パーソナルロボット用機能別並列計算機アーキテクチャ：*ASPIRE*

山崎信行† 安西祐一郎†

近年，AIのプラットホームとして自律移動型の知能ロボットに注目が集まっているが，雑踏のような非人工的な環境で自由に動作可能な実用的自律ロボットはいまだに存在しない．そこで本研究では，実用を目指しレスポンシブ性を重視したパーソナルロボット用機能別並列計算機アーキテクチャ *ASPIRE* (*ASynchronous, Parallel, Interrupt-based and REsponsive architecture*) を提案・設計する．*ASPIRE* は，パーソナルロボットにリアクティブ性，リアルタイム性，緊急（例外）処理，ロバスト性，並列性を持たせた機能別並列計算機として設計を行う．*ASPIRE* では，ロボットを機能分散されたモジュールに分割し，各モジュールごとにプロセッサを配置する．各モジュールは，互いに通信しながら自律的に並列動作を行う．また，割り込み線をモジュール内外のイベントを発生するすべてのデバイス（I/O等）に張りめぐらせ，すべてのイベント伝達手段に割り込みを用いる．そして，その機能を用いて通常処理と例外処理を統一的に扱う．*ASPIRE* の実装例として VME bus で結合し分散共有メモリを持ったプロトタイプのパーソナルロボット *ASPIRE-I* を設計・実装し評価を行う．

***ASPIRE*: Function-Classified Parallel Computer Architecture for Personal Robots**

NOBUYUKI YAMASAKI† and YUICHIRO ANZAI†

Nowadays, intelligent autonomous mobile robots are widely used as testbeds for the study of AI. Yet, there is still no autonomous robot in practical use which can operate satisfactorily in non-artificial and unknown environment. In this paper, we propose and design a novel architecture for personal robots, *ASPIRE* (*ASynchronous, Parallel, Interrupt-based and REsponsive architecture*), with emphasis on the responsiveness. *ASPIRE* is functionally classified into self-contained modules, and each module has a processing unit to process information and decide its own behavior. Modules operate in an autonomous fashion, while maintaining communication with other modules. One of the most important characteristics of *ASPIRE* is that all I/O devices and all modules are systematically connected by interrupt lines. Using the interrupt lines, events can be transmitted immediately, when communication is necessary. So *ASPIRE* is good at both immediate and parallel processing. Because of the one-to-one correspondence between an interrupt and an event, *ASPIRE* can reactively handle both normal processing and exceptional processing as a single kind of unified processing. We design and implement the personal robot *ASPIRE-I*, which is combined by VME bus and has two kind of distributed shared memories, as a prototype of personal robots based on *ASPIRE*. The responsiveness of *ASPIRE-I* is examined and evaluated.

1. まえがき

オフィスや家庭における今後の計算機システムのあるべき姿は，従来の計算機システムが扱っていた「電子世界」だけでなく，我々が活動する「物理世界」までもその処理の対象に含めることが重要になると考えられる．そこで我々は，電子世界と物理世界の両方を扱うことができるものとしてパーソナルロボットに注目をしている．現在，我々はパーソナルロボットに電子世界から物理世界への橋渡しをさせこの2つの世界を統合し，従来の計算機科学の研究を物理世界へと拡張することを試みている[1)]．

ここで「パーソナル」という言葉はパーソナルコンピュータの場合と同じ意味であり，パーソナルユースのロボットのことを意味する．現状では小型で汎用の自律移動ロボットを想定している．用途としては，オフィスワーク支援ロボット，家事支援ロボット，アミュー

† 慶應義塾大学大学院理工学研究科計算機科学専攻
Department of Computer Science, Graduate School of Science and Technology, Keio University

ズメントロボット，福祉ロボット等を考えている．

そして近い将来，オフィスにロボットが数台いて，ルーチンワーク等は人間がロボットに仕事を頼むという社会，さらにはもう一歩進んでロボットの方から人間に仕事を依頼し，人間と協調作業を行うというような社会が来ると考えている．このような状況でロボットと人間が自然に接するためには，まずロボットの機敏で正確な動作が不可欠となる．しかしながら，従来のロボットではアーキテクチャ的制約から来る処理能力不足のために自律的に反応性良く動作することが困難であった．したがって，まず研究のプラットホームとなるパーソナルロボットをアーキテクチャレベルから設計・実装する必要がある．同時に，パーソナルユースを目指しているので，手軽に研究開発でき，壊れにくいといった側面も考慮に入れる必要もある．

そこで，本研究ではパーソナルロボット用機能別並列計算機アーキテクチャ*ASPIRE*（*ASynchronous, Parallel, Interrupt-based and REsponsive architecture*）を提案するとともに，そのアーキテクチャに基づいてプロトタイプのパーソナルロボット*ASPIRE-I*を設計・実装し，基本性能の評価を行う．

2. 背　　景

ロボットアーキテクチャとして近年注目されいるものに，サブサンプションアーキテクチャ[2),3)]がある．サブサンプションアーキテクチャのロボットは，行動型アプローチを採り既存の知能ロボットに比べ反応性が良く学習速度が速いという利点がある．しかし，トップダウンに計画・行動することが困難であるため，既存のAI技術が応用しづらく，パーソナルロボットのアーキテクチャには不向きであると考えられる．

サブサンプションアーキテクチャを改良したものとして，SSS[4)]やATLANTIS[5)]がある．これらは，サブサンプションアーキテクチャに既存の制御技術とAI技術を複合したものである．しかし，これらはナビゲーションのアーキテクチャであり，ハードウェアレベルの考慮は行われていない．

センサ入力を重視したマルチプロセッサ用のリアルタイム，マルチタスク，並列プログラミング環境を提供するロボットシステムとしては，CHIMERA II[6)]がある．CHIMERA IIは，アームロボットを対象としているので，システム自体が大規模となり，パーソナルロボットには不向きであると考えられる．

優秀な小型自律移動ロボットとしては山彦[7),8)]がある．移動速度が速く，反応性も既存の移動ロボットと比較するとかなり良い．これは，センサ情報を重視して設計・実装を行っているためだと考えられる．しかしながら，山彦は機能分散されているが，マスタモジュールが他のスレーブモジュールの細かい制御も行うという形態を採っている．また，イベント伝達を高速に行うための機能が備わっていない．したがって，リアクティブ性やリアルタイム性に限界があると考えられる．

