanB | サーボ制御高速スキャンタスク | <ul style="list-style-type: none"> ・M3 サイクリックコマンド処理 ・サーボ制御処理 (詳細は、「13.5サーボ制御処理分担」を参照) ・データトレース処理 |
| ScanC | サーボ制御低速スキャンタスク | <ul style="list-style-type: none"> ・コンバータ電源制御処理 (power control manager) ・ドライブ状態制御処理 (DB, OT, BK 出力, サーボ ON 等) ・ドライブ状態・アラーム監視処理 (OL, OS, OV, UV 等) ・LED 状態表示処理 |
| Round | サーボ制御用 Round タスク | <ul style="list-style-type: none"> ・M3 メッセージ処理 ・Ethernet メッセージ処理 ・M3 サイクリックコマンドレジスタアクセス処理 ・Fn 機能実行処理 |

13.3. ASIC—CPU 間データ転送

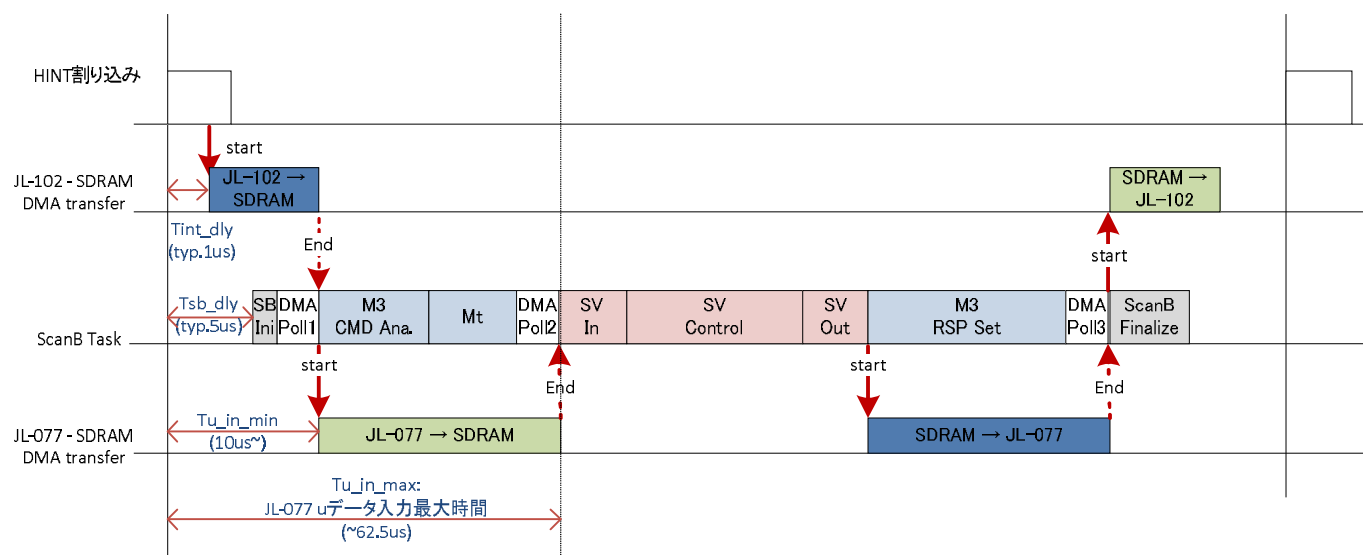
INGRAM では、ScanB タスクから、JL102, 及び JL-077 (マイクロ I/F) に対してメモリのリード・ライトを毎周期行う。このときのメモリ転送にかかる理論上の処理時間は下表のようになる。

[ASIC アクセス時間 : (理論値)]

| | CPU→ASIC (write) | ASIC→CPU (read) | トータルアクセス時間 |
|-------------------------|-------------------------------|-------------------------------|-----------------------|
| JL-077 (125ns/16bit) | 3us/軸 (48byte) 24us/8 軸 | 3us/軸 (48byte) 24us/8 軸 | 6us/軸 48us/8 軸 |
| JL-102 (100ns/32bit) | 1.2us/軸 (48byte) 9.6us/8 軸 | 1.2us/軸 (48byte) 9.6us/8 軸 | 2.4us/軸 19.2us/8 軸 |

※上記の JL-077 アクセスは、マイクロ部への定周期にアクセスするデータ分のみとなり、ScanB で実施している、H/W 制御のためのレジスタアクセスは含まれない。

上記から、ASIC アクセスだけでも 8 軸にすると、トータル約 68us となるため、これらのメモリ転送を DMA にて実施し、CPU の負荷率を下げるように設計する。下図に DMA 転送タイミングを示す。



DMA 転送タイミングにおける制約 :

制約 1: JL-077 からのデータ取り込みは、HINT 割り込み (ScanA) 開始から 10[us] 以降に開始すること。10us 以内に取り込んでしまうと、ScanA でのエンコーダデータ取得が完了せずに、HINT1 周期前のデータとなる。(10us の内訳: INT_AD:8us + ScanA エンコーダデータ取得処理時間 2us)

