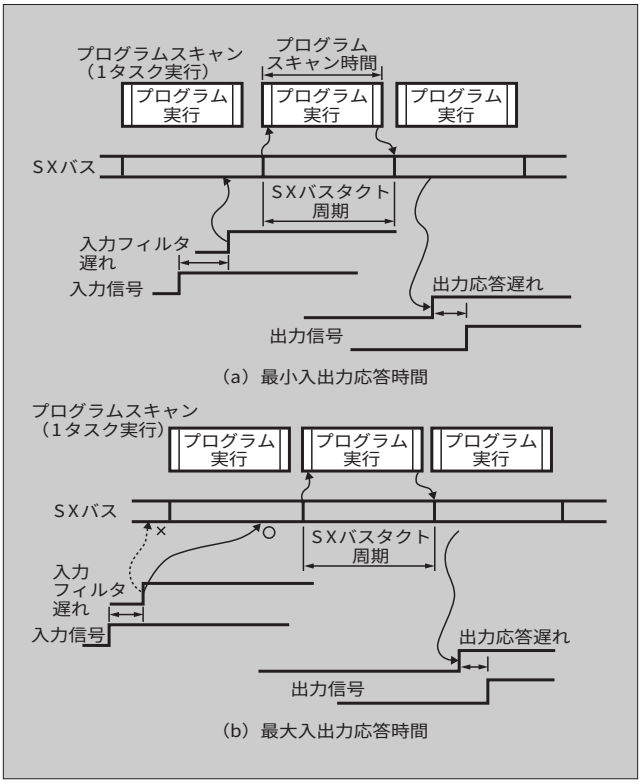
統合コントローラ「MICREX-SX シリーズ」の適用技術

菱沼　正勝（ひしぬま　まさかつ）　　島田　喜秋（しまだ　よしあき）

1　まえがき

統合コントローラ「MICREX-SX シリーズ」の核を構成するスケーラブルマルチコントローラ SPH（以下，SPH と略す）は，高速処理，マルチ CPU システム，国際標準言語 IEC61131-3（旧番号体系 IEC1131-3$_{1993}$）の採用など，従来のプログラマブルコントローラ（PLC）と比較して革新的なアーキテクチャを採用している。SPH は従来の PLC にはない新しい概念も多く，これらを駆使することにより，PLC の適用範囲，応用範囲はさらに拡大していくものと考える。本稿では，SPH をベースにしたシステム構築にあたって，参考になる事項，また必要となる新しい概念のうち，高速処理，特に入出力応答時間，マルチ CPU システム，プログラミングについて述べる。

図１　入出力応答時間



2　入出力応答時間

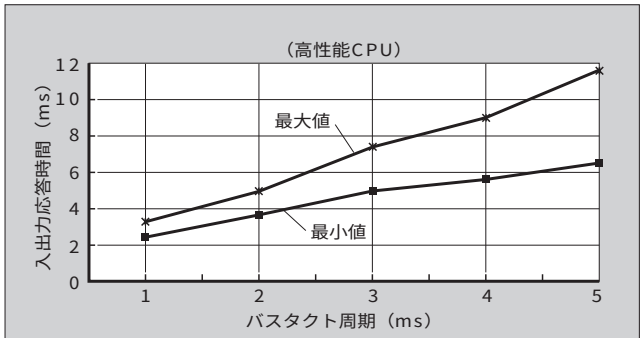
2.1　入出力応答時間を決める要素

高速機械への適用において特に重要な入出力応答時間について説明する。入出力応答時間は入力が入ってプログラムにより処理され，出力するまでの時間で規定される。これは，少し厳密性を欠くが，**図１**のように考えることができる。この図は，タスクで実行されるプログラムが一つで，しかもバスタクト周期内でプログラムがすべて実行されるものと仮定している。実際には，複数のプログラムがデフォルト，定周期，イベントといったタスクに割り付けられて実行されるため，このように単純ではないが，すべてのプログラムが常にバスタクト周期内に実行できれば同様に考えることができる。