我々は，本研究を行うための前段階として小型の自律移動ロボットEinstein Iを4台開発した．Einstein Iは，書類の配送システムを容易・安価に実現することを目標にして設計・実装を行った[9)]．人為的に作られた環境下では，キャスター付きのスチール缶をロボット間で受け渡すことによって，書類をやりとりすることができた．しかし，拡張性の低さ，ロバスト性の低さ，緊急（例外）処理への対応不足など，Einstein Iでは解決できない問題点も同時に明らかになり，実用的なパーソナルロボットを実現するためには，アーキテクチャレベルから設計をやり直す必要がある．

3. *ASPIRE*の提案および設計

3.1 レスポンシブ・システムの提案

パーソナルロボットでは，外部からの入力（センサ，ユーザ入力等）に対して速やかに反応し，その入力に対応する動作を行う必要がある．これらの外部入力（割り込み）は頻繁にかつ連続的に生じ得るので，それらをできるだけ短い反応時間（response time）でかつ短い処理時間（processing time）で正確に処理する必要がある．また，各種デバイス（モータ，センサ等）の制御等は，ある時間制約を満たして実行する必要がある．

そこで我々はパーソナルロボットの動作における最も重要な性質として，「レスポンシブ（responsive）性」というものを導入する．レスポンシブ性とは「リアクティブ（reactive）性とリアルタイム（real-time）性とを兼ね備えた性質」と定義する．そして，パーソナルロボットはレスポンシブ性を有したレスポンシブ・システム（responsive system）として設計されるべきであると提案する．

ここで，リアクティブ性とリアルタイム性は簡単には以下のような性質である．

リアクティブ性　外界からの刺激に対して反射的に動作を行う性質[10),11)]

リアルタイム性　システムの正確さが，（計算あるいは行動等の）結果だけではなく，その結果が導き出された時間にも依存する性質[12)]

レスポンシブ・システムでは，リアルタイム性が要

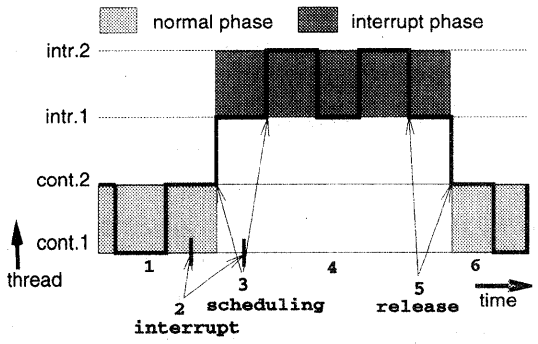


図1 *μ-PULSER* のスケジューリング
Fig. 1 Scheduling of *μ-PULSER*.

求される低レベルなデバイス等の制御や，リアクティブ性が要求される緊急時の処理等を同時に満たすことが求められる．このレスポンシブ性はハードウェアだけでは実現が不可能であり，ソフトウェア（オペレーティングシステム等）の助けを借りて行わなければない．その反対に，ソフトウェアだけでも実現は不可能である．つまり，ハードウェアとソフトウェアの両方を互いに考慮しながら適切に設計することによって初めて実現可能である．我々はすでにマルチスレッドベースのマルチタスク・リアクティブ・リアルタイム・オペレーティングシステム*μ-PULSER*[13]を開発している．*μ-PULSER* では，ハードウェア割り込みとソフトウェア割り込みを DI（Direct Interrupt）という概念で統合し，統一されたイベントとして扱う．また，DI によって複数の割り込み処理の並行実行を可能にする．従来の OS では，ある割り込み処理中に別の割り込みがかかると，現在処理をしている割り込みレベルよりもレベルが高ければ多重割り込みとして処理され，低ければ処理を待たされる．*μ-PULSER* では，これらの複数の割り込みを図 1 のように並行に処理できるため，パーソナルロボットのリアクティブな制御が可能となる．本論文では，*μ-PULSER* と組み合わせてレスポンシブ・システムを構築することが可能となるパーソナルロボット用機能別並列計算機アーキテクチャ *ASPIRE* について述べる．実際に *ASPIRE* と *μ-PULSER* は本研究室で同時期に並列して設計・実装された．

3.2 *ASPIRE* の設計方針

我々は，雑踏のような非人工的な環境でも自律的に正確に動作するパーソナルロボットを実現することを目標とする．そのためには，前節で述べたように，ロボットに「レスポンシブ性」を持たせる必要がある．レスポンシブ性をハードウェア的にサポートする基本的な戦略は，以下のように単純化できる．

- 発生したイベントに対応する反応時間の短縮
- 発生したイベントに対応する処理時間の短縮

これを実現するためには，我々は以下を実現すればよいと考える．

- イベントの即時的伝達
- 異種の複数イベントに対する処理の並列実行
- 同種の複数イベントに対する処理の並行実行

センサ等からイベントが発生すると，それを処理するためのモジュールに即時的にそのイベントを伝達する．そして，そのモジュールでは，そのイベントに対する処理をレスポンシブに実行する．複数のイベントがほとんど同時にあるいは連続的に発生した場合，その処理を並列もしくは並行に実行し，処理時間と反応時間を短縮させることによって，レスポンシブ性を維持する．つまり，異種のイベントに関しては処理するモジュールを機能分散し並列実行させることによって処理時間と反応時間の両方を短縮させ，同種の複数イベントに関してはその処理を並行実行することによって反応時間を短縮させる．これらは，オペレーティングシステム*μ-PULSER* によって制御（スケジューリング）される．並行実行は*μ-PULSER* によりソフトウェア的に実現されるが（3.1 節参照），そのための柔軟な時間管理をサポートする機構を設ける．

雑踏のような実環境では，常に周囲の状況が変化していくので，「緊急（例外）処理」を迅速に行うことができなければならない．このような状況では，緊急処理が頻繁に行われるので，緊急処理をあたかも通常処理のように扱う必要がある．

また，パーソナルロボットには「ロバスト性」も必要である．ここでいうロバスト性とは，ロボットのある一部分のみが故障したとしても，その故障した部分を切り離して動作し続けることのできる性質のことである．