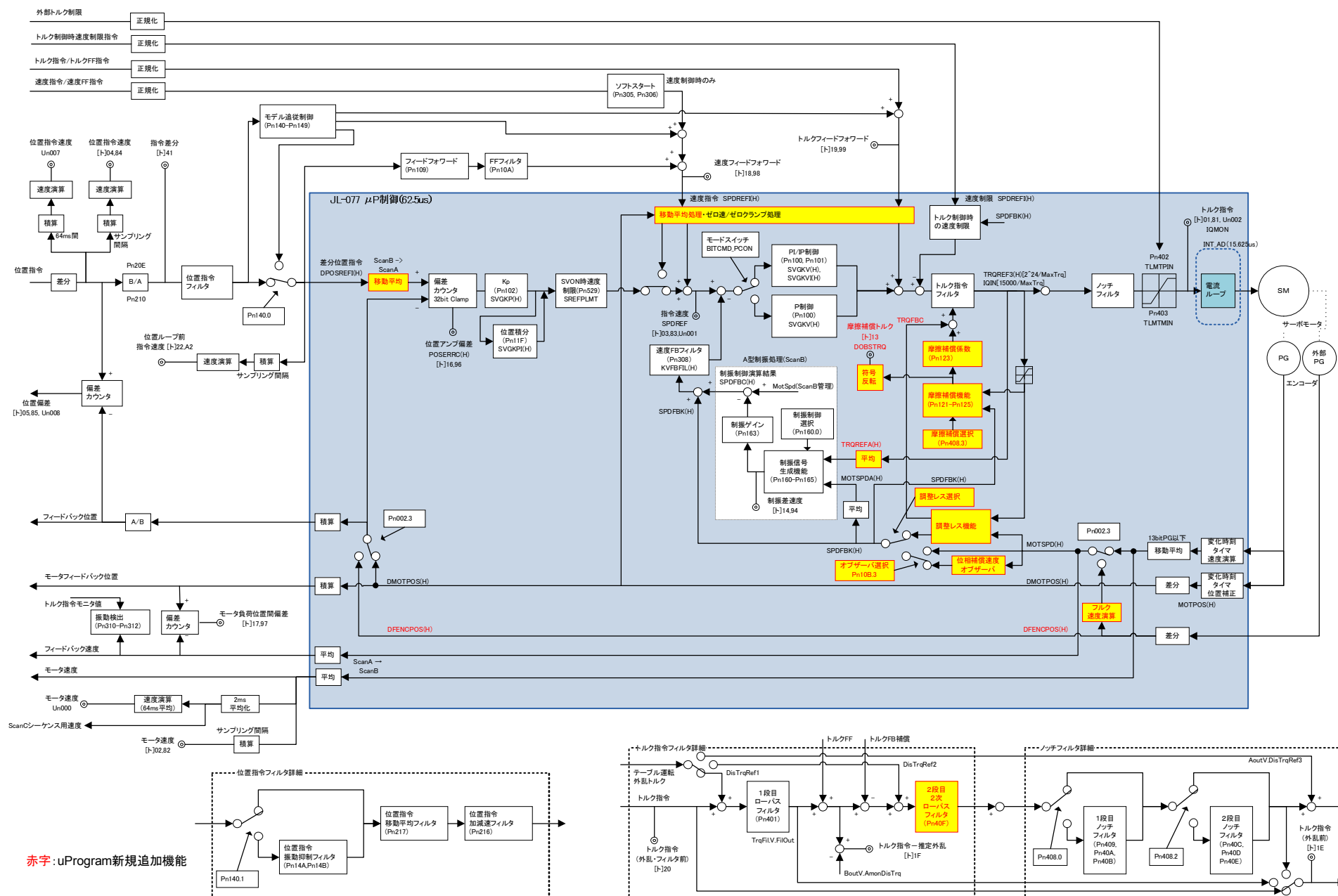
制約 2: DMA 転送の完了待ち (DMA Poll) タイムアウトは、25us 程度とし、それまでに転送が完了しなかった