バスタクト周期内にプログラムが実行されるものと考えると，入出力応答時間を構成する要素は入力フィルタ遅れ，バスタクト周期，出力応答遅れである。

バスタクト周期の増加に伴う入出力応答時間の実測値を**図２**に示す。本測定は CPU に SPH300 を使用して実施した。また，標準入力モジュールを使用したため，入力フィ

図２　入出力応答時間（実測値）



〈注〉（1）プログラムはバスタクト周期内に実行可能なこと。
（2）入力フィルタの設定は1ms。
（3）出力応答遅れは0.5msと推定される。
（4）諸条件により変わるため本値は参考値とする。

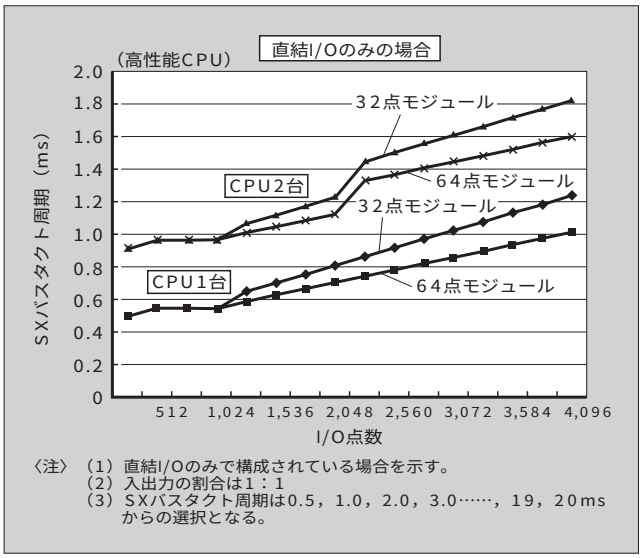
菱沼　正勝

制御機器の開発企画および機械制御システムの設計，エンジニアリング業務に従事。現在，機器事業本部営業統括部商品技術部長。

島田　喜秋

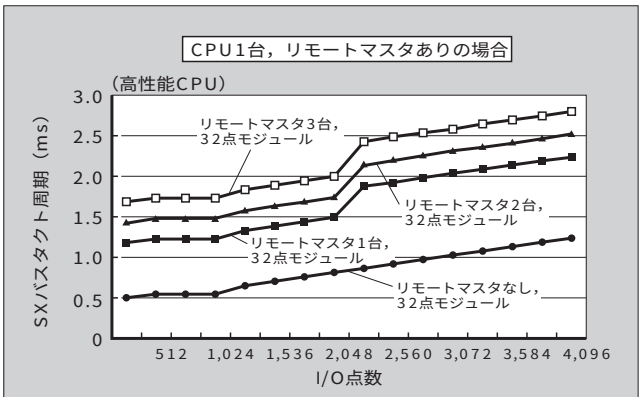
制御機器，可変速駆動機器のシステムエンジニアリング業務に従事。現在，機器事業本部営業統括部商品技術部主査。

図３　バスタクト周期（直結 I/O，計算値）



〈注〉（1）直結I/Oのみで構成されている場合を示す。
（2）入出力の割合は1：1
（3）SXバスタクト周期は0.5，1.0，2.0，3.0……，19，20ms からの選択となる。

図４　バスタクト周期（リモート I/O，計算値）



〈注〉（1）リモートマスタとは，TリンクマスタおよびJPCN1マスタである。
（2）リモートI/Oは，2,048点（128W）/1回線としている。
（3）SXバスタクト周期は0.5，1.0，2.0，3.0……，19，20ms からの選択となる。