ロボットの各動作は本質的に並列動作しているので（たとえば，センシングしながら移動する等），パーソナルロボットの各モジュールやデバイスを効率良く「並列」に動作させ，前述の性質を持たせるようにする．

3.3 *ASPIRE* の設計

並列性を引き出すために，ロボットを機能別のモジュールに分割し，各モジュールに PU（Processing Unit）を置き，機能別並列計算機として設計する．この機能別並列計算機上に*μ-PULSER* が載り，機能分散されたタスク（スレッド）が各モジュールで並列でかつ並行に実行される．また各モジュールに PU があるので，モジュールごとに単独で自律的に処理するこ

とが可能となる．その結果としてロバスト性も増大する．この機能別モジュールの分割は，モジュールごとに必ずレスポンシブな処理が可能となるように，各モジュールの計算量によって決定する．

各モジュールの個々の制御はそのモジュール内で行うが，モジュール間で相反する行動を始めた場合，どこかで調停を行わなければならない．また，システム全体のプランニングなどを行うことのできるモジュールが必要である．そこで *ASPIRE* では，機能別モジュールの1つとしてメインモジュールを用意する．メインモジュールは，システム全体のプランニングやスケジューリングを行う．また，モジュール間の矛盾する行動を調停し，システム全体のコンシステンシを維持する．したがってシステム全体の制御は，基本的には分散制御であるが，システムとして不整合が生じた場合はメインモジュールがアービタとなるアービタ付き分散制御となる．

モジュール間やモジュール内で何らかのイベントが生じたことを伝達するためには，必ず割り込みを用いる．人間の神経と同様に，割り込み信号線をイベントを起こすすべてのデバイス（I/O 等）に張りめぐらせ，通常処理，例外処理の区別なく，何らかのイベントが発生すると，この割り込み信号線を用いて即時的にイベントを伝達する．これらのイベントを *μ-PULSER* の DI により，モジュールごとに並列に，同じモジュール内で並行に処理することによって即時性と並列性を最大限に活用し，レスポンシブな処理を行う．このように *ASPIRE* では基本的にポーリングが存在しないので，*μ-PULSER* 上ではイベント（DI）が起こるまでは他の複数のスレッドをタイムスライスしながら並列および並行実行することができ，各プロセッサのアイドリングを可能な限り抑えることができる．その結果，各モジュールのスループットが向上し，ロボット全体のレスポンシブ性が向上する．これらの点が，*ASPIRE* の最大の特徴となっている．

すべてのイベントの伝達に割り込みを用いるので，互いに結合しているモジュール間では，双方向に割り込みをかける機構を設ける．

イベントを発生させた後，各モジュール間で何らかの通信をするために，モジュール間にコミュニケーションメモリを設け，通信を非同期に行う．

4. パーソナルロボット *ASPIRE-I* の実装

ASPIRE に従い，プロトタイプのパーソナルロボット *ASPIRE-I* の実装を行う（**図 6** 参照）．*ASPIRE-I* は人間とパーソナルロボットとのインタラクションに関する様々な実験（書類の配送システムや TV 会議システム等）を行うことのできるテストベッドとして実装する．

4.1 モジュール間結合

ASPIRE-I のモジュールの分割は *ASPIRE* に従い，メインモジュール，モータ制御モジュール，センサ制御モジュール等の機能別モジュールに分割する．

これらのモジュール間結合としては，リンク結合，バス結合等が考えられる．*ASPIRE-I* では，以下の理由から汎用バスである VME bus を用いる．

- 小規模並列なのでバスネックになる可能性小
- マルチプロセッサ対応
- 優先順位付多重割り込み対応
- 32 ビットバス
- 国際規格（ISO，IEEE）として厳密に規格化
- VME bus 対応の多種多様な市販ボードを利用可

しかしながら，パーソナルロボット用であることを考慮すると，VME bus には，バス自体の消費電力が大きい，ボードサイズが固定，ラックの大きさにロボットの筐体が左右されてしまう等の問題点もある．将来的には，これらの問題を解決するロボット専用のバスもしくはリンクを設計したいと考えている．

また，データ転送量の多いモジュール間では，専用のバスを設けて，データ転送を局所的に高速に行えるように設計する．

4.2 機能別モジュール

ASPIRE-I では，以下の機能別モジュールを実装する（**図 2** 参照）．

- メインモジュール：MC68030，MC68882，イーサネット，SIO，PIO
- モータモジュール：TMP68301，DC モータ（PWM 制御），SIO，PIO

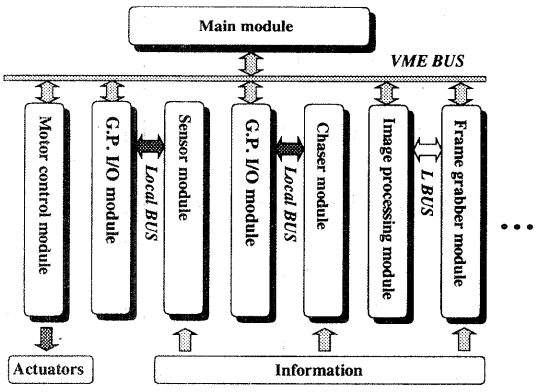


図 2 機能別モジュール
Fig. 2 Function-Classified Modules.

- センサモジュール：TMP68301，超音波センサ，赤外線センサ，タッチセンサ，光ファイバジャイロ，SIO
- *Chaser* モジュール[14]：TMP68301，MC68882，音源方向定位センサ，熱源方向定位センサ，SIO
- 汎用 I/O モジュール：TMP68301，Local bus，SIO
- 画像取り込みモジュール：640 × 480，24bit color 2 planes，Local bus
- 画像処理モジュール：T805 × 5，Local bus
- 音声合成モジュール：TMP68301，VCS11A-CIF，SIO
- 音声認識モジュール：TMP68301，MN1901611，SIO

ここですべてのモジュールの実装について述べることはできないので，以下では上記のモジュールのうち最も多くのモジュールで使用されている，PU に TMP68301 を用いているモジュール（モータモジュール，センサモジュール等）の設計・実装について述べる．

4.3 ブロックロボット

ASPIRE-I では，ユーザは自分が用いたいモジュール群を VME bus に挿せば自分の好みの構成のパーソナルロボットを実現できるようにブロックロボットとして設計する．