場合は、アラーム A.BF2 とする。

制約 3: HINT 割り込みが起動したにもかかわらず DMA 転送が完了してない場合は、処理時間オーバとみなし、

アラーム A.BF*, 又は A.E02 を検出する。

13.4. サーボ制御ブロック図



13.5. サーボ制御処理分担

| 分類 | 制御処理 | Σ-V | | IDM/SGDZ-SVN | | INGRAM | |
|-------|--------------------------|------------|--------|--------------|--------|------------|--------|
| | | JL-076/077 | CPU | JL-076/077 | CPU | JL-076/077 | CPU |
| 電流制御 | U相V相電流ゼロ調ゲイン調 | ●INT_AD | | ●INT_AD | | ●INT_AD | |
| | 位相補間処理 | ●INT_AD | | ●INT_AD | | ●INT_AD | |
| | dq変換(UVW→dq変換) | ●INT_AD | | ●INT_AD | | ●INT_AD | |
| | 電流オブザーバ | ●INT_AD | | ●INT_AD | | ●INT_AD | |
| | 弱め界磁用 Id 指令計算処理 | ●INT_AD | | ●INT_AD | | ●INT_AD | |
| | d軸電流 PI 制御 | ●INT_AD | | ●INT_AD | | ●INT_AD | |
| | q軸電流 PI 制御(トルク制限含む) | ●INT_AD | | ●INT_AD | | ●INT_AD | |
| | 電圧 FF 補償(EMF 補償、非干渉制御) | ●INT_AD | | ●INT_AD | | ●INT_AD | |
| | 電圧ベクトル補正 | ●INT_AD | | ●INT_AD | | ●INT_AD | |
| | dq変換(dq→UVW変換) | ●INT_AD | | ●INT_AD | | ●INT_AD | |
| | 過変調補正 | ●INT_AD | | ●INT_AD | | ●INT_AD | |
| | オンディレイ補償 | ●INT_AD | | ●INT_AD | | ●INT_AD | |
| | 電圧指令→PWM カウンタ変換 | ●INT_AD | | ●INT_AD | | ●INT_AD | |
| | 位相補償 | | ●ScanB | ●INT_SCANA | | ●INT_SCANA | |
| トルク制御 | 1 段目ノッチフィルタ | ●INT_HOST | | ●INT_SCANA | | ●INT_SCANA | |
| | モータ共振用ノッチフィルタ | ●INT_HOST | | ●INT_SCANA | | ●INT_SCANA | |
| | 1 次遅れフィルタ(不使用) | ●INT_HOST | | ●INT_SCANA | | ●INT_SCANA | |
| | 2 段目ノッチフィルタ | ●INT_HOST | | ●INT_SCANA | | ●INT_SCANA | |
| | トルク制限 | ●INT_AD | | ●INT_AD | | ●INT_AD | |
| | トルク一次遅れフィルタ(Pn401) | | ●ScanA | ●INT_SCANA | | ●INT_SCANA | |
| | トルク二次遅れフィルタ(Pn40F/Pn410) | | ●ScanA | × | × | ●INT_SCANA | |
| | 外乱オブザーバ | | ●ScanA | × | × | ●INT_SCANA | |
| | トルクフィードフォワード | | ●ScanB | ●INT_SCANA | | ●INT_SCANA | |
| | 速度制限 | | ●ScanB | ●INT_SCANA | | ●INT_SCANA | |
| | リップル補正 | | ●ScanA | × | × | × | |
| 速度制御 | 速度比例制御 | | ●ScanA | ●INT_SCANA | | ●INT_SCANA | |
| | 速度積分制御 | | ●ScanA | ●INT_SCANA | | ●INT_SCANA | |
| | 速度 FB フィルタ | | ●ScanA | ●INT_SCANA | | ●INT_SCANA | |
| | モードスイッチ | | ●ScanA | | ●ScanB | | ●ScanB |
| | 速度指令フィルタ | | ●ScanB | | ●ScanB | | ●ScanB |
| | 速度算出 | | ●ScanA | ●INT_SCANA | | ●INT_SCANA | |
| | ソフトスタート | | ●ScanA | | ●ScanB | | ●ScanB |
| | 停止時振動抑制 | × | × | × | × | × | × |
| | M 型制振制御 | × | × | × | × | × | × |
| | A 型制振制御 | | ●ScanB | × | × | × | ●ScanB |
| | R 型制振制御 | × | × | × | × | × | × |
| | 発振検出速度オブザーバ | | ●ScanB | × | × | × | ●ScanB |
| | 速度オブザーバ | × | × | × | × | × | × |
| | 位相補償速度オブザーバ | | ●ScanA | × | × | ●INT_SCANA | |
| | 速度フィードフォワード | | ●ScanB | | ●ScanB | | ●ScanB |
| | フルクローズ速度算出 | | ●ScanA | × | × | ●INT_SCANA | |
| 位置制御 | ゆれ止め制御 | × | × | × | × | × | × |
| | バックラッシュ補正 | | △ScanB | × | × | × | × |
| | 通信指令補間 | | ●ScanB | | ●ScanB | | ●ScanB |
| | モデル追従制御 | | ●ScanB | | × | | ●ScanB |
| | 位置指令加減速フィルタ | | ●ScanB | × | × | × | ●ScanB |
| | 振動抑制フィルタ | | ●ScanB | | × | | ●ScanB |
| | 位置指令移動平均時間 | | ●ScanB | × | × | × | ●ScanB |
| | 位置偏差クリア | | ●ScanA | ●INT_SCANA | | ●INT_SCANA | |
| | 位置比例制御 | | ●ScanA | ●INT_SCANA | | ●INT_SCANA | |
| | 位置積分制御 | | ●ScanA | ●INT_SCANA | | ●INT_SCANA | |
| | 位置微分制御 | × | × | × | × | × | × |
| | フィードフォワード | | ●ScanB | | ●ScanB | | ●ScanB |
| | 電子ギア | | ●ScanB | | ●ScanB | | ●ScanB |
| | 位置安定化制御 | × | × | × | × | × | × |
| | 残留振動抑制制御 | × | × | × | × | × | × |
| | 速度バイアス | × | × | × | × | × | × |
| | サーボオン時速度制限 | | ●ScanA | ●INT_SCANA | | ●INT_SCANA | |
| | ゼロクランプ | | ●ScanA | ●INT_SCANA | | ●INT_SCANA | |
| その | 原点サーチ | | ●ScanB | | ●ScanB | | ●ScanB |
| | ゲイン切替 | | ●ScanB | | ●ScanB | | ●ScanB |