ルタは 1 ms で測定しているが，さらに高速な応答を必要とする場合にはノイズなどの影響を考慮しながら，入力フィルタ 1 ms 以下の設定が可能な高速入力モジュールを使用することで対応することができる。出力応答遅れは 0.5 ms 程度と推定される。バスタクト周期が大きくなると，ばらつき（最小値と最大値の差）も大きくなることは注意すべき点である。要求される入出力応答時間を満たすためには，まず，バスタクト周期が**図２**に示す入出力応答時間内に実行され，かつバスタクト周期の時間よりプログラムスキャンが短いことが必要である。そこで以下に主要な検討項目であるバスタクト周期とプログラムスキャンについて説明する。

2.2　バスタクト周期

バスタクト周期とは，SX バス上の I/O リフレッシュ周期である。バスタクト周期とプログラムの実行周期は同期しており，**図１**からも分かるように入力はこのバスタクト周期でプログラムに取り込まれ，演算処理され出力される。バスタクト周期を決める要素としては，I/O 点数，CPU 台数，リモート I/O マスタ台数，通信モジュール台数がある。タクト制御の詳細は省くが，ここではシステムを検討するうえでの大まかな目安を紹介する。SX バスが直結 I/O だけで構成されている場合のバスタクト周期を**図３**に，リモート I/O マスタがある場合のバスタクト周期を**図４**に示す。

2.3　プログラムスキャン

プログラムスキャンとは，プログラムの総実行時間である。詳細はここでは省略するが，以下に注意すべき点を述べる。

一点は，ユーザーファンクションブロック（FB）とユーザーファンクション（FCT）のコール回数が多くなるとスキャンタイムへの影響が無視できない点である。高速制御の場合には特に考慮すべき点である。

二点目はプログラムの実行時間がバスタクト周期の時間を超える場合には，そのプログラムの実行終了後の SX バスの I/O リフレッシュ時にプログラムの実行結果が出力される点である。

三点目は，必要な最終出力が出るまでに数スキャンのプログラム実行を必要とするような場合もあるため，特に FB 化されたプログラムを高速制御に使用する場合には注意する必要がある。

3　マルチ CPU システム

SPH のアーキテクチャの大きな特長の一つであるマルチ CPU システムについて，システムにおけるその必要性と適用方法および注意事項について紹介する。

3.1　マルチ CPU の必要性と適用方法

システム構築の観点からマルチ CPU が必要な場合は高速処理と機能分散の 2 点が考えられる。

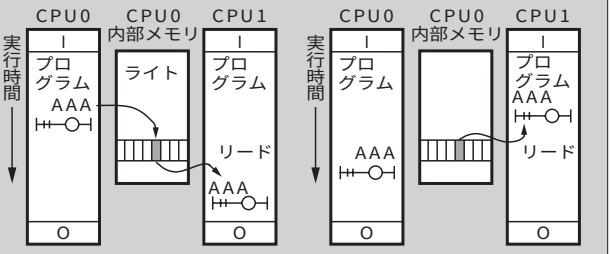
3.1.1　プログラムの大規模化に対する高速処理の実現

プログラム容量が大きくなると，それに比例して当然プログラムの実行速度は遅くなる。大規模化するプログラムに対して，必要な処理速度を維持することは従来の PLC システムにおいては非常に困難であった。また，高速処理を維持する一つの方法として定周期プログラムがあるが，制御の複雑化に伴い，定周期プログラムのプログラム容量が大きくなれば，他のプログラムの実行時間を延ばすことになり，高速処理を維持することに限界が生じる。SX ではプログラムを複数の CPU で並列実行させることにより，プログラムの大規模化と高速処理の相反する課題を解決する手段を提供している。これにより，1 台の CPU でスキャンタイムがバスタクト周期を超える場合でもマルチ CPU 構成により，必要な入出力応答時間を確保することができる。