さらに，*ASPIRE-I* はモジュール自体を容易に追加できるように拡張性を考慮して設計を行う．そのために，まず VME bus インタフェースとローカルバスインタフェースを持ったマザーボード（汎用 I/O モジュール）を設計・実装する．新しい機能を持つモジュールの設計時には，そのローカルバスインタフェースを持つドータボードを設計するだけで，新しいモジュールを設計することが可能となる（図 2 参照）．

センサモジュールや *Chaser* モジュール等は汎用 I/O モジュール上のドータボードとして設計・実装されている．たとえば，**図 3** の汎用 I/O モジュール（VME ダブルハイトサイズ，6 層基板）にセンサ用ドータボードをアドオンすると**図 4** のセンサモジュールになる．

4.4 コミュニケーションメモリ

ASPIRE では，モジュール間で通信をするために，モジュール間にコミュニケーションメモリを配置する．*ASPIRE-I* ではモジュール間結合としてバス結合を用いるので，コミュニケーションメモリを共有メモリとして実装する．

4.4.1 分散共有メモリ

バス結合型の並列計算機では，一般に各モジュールが共有メモリや共有 I/O に頻繁にアクセスすると共有バスがボトルネックとなりシステム全体のスループットが低下する．*ASPIRE-I* ではこのボトルネックを避けるために分散共有メモリを採用する．自モジュール上の分散共有メモリをアクセスするときは共有バスを介さないので，自モジュール上の分散共有メモリには頻繁にアクセスをしても問題はない．

たとえば，モータモジュールの場合，モータに直結されているロータリエンコーダからの情報を元に計算した位置，移動距離，速度，加速度等を自モジュール上の分散共有メモリにリアルタイムに書き込み，ナビゲーションに関する最新情報を他モジュールから常に参照可能にする．同様に，センサモジュールの場合は，最新のセンサデータが得られるごとに自モジュール上の分散共有メモリにリアルタイムに書き込む．これら

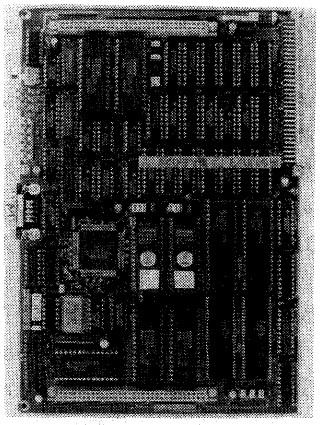


図 3 汎用 I/O モジュール
Fig. 3 General Purpose I/O Module.

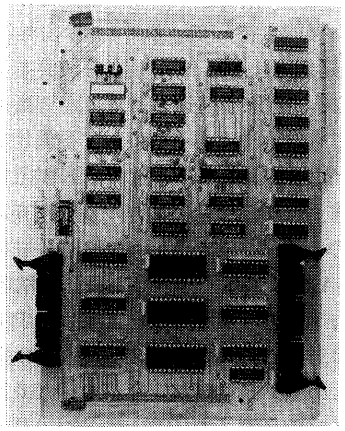


図 4 センサモジュール（汎用 I/O モジュール + センサ用ドータボード
Fig. 4 Sensor Module (G.P.I/O Module + Senser Daughter Board.

の情報は，できるだけ共有バス上のトラフィックを増やさないように，他モジュールがその情報を参照したいときのみ共有バスを介してアクセスされる．

分散共有メモリは，DPM（Dual Port Memory）素子を用いた高速小容量の分散共有メモリと，バスコントローラと疑似SRAMを用いた比較的大容量の分散共有メモリの2種類を用いて設計・実装する．

4.4.2 DPMを用いた分散共有メモリ

DPM素子は，2つのポートとアービタを内蔵し，両方のポートから自由に読み書き可能な素子である．このDPM素子の一方のポートをVME busに接続し，もう一方をLocal busに接続する．このようにDPM素子を使用すると，**図5**のShared bus☆の部分が存在しないので，Local bus側からもVME bus側からも高速にアクセスすることができる．また，DPM素子は2つのポート間で互いに割り込みをかけられるという重要な機能を持っている（4.5.2節参照）．

4.4.3 疑似SRAMを用いた分散共有メモリ

DPM素子は特殊な構造を持ち，かつ高速なアクセスを可能とする素子であるため，大容量化が困難であり，*ASPIRE-I*の実装では容量が4[KB]しかない．容量不足を補うため，もうひとつの分散共有メモリとして，図5のようにShared bus上にバスコントローラと疑似SRAMによって構成される比較的容量の大きな分散共有メモリを配置する．容量は1[MB]であるが，バスコントローラを介してShared bus上に配置されているので，アクセス速度がDPM素子を用いた実装にくらべると遅くなる．

また，これらの分散共有メモリを使用して各モジュール間で平等にデータ通信等を行うために，すべてのモジュールが平等にバスマスタになれるように設計する．

4.4.4 メールボックスアーキテクチャ

各モジュール間の通信は，自モジュール上あるいは通信先モジュール上の分散共有メモリ上に情報を書き込み，割り込みを用いて相手モジュールに知らせることによって行う．この機構によって，メイルボックスアーキテクチャ[15]をハードウェア的にサポートし，通信にレスポンシブ性を持たせることを可能にする．

4.5 割り込み

*ASPIRE-I*では*ASPIRE*に従い，すべてのI/O（センサ，I/Oポート等）に割り込み信号線を張り巡らせる（図5参照）．

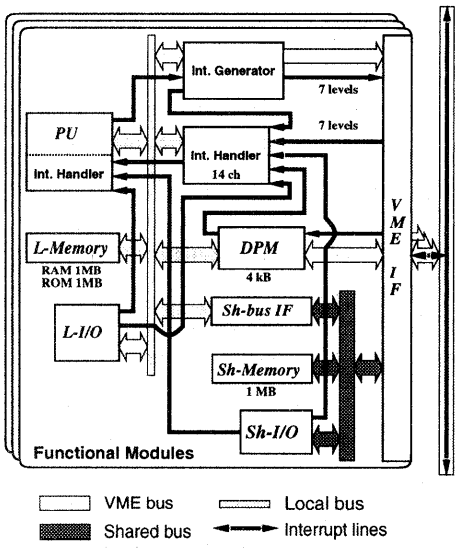


図5 *ASPIRE*上での通信
Fig. 5 Communication on *ASPIRE*.