| 分類 | 制御処理 | Σ-V | | IDM/SGDZ-SVN | | INGRAM | |
|-----------------------|-------------------|------------|--------|--------------|--------|------------|--------|
| | | JL-076/077 | CPU | JL-076/077 | CPU | JL-076/077 | CPU |
| 他 | 位置 FB ラッチ | | ●ScanB | | ●ScanB | | ●ScanB |
| | 制御切替補正 | | ●ScanB | | ●ScanB | | ●ScanB |
| | | | | | | | |
| | INHIBIT | | ●ScanB | × | × | × | × |
| リニア | 磁極検出 | | ●ScanB | | ●ScanB | | ●ScanB |
| | 位相誤検出チェック | | ●ScanA | × | × | | ●ScanB |
| | ホールセンサ異常チェック | | ●ScanA | × | × | | ●ScanB |
| | | | | | | | |
| モータエンコーダ | S-PG 受信 | | ●ScanA | ●INT_SCANA | | ●INT_SCANA | |
| | | | | | | | |
| | S-PG 受信失敗 | | ●ScanA | ●INT_SCANA | | ●INT_SCANA | |
| | S-PG 位置異常チェック | | ●ScanA | ●INT_SCANA | | ●INT_SCANA | |
| | S-PG 受信位置補正 | | ●ScanA | ●INT_SCANA | | ●INT_SCANA | |
| | S-PG 位置生成 | | ●ScanA | ●INT_SCANA | | ●INT_SCANA | |
| | S-PG マルチターンラッチ | | ●ScanA | × | × | × | × |
| | S-PG 初期インクレパルス出力 | | ●ScanA | × | × | × | × |
| | S-PG 原点通過検出 | | ●ScanA | ●INT_SCANA | | ●INT_SCANA | |
| | S-PG 分周出力 | | ●ScanA | × | × | × | × |
| フルクエンコーダ | FS-PG 受信 | | ●ScanA | × | × | × | ●ScanB |
| | | | | | | | |
| | FS-PG 受信失敗 | | ●ScanA | × | × | × | ●ScanB |
| | FS-PG 位置異常 | | ●ScanA | × | × | × | ●ScanB |
| | FS-PG 受信位置補正 | × | × | × | × | × | ●ScanB |
| | FS-PG 位置生成 | | ●ScanA | × | × | × | ●ScanB |
| | FS-PG 絶対値ラッチ | | ●ScanA | × | × | × | ●ScanB |
| | FS-PG 初期インクレパルス出力 | | ●ScanA | × | × | × | ●ScanB |
| | FS-PG 原点通過検出 | | ●ScanA | × | × | × | ●ScanB |
| | FS-PG 分周出力 | | ●ScanA | × | × | × | × |
| マイクロ分周出力 Safety 対応 | S-PG 受信 | ●INT_ENC | | — | — | — | — |
| | S-PG 受信失敗 | ●INT_HOST | | — | — | — | — |
| | S-PG 位置異常チェック | ●INT_HOST | | — | — | — | — |
| | S-PG 受信位置補正 | × | | — | — | — | — |
| | S-PG 位置生成 | ●INT_HOST | | — | — | — | — |
| | S-PG マルチターンラッチ | × | | — | — | — | — |
| | S-PG 初期インクレパルス出力 | × | | — | — | — | — |
| | S-PG 原点通過検出 | × | | — | — | — | — |
| | S-PG 分周出力 | ●INT_HOST | | — | — | — | — |
| | | | | | | | |

INT_SCANA : μプログラム ScanA (62.5us) 割り込み処理

INT_HOST : μプログラム Host CPU 割り込み処理 (CPU 側 ScanA 処理終了で割り込起動)