3.1.2　機能分散による開発の効率向上と保守性の向上

プログラムの複雑・大規模化が進む状況において，例え

図5　CPU 間のメモリリード・ライト



ば 32 k ステップもあるプログラムを 1 CPU のなかで管理することは開発・保守の面でも，限界を超えているといっても過言ではない。実際に，このレベルのプログラム規模を有するユーザーにとって，開発・保守に要する時間，労力が大きな負担になっている。SPH はプログラム言語として国際標準言語である IEC61131-3 を採用することによりプログラムの構造化・階層化設計が可能となっている。これにより，プログラムをリソース単位（CPU），プログラム構成単位（POU），FB 単位といった単位で管理することができる。これを，ハードウェアの面でサポートしているのが複数のリソース（CPU）によるプログラムの実行制御（マルチ CPU システム）である。CPU を複数に分散することにより，プログラムの開発やプログラム変更・機能の追加などに対して容易に対応することが可能となる。例えば，制御システム全体を機能によって，シーケンス制御用 CPU，サーボコントロール用 CPU，通信用 CPU などといった単位で分割することにより，システムの開発・管理および保守が容易になる。

SPH による機能分散の特長は，従来のネットワーク分散と比較して，プロセッサバスを使用することで CPU 間のメモリのやり取りが非常に高速にでき，しかも相手の CPU メモリをダイレクトに読み書きすることができることにある。また，ネットワーク分散のプログラム管理単位が CPU 単位なのに対して，SPH のマルチ CPU システムは全 CPU を一つの PLC システムとして一つのプロジェクト（プログラムの管理単位）で管理できることにある。ネットワーク分散と比較してより密な機能分散といえる。この管理システムにより，高速処理の必要性から仕方なくネットワーク分散を実施していたシステムに対して，より適切なシステムを提供できる。SPH では最高 8 台までのマルチ CPU 構成が可能となっている。

3.1.3　適用における注意事項

マルチ CPU システムにおいては，CPU 間でメモリをリード・ライトするタイミングに注意する必要がある。CPU が他の CPU のメモリをリードするタイミングは**図 5** に示すようにその命令が実行される瞬間である。CPU 間のプログラム実行は個々の命令に対しては非同期で行われているため，**図 5** のようなタイミングのずれが生じる場合がある。高速処理を必要とする制御は同一 CPU にプログラムをまとめることが望ましく，幾つもの CPU を経由して処理をするようなプログラムは作成しないように注意する必要がある。

4　プログラミング

4.1　プログラミングの現状

現状の PLC プログラミングの問題は以下のとおりである。

(1)　プログラムの複雑化による生産性と品質の悪化

PLC の機能・性能の向上に伴い制御内容が複雑化・大規模化しているが，プログラミング技法は依然として変わらないため，プログラムはますます複雑化し，生産性と品質の悪化を招いている。

(2)　一向に進まぬプログラムの再利用

PLC プログラミングの世界では，他人の作成したプログラムを解析することが非常に困難であり，解析しているよりも新たに作成した方が早いことから，結局一から作成している場合が多い。また，再利用の仕組みがあっても，制約が多く一般に利用されているとは言い難い。したがって，似たようなプログラムが無数に存在することになり，生産性は一向に改善されない。

(3)　ドキュメント性の低さによる保守性の低さ

従来の PLC はプログラムの構造化や階層化の機能が低く，アドレス指定やデータ構造の仕組みが貧弱であることから，どうしてもプログラムのドキュメント性は低くなり，システム稼動後の保守に多大な労力を要している。

以上のような問題があるにもかかわらず，PLC はその潜在的な能力により，小規模システムから大規模システムまで広範囲に適用され，多くのシステムが稼動している。

SPH で採用されている IEC 言語は，限界に近づきつつあるソフトウェアの開発環境に対して，革新的なプログラミング環境を提供し，PLC の適応範囲と能力をさらに拡大するものと考える。