4.5.1 VME busの割り込み

VME busの割り込みは，7レベル優先度付き割り込みである（付録参照）．この7レベルの割り込みをプログラマブルに変更でき，かつモジュール間で自由にかけたりかけられたりすることができるように設計を行う．

割り込みハンドリングは，2つの割り込みコントローラによって制御を行う．VME割り込みとローカル割り込み（計14チャネル）を一度に制御する割り込みコントローラ（SCB68155）とPU（TMP68301）に内蔵されている割り込みコントローラ（外部3チャネル，内部9チャネル）をカスケード接続することにより，多チャネルの割り込みを柔軟に処理可能にする．

VME busに対してソフトウェア的に割り込みをかける機構を設けるために，インタラプトジェネレータ（SCB68154）を用いる．インタラプトジェネレータは，プログラマブルに7レベルの優先度付き割り込みを発生させ，割り込みベクタをVME busに出力することができる．

4.5.2 DPMを用いたVME割り込み

モジュール間割り込みには，VME割り込み以外にDPM素子の割り込み機能を用いた別系統の割り込みを用意する．DPM素子には，2つのポート間のハンドシェイクを容易に実現するために，割り込み機能付きのものがある．そのようなDPM素子では，片方のポートからあるアドレスをアクセスすると，反対のポート側に割り込みをかけることができる．このDPM

☆ バスコントローラの制御によってVME bus側とLocal bus側の両方からアクセス可能なバス（図5参照）

素子の割り込み信号線を外付割り込みコントローラのローカル割り込み部に接続し，VME bus 側から DPM のあるアドレスにアクセスするとそのモジュール上にローカル割り込みがかかるように設計する（図 5 参照）．この DPM を用いた VME 割り込みでは優先度や割り込みベクタを用いることができないが，あるモジュールが他のモジュールに対して，正規の VME 割り込みを用いることなくプライベートに割り込みをかけることを可能にする．同時に，複雑なプロトコルの VME 割り込み（付録参照）に比べて，高速に割り込みをかけるという副作用も期待できる．この機構によって VME 割り込みを用いることなく DPM だけでメールボックスアーキテクチャを実現できる．

4.5.3 タイマ割り込み

μ-PULSER でのスレッドの並行実行およびリアルタイム性をサポートするために，PU に内蔵している 3 本のプリスケーラ付きカスケード接続可能 16bit タイマ/カウンタ以外に，外付の 3 本の 16bit タイマ/カウンタ（μPD71054）を実装する．これらのタイマ/カウンタを用いて，*μ-PULSER* 上で柔軟な時間管理を行うことを可能にする．

4.5.4 割り込み処理例

ASPIRE-I で提供するこれらの柔軟なハードウェア割り込み機構とオペレーティングシステム *μ-PULSER* での DI 機構によって各種のイベントをレスポンシブに処理することを可能にする．

たとえば，A モジュールが B モジュールに割り込みをかける場合，以下のようになる．

(1) A モジュール：インタラプトジェネレータを用いて B モジュールに割り込み
　　B モジュール：他のタスクの処理
(2) A モジュール：他のタスクの処理
　　B モジュール：割り込みアクノリッジサイクル
(3) A モジュール：割り込みベクタを VME bus に出力，同時に B モジュールに割り込みがかかったことを割り込みで PU に伝達
　　B モジュール：割り込みベクタの取得
(4) A モジュール：B モジュールに割り込みがかかった後の処理
　　B モジュール：割り込み処理

このように，VME bus に割り込みがかかったかどうかということまで，割り込みで知らせる．すべてのイベントが起こる可能性のあるものに徹底的に割り込み線を張りめぐらせている結果，イベント伝達の即時性およびプロセッサのアイドリングを可能な限り低く抑えることを実現する．

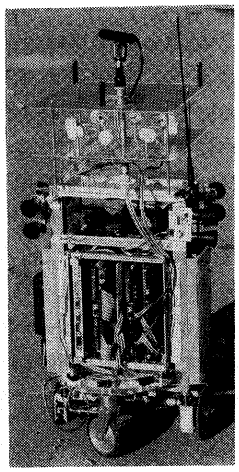


図 6 パーソナルロボット *ASPIRE-I* の概観
Fig. 6 Personal Robot *ASPIRE-I*.

4.6 筐　　体

今回実装したパーソナルロボット *ASPIRE-I* の筐体を含めた概観を図 **6** に示す．*ASPIRE-I* は，左右に駆動輪を持ち前後に補助輪を持つ構成をとる．図 6 を見ると，赤外線センサ，超音波センサ，タッチセンサ，光ファイバジャイロ，無線通信装置等が取り付けられているのが分かる．

4.7 機能別並列計算機

ASPIRE-I は，PU としてマイクロプロセッサを使用した汎用バス結合型機能別並列計算機として設計を行っている．いわゆる並列計算機は（ほとんど）同じ PU が複数結合し互いに協調して計算を行うものである．しかしながら *ASPIRE-I* の場合，各モジュールは機能の異なる I/O を持ち，粒度の大きい機能分散されたタスクを互いに協調しながら並列実行を行うという，通常とは異なる形式の MIMD 型並列計算機である．さらに *ASPIRE-I* の場合，モジュールごとの目的に合わせて PU さえも異なる種類のものを使用している．基本的には各モジュールをエージェントとしてとらえ，あるモジュールが別のモジュールにタスクを依頼することによって並列処理を行う．

我々は，一般の並列計算機と区別するためにこのような構造の並列計算機を機能別並列計算機と呼ぶ．近年，様々な汎用もしくは科学技術計算専用の並列計算機が設計・実装されている．それらの並列計算機の多くでは，本論文で問題にしているレスポンシブ性はほとんど必要がないので，そのシステムの設計時にリアルタイム性やリアクティブ性はほとんど考慮されていない．また，汎用もしくは科学技術計算専用の並列計算機では，PU ごとに I/O が異なるという特殊な状況