INT_AD : μプログラム電流検出 AD (15.625us) 割り込み処理

ScanB : CPU 側 ScanB (125us) タスク ※INGRAM はデフォルト 250us

ScanA : CPU 側 ScanA (62.5us) タスク

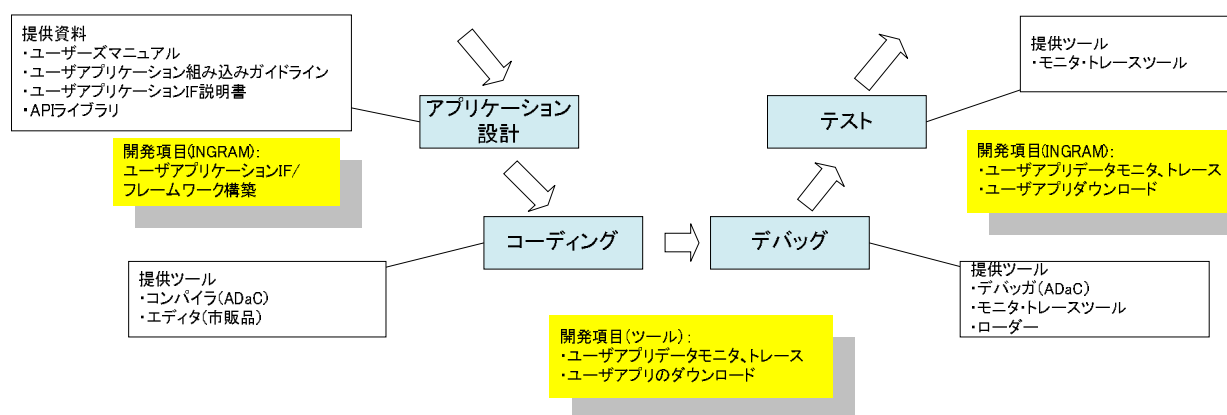
14. ユーザアプリケーション I/F（暫定）

14.1. 目的・用途

下表に、主なユーザアプリケーションの目的、用途を記す。

| No | 目的・用途 | 実現手段 | 実行周期 |
|----|------------------------------|--|------|
| 1 | シーケンス的な動作の実現 | モーション、I/O 操作、パラメータアクセス I/F の構築 | 低速 |
| 2 | 特殊な加減速パターンの実現 | INTERPOLATE を使用してオリジナルモーションを実装 | 高速 |
| 3 | 速度・トルク補正制御 | 速度 FF, トルク FF 用 I/F の準備 | 高速 |
| 4 | 多軸協調動作の実現 (ロボット、ガントリ、etc) | 多軸同時指令が可能な I/F の準備 | 高速 |
| 5 | カスタマイズした制御ループ実現 | システム制御周期 (ScanB) と同じ周期でのアプリケーション実行。(内部指令遅れを最小減にする必要あり) | 高速 |