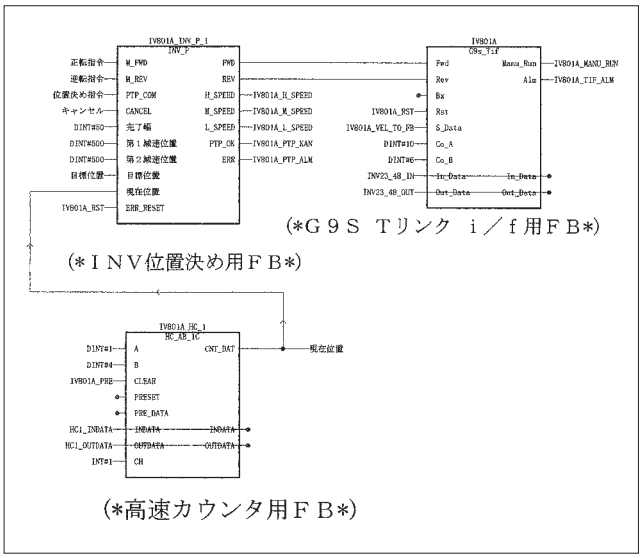
IEC 言語の説明は本特集号の別稿（統合コントローラ「MICREX-SX シリーズ」の統合プログラミング支援システム）で紹介しているので，本稿では特にシステムのソフトウェアを作成する場合の考え方の例をユーザー FB を中心に紹介する。

4.2　プログラムの構造化と FB の適用方法

IEC 規格の最大の特長は，プログラムの構造化，階層化設計に大きな重点を置いていることである。これにより，大規模で複雑なプログラムを作成する場合には最初に大きな機能部分に分解し，次にその機能部分をさらに機能を絞ったブロックに分解表現していくことにより，最終的により小さなプログラムブロックに分解できることである。このプログラムブロックを POU，つまりプログラム，FB，FCT で表現することができる。

このプログラムブロックを，プログラムとするか FB に

図６　インバータによる簡易位置決め FB の一例



するかという点では，システムにより違いがあるが，基本的には最初の大きな機能分解（例えば，手動運転，自動運転，通信処理，異常処理など）はプログラムとしてまとめ，個々のプログラムのなかはできるだけ機能単位の FB としてまとめていく方法がよいと思われる。この場合に，どのような単位で FB を抽出するかが問題となるが，まずは制御対象（例えば，昇降装置，搬送装置，X 軸駆動装置など）と制御目的（例えば，搬入動作，搬出動作，巻取り動作など）の単位で分解していくのがよいと考える。つまり，実際の制御システムのハードウェアと制御目的に合わせて分解していく。この分解作業（構造化）を進めていくと最終的には小さな制御対象と制御目的（機能）の FB になる。**図６**は高速カウンタを利用したインバータによる簡易位置決め FB の一例である。

このとき，各制御対象のインタフェース（FB の INPUT または OUTPUT 端子）を共通化することが再利用性を高める重要なポイントになる。

4.3　FB による標準化と生産性・品質の向上

前節で述べたように，基本的にはハードウェアコンポーネントに合わせてプログラムの FB 化を進めていくこと，そして，FB は小さな機能 FB の組合せで構築していくことがプログラムの標準化，再利用を促進するかぎである。

従来の PLC には IEC 言語のような構造化，再利用の仕組みが存在しなかったり，あっても貧弱であるため，なかなか標準化も進まず，また積極的に標準化を進めている設計部門においても，それはローカルな標準化でしかなかった。しかし，IEC 言語は国際標準規格であり，PLC の世界もグローバルなプログラムの標準化に一歩近づいたといえる。

また，十分に試験検証された FB の組合せによりプログラムを構築していく手法は生産性と品質の向上に大いに貢献するものである。

5　あとがき

スケーラブルマルチコントローラ SPH の適用方法を入出力応答時間，マルチ CPU システム，プログラミングに絞って紹介した。ますます高速化，高機能化するシステムへの PLC の適用に際して，本稿が少しでも参考になれば幸いである。

また，本稿で紹介できなかった特長や機能に関する適用方法についても今後，順次紹介していく所存である。