もほとんどあり得ない．

従来は組み込み用途としてワンボードマイコンやパーソナルコンピュータで制御されていたロボット等の制御の分野において，さらに複雑で大規模な処理を自律的に行うために，我々は従来の並列計算機の研究を応用し，かつレスポンシブ性を重視し機能分散した並列計算機としてのアーキテクチャ *ASPIRE* を提案する．この意味では，*ASPIRE* は組み込み用途や制御向けの並列計算機アーキテクチャであると考えられる．我々は，このように各モジュールが異なるI/Oや異なる種類のPUを持って機能分散している形態の機能別並列計算機が，制御の分野などで今後さらに重要になっていくと考えている．

5. *ASPIRE-I* の動作

ASPIRE-I では，基本的にメインモジュールがプランニングを行ったタスクを各モジュールに依頼し，各モジュールはその依頼に従ってタスクを実行するという形式でシステム全体の制御を行う．たとえば，メインモジュールは，パスプランニングした経路情報をモータモジュールに渡してナビゲーションを行うように依頼する．同様にセンサモジュールには，進行方向の50 cm前方に障害物が現れたらモータモジュールに割り込みを掛けること，および最新のセンサ情報を常に分散共有メモリに書き込み他モジュールからの参照ができるように依頼する．この場合，モータモジュールが経路情報に従ってナビゲーションをしている際にセンサモジュールが急に障害物を発見すると，センサモジュールはメインモジュールを介することなくモータモジュールに直接割り込みをかけ緊急停止を行う．その後，センサモジュールとモータモジュールが協調しながら障害物を回避するが，そのときメインモジュールは障害物回避後のリプランニングを並列して行う．障害物回避が行われた後には，すでに新しいプラン（経路情報等）が求められており，メインモジュールは新たに各モジュールにタスクの依頼を行う．

各モジュールは常に内部状態を自モジュール上の分散共有メモリに書き込む．メインモジュールは各モジュールの分散共有メモリ上にある内部状態をチェックし，各モジュール間で背反した行動をとっている場合には調停を行う．

6. 評価および考察

ここでは，*ASPIRE-I* のハードウェアの基本性能を測定し評価を行う．*μ-PULSER* と組み合わせた性能は，オペレーティングシステムの設計に強く依存し，*μ-PULSER* の設計を述べたうえでないと意味がないので，他の文献[10),13),16)~18)]を参照していただきたい．

表1 VMEの割り込み時間
Table 1 VME interrupt time.

	Motor → Sensor	Sensor → Motor
Num. of data	10	10
Average [μ sec]	2.67	2.45
Std deviation	0.42	0.31

表2 DPM割り込みを用いたVME間の割り込み時間
Table 2 VME interrupt time using DPM.

	Motor → Sensor	Sensor → Motor
Num. of data	10	10
Average [μ sec]	1.32	1.31
Std deviation	0.10	0.11

6.1 割り込み

6.1.1 モジュール間割り込み

VME間の割り込み処理の速さは，モジュール間のイベント伝達の速度に直接影響を与える．

ここでは，モータモジュールとセンサモジュールを用いて割り込みの伝播時間を測定した．測定には，デジタルストレージを使用し，割り込みをかけた瞬間からその割り込みがアクノリッジされた瞬間までの時間間隔を測定した（**表1** 参照）．

同様にモータモジュールとセンサモジュールを用いて，DPM割り込みを用いたVME間の割り込みの伝播時間を測定した（**表2** 参照）．

表1よりVMEの割り込み時間は約2.5 [μsec] となっている．ここで *μ-PULSER* の1量子時間は10 [msec] なので，量子時間の約0.025 [%] でモジュール間でイベントを伝達することができる．これによって，*μ-PULSER* のリアクティブなスケジューリングをサポートする．

また，表1と表2の結果から，DPMを用いたVME間の割り込みは通常のVME割り込みに比べて2倍程度速いことが分かる．この高速なDPM割り込みを用いることによって，メールボックスアーキテクチャをハードウェア的にサポートする．

6.1.2 緊急停止処理

センサモジュールとモータモジュールを用い，センサモジュールのタッチセンサを触った瞬間からモータが停止を始めるまでの時間をデジタルストレージを用いて10回測定した（**表3** 参照）．

ASPIRE-I の最大速度は約1.5 [m/sec] なので，表3よりタッチセンサに接触してから停止動作を開始するまでの距離は，

$$1.5\,[\mathrm{m/sec}] \times 136\,[\mu\mathrm{sec}] = 0.204\,[\mathrm{mm}]$$

表3 緊急停止時間
Table 3 Emergency halt time.

	Touch sensor
Num. of data	10
Average [μsec]	136.0
Std deviation	1.3

表4 分散共有メモリのアクセス時間
Table 4 Access time of distributed shared memory.

	DPM	P-SRAM
Num. of data	10	10
Average [nsec]	427	602
Std deviation	4.0	16.5

となり，最大速度で移動している場合でも，障害物に接触してから約0.2 [mm] 移動した後，停止動作を開始することができる．人間の早歩きの速度が約1.5 [m/sec] なので，人間と *ASPIRE-I* が向き合って接近している場合でも同じオーダで議論できる．実際には，タッチセンサに接触してから停止するのは稀であり，ほとんどの場合は，超音波センサか赤外線センサによって障害物が発見される．たとえば，障害物かどうかを判断する閾値を50 [cm] に設定した場合，50 [cm] 前方に障害物が発見されると，停止距離を考慮しても通常の航行速度（40～70 [cm/sec] 程度）であれば数 [cm]～10数 [cm] 進んだだけで停止することができる（路面の摩擦係数によって緊急停止距離は変化する）．

6.2 コミュニケーションメモリ

6.2.1 分散共有メモリのアクセス速度

VME bus を介して他モジュール上のDPM素子と疑似SRAMにワードアクセスし，データを1ワード転送するまでの時間を測定した．測定にはデジタルストレージを用い，モジュール上のアドレスデコーダからチップセレクトが出力されてから，1ワード転送（read）終了するまでの時間を測定した（**表4** 参照）．