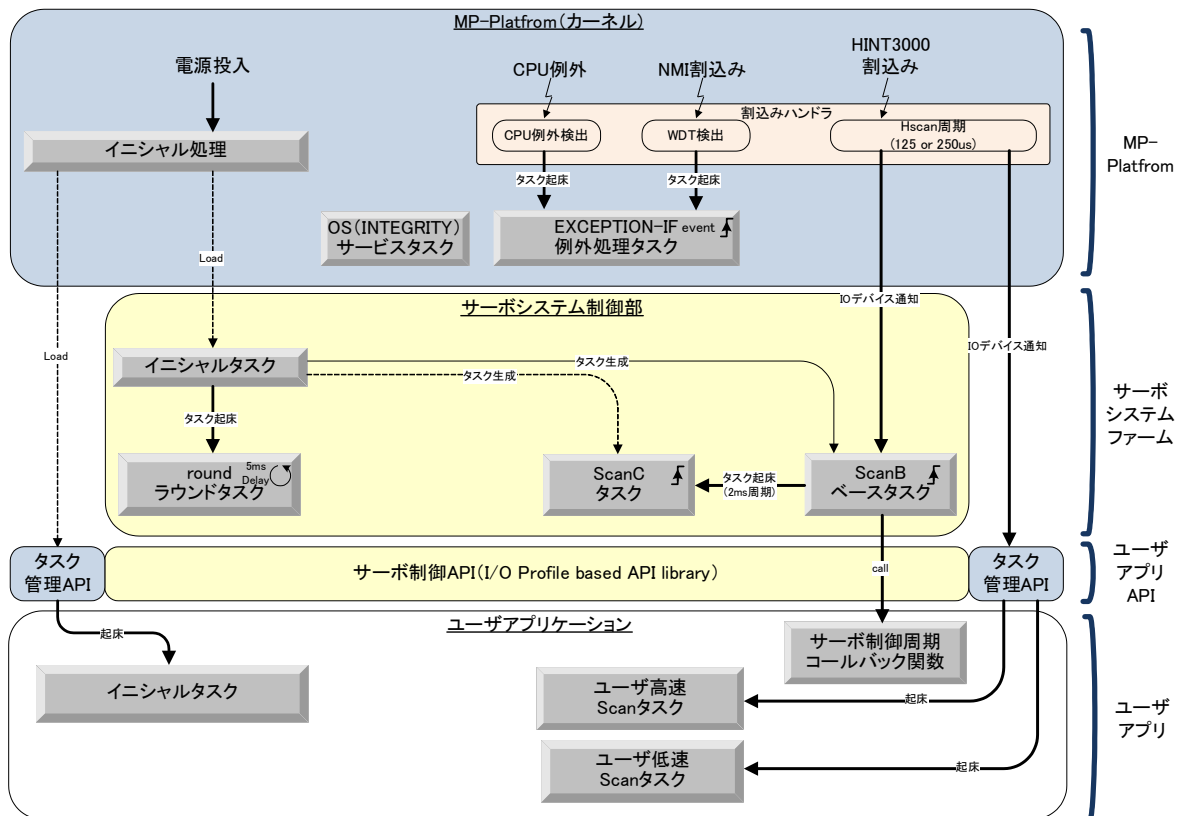
14.2. ユーザへの提供資料・ツール



| 分類 | 項目 | 内容 |
|-------|-----------------------|--------------------------------------|
| 資料 | ユーザーズマニュアル | H/W のセットアップ、及びサーボ標準機能についての説明が記載された資料 |
| | ユーザアプリケーション組み込みガイドライン | ユーザアプリケーション構築時に最初に読む資料 |
| | ユーザアプリケーション IF 説明書 | I/F 機能の詳細が記載された資料 |
| ツール | コンパイラ・デバッガ (ADaC 製) | MP3000 シリーズのアプリケーション開発環境を展開 |
| | モニタ・トレースツール | 未定 |
| | ローダー | 未定 |
| ライブラリ | API ライブラリ (API パッケージ) | ユーザアプリケーション用 API ライブラリ |

14.3. ユーザアプリフレームワーク

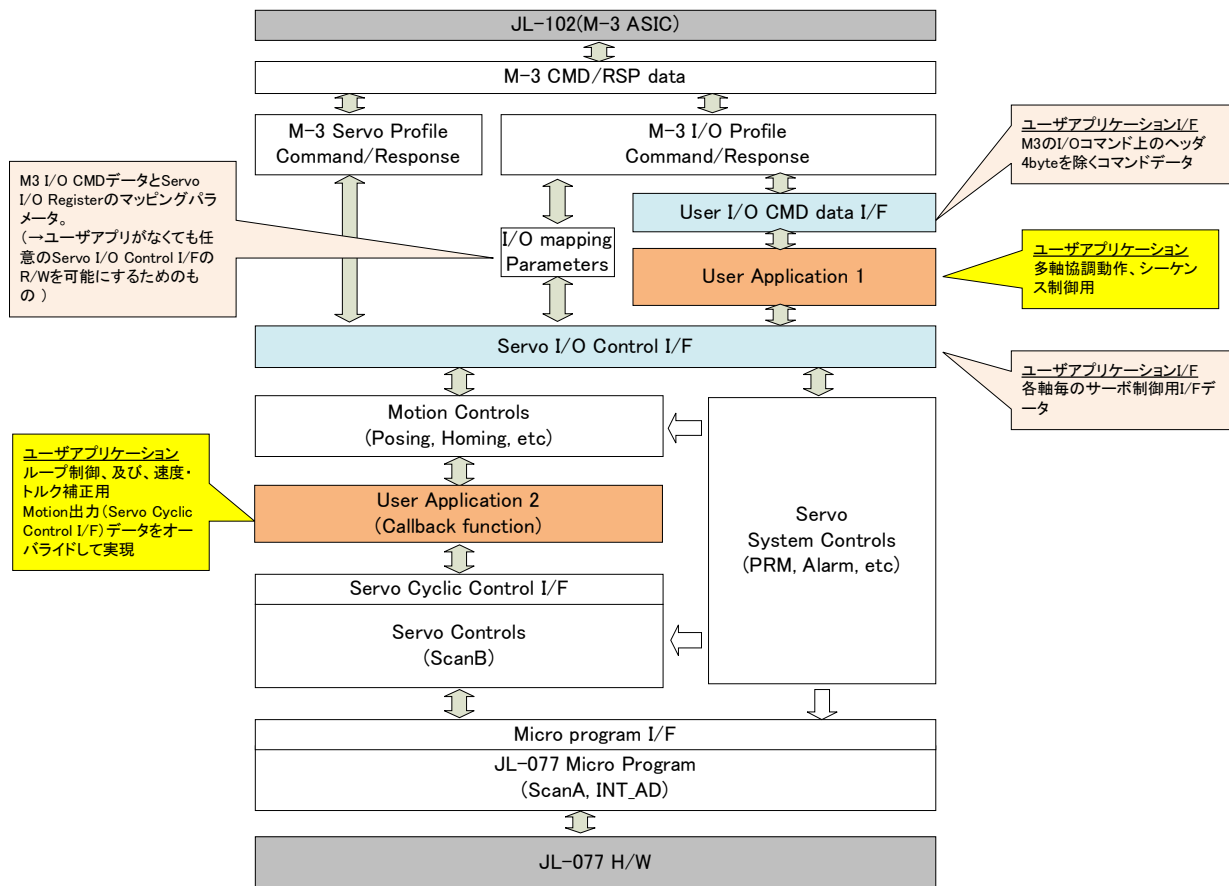
14.3.1. ユーザタスク起動



| 分類 | タスク、I/F | 説明 | 備考 |
|------------------|-----------------|--|---|
| ユーザアプリケーションタスク | イニシャルタスク | M3 プロファイル選択, ユーザアプリケーションタスクの生成, サーボパラメータのコンフィギュレーションを行うタスク | |
| | ユーザ高速スキャンタスク | ScanB に同期したユーザアプリケーションタスク 主に各軸へのモーション指令の行う | ScanB (HINT) の整数倍の周期 0. 25ms ~ 2ms タスクは1個まで |
| | ユーザ低速スキャンタスク | 高速実行が必要のないユーザアプリケーション用のタスク (※ScanB とは非同期) | 4ms ~ 100ms 複数タスク OK |
| | サーボ制御周期コールバック関数 | 内部指令遅れを最小減にした、ScanB から起動されるユーザアプリケーション関数。カスタマイズした制御ループ実現用。 | まずは、社内利用のみ。 |
| アプリケーション I/F API | タスク管理 API | ユーザアプリケーションタスクの生成、起動管理を行うAPI (関数型の API) | |
| | サーボ制御 API | サーボ制御用の API I/O レジスタ (Virtual DPM) 方式の I/F | |

14.3.2. フレームワーク概略

以下に、ユーザアプリケーション実行時の概略データフローを示す。



ユーザアプリケーション用のコントローラから指令データ、及び応答データは、以下のユーザアプリデータ入出力用コマンドを使用して、データのアクセスを行う形とする。

| | |
|-----------------------|---|
| ユーザアプリデータ 入出力用コマンド | I/O プロファイル時 : ・ データ READ/WRITE_A (非同期型) コマンド ・ データ READ/WRITE_S (同期型) コマンド 標準サーボプロファイル時 : ・ ベンダースペシフィック I/O データ READ/WRITE コマンド (コマンドコード : D0h 新規追加) |
|-----------------------|---|

※CPU 処理時間 (最低限 ScanB のみ)、CPU オーバロードアラーム検出機能が必要。

検討課題 :

1) サーボプロファイル時に、ユーザアプリを実行できる仕組み？

例えば、コントローラからは POSING コマンドを投げるが、加減速を少し変えたいだけのときはどうするのか？

→元々(システムファーム上)の POSING 処理を無効とし、ユーザアプリが実行できるようにするか？

→サーボプロファイルでのコマンド動作が変わるとまずいので、別コマンドで実行できるようにするか？

(上記のユーザアプリデータ入出力コマンドを使って、ユーザアプリ独自のモーションを持つ)

2) サーボプロファイルと I/O プロファイル混在時はどうなる？

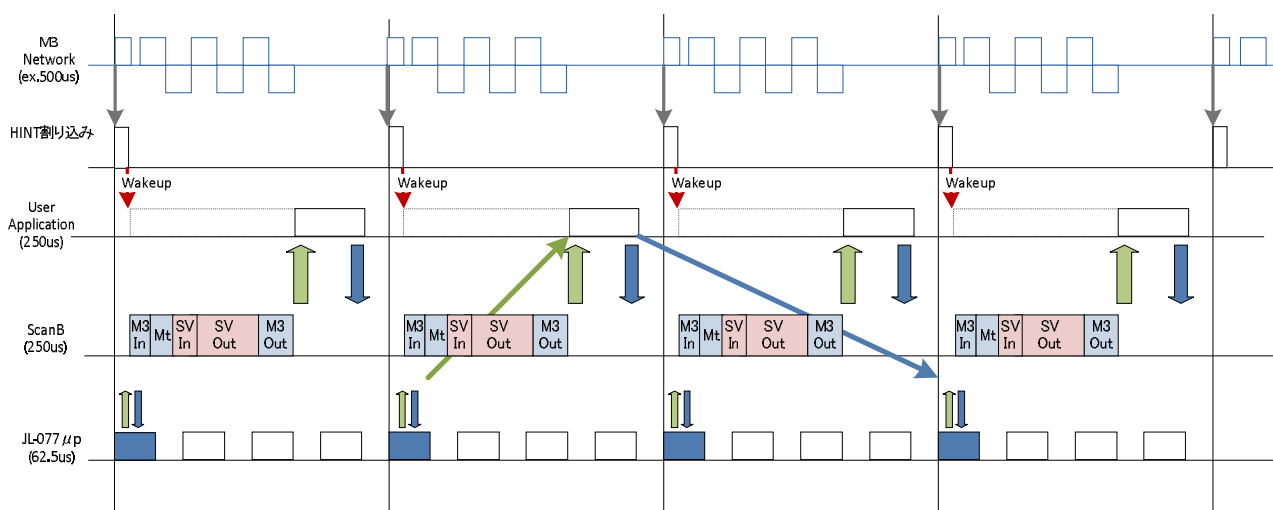
14.3.3. 実行タイミング

ユーザアプリの実行タイミングは、下表の2パターンとなる。(併用可能)