DPM のアクセス時間は 427 [nsec] であり，疑似SRAMのアクセス時間は602 [nsec] であった．これより，疑似SRAMよりもDPMの方が約1.4倍アクセスが速いことが分かる．このように，アクセス速度はDPMの方が速いが，各モジュールごとに4 [KB] しかない．一方，疑似SRAMの方はアクセス速度は遅いが，各モジュールごとに1 [MB] ある．したがって，プログラミング時には，この2種類の分散共有メモリの使い分けが重要となる．

6.2.2 バス使用率とデータ転送速度

各メモリ階層に関して，バス使用率とデータ転送速度に関する評価を行った．VME bus に負荷をかけながら，VME bus を介して1 [KB] のデータをワード単位（16 [bit]）で転送を行い，データ転送に要する時間を測定した．2系統の分散共有メモリを VME bus を介して他モジュールからアクセスする場合，同じく自モジュール側からアクセスする場合，比較のためのローカルメモリにアクセスする場合について測定を行った．PU の move 命令を用いて1 [KB] のデータを転送するのにかかる時間を内蔵タイマによって測定した（**図7** 参照）．

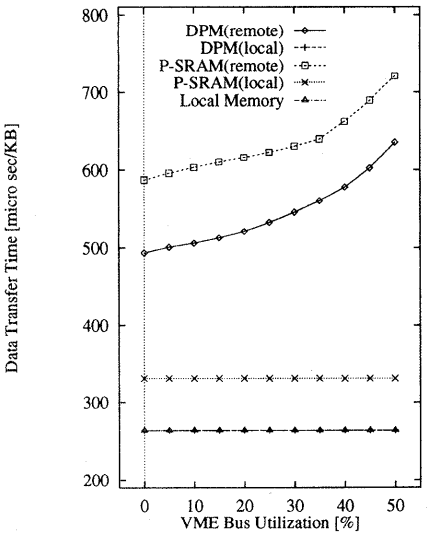


図7 バス使用率とデータ転送速度の関係
Fig. 7 Data Transfer Rate.

まず，VME bus 側（remote）からの分散共有メモリへのデータ転送速度を比較する．図7より，VME bus に全く負荷をかけていない場合，DPM へのデータ転送時間は494 [μsec/KB]（1.98 [MB/sec]），疑似SRAMの方は588 [μsec/KB]（1.66 [MB/sec]）となり，約1.2倍ほどDPMを用いて実装した分散共有メモリの方がデータ転送が速いことが分かる．同様に，測定したすべてのバス使用率において，DPMの方が疑似SRAMに比較して高速転送できることが分かる．

また，バス使用率が30 [%] くらいまでは，DPMと疑似SRAMともにほぼ線形にデータ転送時間が増加しているが，30～40 [%] くらいから，急激にデータ転送時間が増加し始め，バスの飽和が起こり始めているのが分かる．この図7より，バス使用率からデータ転送時間を線形予測可能にしてリアルタイム性を保証するためには，バス使用率は30 [%] 以内が望ましいと考えられる．

次に，自モジュール（local）上の分散共有メモリへのデータ転送速度を比較する．図7より，VEM bus

の使用率にかかわらず，Local bus 側からの DPM へのデータ転送時間は 264 [μsec/KB]（3.70 [MB/sec]），疑似 SRAM の方は 331 [μsec/KB]（2.95 [MB/sec]）となり，約 1.3 倍ほど DPM を用いた実装の方が速いことが分かる．自モジュール上の DPM は，ローカルメモリと全く同じ性能を有しており，同様に自モジュール上の疑似 SRAM は少し遅いローカルメモリとみなすことができる．また，VME bus に他の負荷がかかっていないときでも，同一デバイスへの他モジュール（remote）との転送の場合に比べて 1.8～1.9 倍データ転送速度が速いことが分かる．

よって，VME bus の使用率を約 30 [%] 以内に抑えるとリアルタイム性が保証できること，および VME bus の使用率にかかわらず自モジュール上の分散共有メモリには高速にアクセスでき，かつ共有バスである VME bus には悪影響を与えないということから，自モジュール上の分散共有メモリに最新の位置情報（モータモジュールの場合）やセンサ情報（センサモジュールの場合）をつねに書き込んでおき，他モジュールがそれらの情報が必要になったときにのみ参照する方法は，非常に有効であることが分かる．

7. むすび

本論文では，レスポンシブ・システムとしてパーソナルロボット用機能別並列計算機アーキテクチャ *ASPIRE*（*ASynchronous, Parallel, Interrupt-based and REsponsive architecture*）を提案・設計した．そして，*ASPIRE* に基づいて，プロトタイプのパーソナルロボット *ASPIRE-I* を設計・実装した．*ASPIRE-I* は，すべての I/O に割り込み線を張り，VME bus で結合し，2 系統の分散共有メモリと 2 系統の VME 間割り込みを持ち，ブロックロボットとして実装を行った機能別並列計算機である．

ASPIRE ロボット上のアプリケーションとしては *Active Interface* という新しいユーザインタフェースの概念を提案し[14]，それを用いて音声対話システム[14]，TV 会議システム[19]，会場案内システム[20]等に応用している．これらのアプリケーションでは人間とのインタラクションの際のインタフェースが重要となってくるが，*ASPIRE* で得られるレスポンシブ性を用いて，ユーザにストレスのかからないユーザフレンドリなインタフェースおよび動作の実現を可能とする．

また，我々は現在も *ASPIRE* を発展させる試みを行っている．現在，レスポンシブ性と高速演算性の両方を実現するために，*ASPIRE* に基づき PU に，従来は組み込み用途には不向きだと考えられていた RISC（μ-SPARC, SPARClite）を用いてパーソナルロボット *ASPIRE-II* を設計・実装し，評価を行っている[21]．

謝辞 *ASPIRE* の実装に協力していただいた安西・天野研究室の皆様に感謝いたします．

参考文献

1) Anzai, Y.: Towards a New Paradigm of Human-Robot-Computer Interaction, *Proceedings of IEEE International Workshop on Robot and Human Communication*, pp.11–17 (1992).
2) Brooks, R.A.: A Robust Layered Control System For A Mobile Robot, *IEEE Journal of Robotics and Automation*, Vol.RA–2, No.1, pp.14–23 (1986).
3) Brooks,R.A. and Flynn,A.M.: FAST, CHEAP AND OUT OF CONTROL: A ROBOT INVASION OF THE SOLAR SYSTEM, *The British Interplanetary Society*, Vol.42, pp.478–485 (1989).
4) Connell, J.H.: SSS: A Hybrid Architecture Applied to Robot Navigation, *Proceedings of IEEE International Conference on Robotics and Automation*, Nice, France, pp.2719–2724 (1992).
5) Miller, D.P. and Gat, E.: Exploiting Known Topologies to Navigate with Low-Computation Sensing, *SPIE Sensor Fusion III: 3-D Perception and Recognition*, Vol.1383, pp.425–435 (1990).
6) Stewart, D.B., Schmitz, D.E. and Khosla, P.K.: Implementing Real-Time Robotic Systems Using CHIMERA II, *IEEE International Conference on Robotics and Automation*, Cincinnati, Ohio, pp.598–603 (1990).
7) 油田信一：自立移動ロボット，機械の研究，Vol.43, No.1, pp.137–144 (1991).
8) 油田信一，飯島純一：自律移動ロボットの制御系のアーキテクチャ，第 8 回日本ロボット学会学術講演会予稿集，pp.967–970 (1990).
9) 山崎信行，安西祐一郎：パーソナルロボットのためのアーキテクチャの提案，ロボティクス・メカトロニクス講演会 '92 講演論文集，Vol.A, pp.51–56 (1992).
10) 矢向高弘，菅原智義，安西祐一郎：PULSER: リアクティブシステムの構築に適したオペレーティングシステム，コンピュータ・ソフトウェア，Vol.11, No.1, pp.24–35 (1994).
11) Benveniste, A., Guernic, P.L., Sorel, Y. and Sorine, M.: A Denotational Theory of Synchronous Reactive Systems, *Information and Computation*, Vol.99, pp.192–230 (1992).
12) Stankovic, J.A.: Misconceptions About Real-Time Computing, *IEEE Computer*, pp.2–10

(1988).
13) 矢向高弘，菅原智義，安西祐一郎：μ-PULSER: パーソナルロボットを構築するためのオペレーティングシステム，電子情報通信学会論文誌，Vol.J77-D-I, No.2, pp.207-214 (1994).
14) Yamasaki, N. and Anzai, Y.: *Active Interface* for Human-Robot Interaction, *Proceedings of IEEE International Conference on Robotics and Automation*, Vol.3, Nagoya, Japan, pp.3103-3109 (1995).
15) Faro, A., Mirabella, O. and Vita, L.: A Multimicrocomputer-based Structure for Computer Networking, *IEEE Micro*, Vol.5, No.2, pp.53-66 (1985).
16) Yakoh, T. and Anzai, Y.: A New Reactive Operating System for Human-Robot Interaction, *Advanced Robotics*, Vol.8, No.4, pp.371-383 (1994).
17) 河野通宗，飯田浩二，矢向高弘，安西祐一郎：分散共有メモリを用いた実時間ページングの提案と実装，情報処理学会システムソフトウェアとオペレーティングシステム研究会報告，pp.9-16 (1994).
18) 河野通宗，飯田浩二，矢向高弘，安西祐一郎：分散共有メモリを用いた実時間での記憶管理に関する考察，情報処理学会第48回（平成6年前期）全国大会論文集，pp.4-23-4-24 (1994).
19) Yamasaki, N. and Anzai, Y.: Applying Personal Robots and *Active Interface* to Video Conference Systems, *Proceedings of 6th International Conference on Human-Computer Interaction*, Vol.2, Yokohama, Japan, pp.243-248 (1995).
20) 山崎信行，安西祐一郎：*Active Interface*を用いた会場案内システムの設計と実装，第13回ロボット学会学術講演会講演集，No.3, pp.1107-1108 (1995).
21) 山崎信行，澤村省治，山本 純，安西祐一郎：パーソナルロボット用アーキテクチャ*ASPIRE*のRISCを用いた設計と実装，第12回ロボット学会学術講演会，No.1, pp.303-304 (1994).

付 録

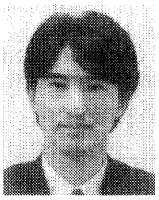
A.1 VME busのバス使用権

あるモジュールがVME busでバス使用権を獲得するには，そのモジュール上のリクエスタが4レベルあるバスリクエストのうちの1つを出力する．バス獲得の調停のためにバス上（メインモジュール）に1つのアービタがある．このアービタが，固定優先度，ラウンドロビン，単一レベルの3種類のバス調停法のうちのどれかを行い，バス使用権を各モジュールに与える．*ASPIRE-I*では，ラウンドロビンを用いる．

A.2 VME busの割り込み

VME busの割り込みは，7レベルの優先度付き割り込みである．あるモジュールがVME割り込みをかけたいときには，インタラプタによって要求し，割り込みレベルがあっていればインタラプト・ハンドラによって受理される．このとき，インタラプト・ハンドラは，リクエスタを使って上述のようにしてバス使用権を獲得し，アクノリッジした割り込みレベルをアドレスバス（A1-A3）を用いてインタラプタに知らせる．インタラプタは，自分の要求した割り込みレベルと合致していれば，割り込みベクタをVME busのデータバスに出力する．この一連のサイクルは，割り込みアクノリッジサイクルと呼ばれる．同一レベルの割り込みは，アービタに近いスロットからディージーチェーンされる．

(平成7年 4 月11日受付)
(平成7年11月 2 日採録)



山崎 信行（学生会員）

昭和41年5月1日生．平成3年慶應義塾大学理工学部物理学科卒業，平成5年同大学大学院理工学研究科計算機科学専攻修士課程修了．同年同専攻後期博士課程入学．現在，同専攻博士課程在学中．自律移動ロボット，並列計算機，OS，コンピュータアーキテクチャなどに興味を持つ．日本ロボット学会会員．

安西祐一郎（正会員）

昭和21年8月29日生．昭和49年慶應義塾大学大学院博士課程修了．昭和63年より慶應義塾大学理工学部教授．平成元年より同大学大学院計算機科学専攻教授兼任．この間，昭和56～57年カーネギーメロン大学客員助教授．計算機科学，認知の情報処理過程の研究に従事．工学博士．日本ロボット学会，電子情報通信学会，日本神経回路学会，日本心理学会，ACM，IEEEなどの会員．