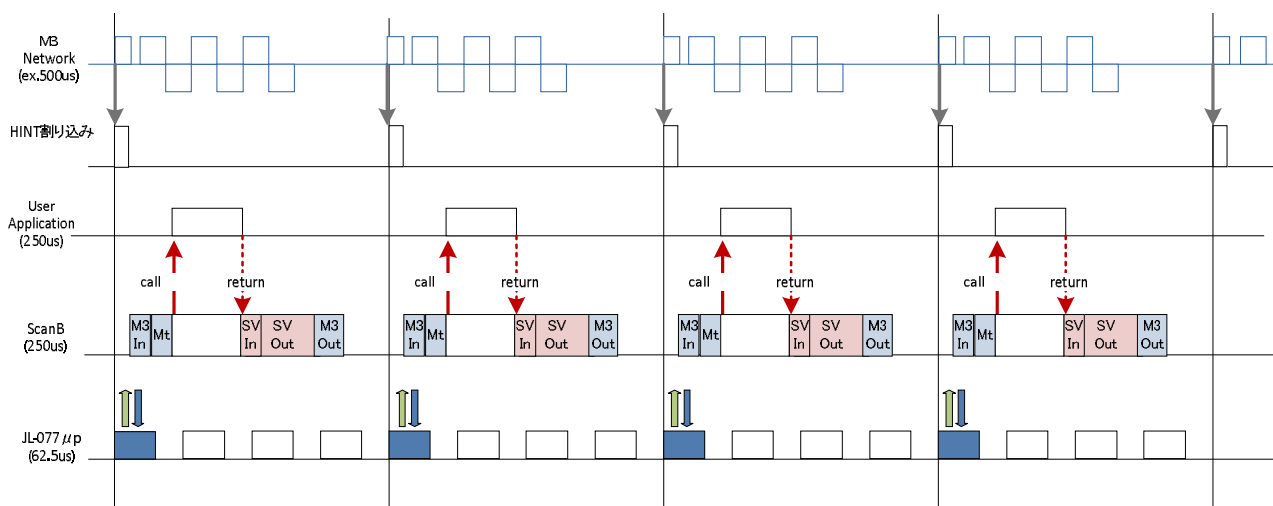
| 種別 | アプリケーション起動周期 | 実行アドレス空間 | 用途等 |
|--------------------|----------------------------------|-------------------|----------------------|
| User Application 1 | ScanB 周期の整数倍のタスク | ユーザアプリ専用アドレス空間 | 通常のモーション制御をする場合 |
| User Application 2 | ScanB 周期 (ScanB タスクからのコールバック) | サーボシステム制御部アドレス空間？ | アプリケーションにてループ制御をする場合 |

下図に、それぞれの場合のタイミングチャートを示す。

(1) User Application 1: アプリケーション周期 = $n \times \text{ScanB 周期}$ ($n=1, 2, 3, \dots$)



(2) User Application 2: ScanB タスクからの コールバック型



14.4. ユーザアプリケーション IF

ユーザアプリケーション用の I/F として、下表の機能を実行できる I/F を準備する。

| 項目 | 機能 | 備考 |
|---------------------|--|------------------------|
| タスク 操作 | イニシャルタスク登録 | |
| | ユーザ高速スキャンタスク登録 | |
| | ユーザ低速スキャンタスク生成 | |
| | サーボ制御周期コールバック関数登録 | |
| 位置制御 | 補間送り指令 | |
| | 位置決め指令 | |
| | 定速送り指令 | |
| | 外部信号位置決め指令 | |
| | 外部信号位置決め付き定速送り指令 | |
| | 原点復帰指令 | |
| | S 字加減速位置決め指令 | |
| 速度制御 | 速度制御モーション指令 | |
| トルク 制御 | TRQCTRL モーション指令 | |
| サーボ 状態制御 | センサオン・オフ | |
| | サーボオン・オフ | |
| | ラッチ操作 | |
| 状態 モニタ | ドライブ状態（サーボオン、主回路、センサオン、HWBB、OT、ソフトリミット等） | |
| | FB 位置、FB 速度、トルク指令、指令位置、ラッチ位置等 | |
| I/O 信号 操作 | I/O 信号 (FBA) リード | |
| | I/O 信号 (FBA) ライト | |
| コンフィ ギュレー ション | M3 プロファイル選択 | イニシャル、低速タスクからのみ 実行可 |
| | M3 通信軸数設定 | イニシャル、低速タスクからのみ 実行可 |
| | サーボレジスタリード | イニシャル、低速タスクからのみ 実行可 |
| | サーボレジスタライト | イニシャル、低速タスクからのみ 実行可 |
| | ユーザアプリ I/O レジスタ登録 | イニシャル、低速タスクからのみ 実行可 |
| | ユーザアプリ I/O レジスタデータ保存 | イニシャル、低速タスクからのみ 実行可 |