3M−3

デリバリー用LISPシステムの生成手法*

杉山広幸†
sugiyama@ntt-elis.ntt.jp
NTTヒューマンインタフェース研究所‡

1 はじめに

新ELIS[1]では、LISPの適応領域をより広げることを狙い、ハードウェアの形態として、従来のワークステーション型に加え、システム組み込みをめざしVMEバス用[2]、PC用[3]の2種類のボードを実現している。表1にこれらの主要諸元を示す。このように利用できるリソースの異なる複数のハードウェア上に、組み込みシステムから、実行機、開発機まで種々な形態のLISPシステムを実現するためのシステム生成機構を検討、実現した。

本稿では、このシステム生成機構と、それを用いて生成したELIS-VMEボード(以下VMEボード)用LISPシステムについて報告する。

2 モジュール化

柔軟なシステム構成のためには、適切なモジュール分割がなされていることが必要である。ELISではモジュール分割の単位として'system'を用いている。'system'は、(1)パッケージの定義、(2)'system'の初期化を行うための関数名、(3)'system'を構成するファイルのパス名とロードされるパッケージ名、(4)'system'のインタフェースとして「export」されるシンボル名、等からなり、'defsystem'構文を用いて定義する。'system'は'system'の集合としても定義される。'system'を操作する関数として、'system'をロードする「load-system」、'system'の初期化関数を実行する「apply-system-initfn」、'system'をコンパイルする「compile-system」等がある。

ネットワークの'system'定義の例を以下に示す。

```
(defsystem :network
  (:package "NETWORK" :global t
     :nicknames ("NET") :reentrant t)
  (:initialize-function :startup
     ("INIT-TCP-DEVICE" "IO")
     ("INIT-UDP-DEVICE" "IO") )
  (:default-directory "system\\network\\" "net:")
  (:default-loading-package "NETWORK")
  (:files ("INPUT-OUTPUT" "netdr") "netdb")
  (:export "NETWORK"
    ("CONNECT-NET" "LISTEN-NET"
       ...
     "GET-GW-ADDRESS" )))
```

3 システム生成

システム生成は、システムの構成を定義する'boot'にもとづき行う。'boot'はシステムを構成する'system'と、システム生成に用いる幾つかの情報(主記憶の割り当て、サイト情報、feature等)からなり、その定義は'defboot'構文で行う。相互に参照関係にある'system'はまとめて'phase'としてリストで表し、参照される'phase'から並べて、':systems'スロットに記述する。'boot'を扱う関数として、'boot'に定義された'system'をロードする「load-boot」、シンボルを'export'する「export-symbols」、初期化関数を実行する「apply-init-function」、コンパイルする「compile-boot」、'boot'に定義されている情報を得るための「get-boot-info」等がある。

表1: 新ELISのハードウェア形態

ハードウェア形態	ワークステーション デスクサイド	ワークステーション デスクトップ	PCボード	VMEボード
制御記憶	64Kword WCS	64Kword WCS	64Kword WCS	32Kword ROM
主記憶	8MB(最大32MB)	8MB(最大16MB)	8M(最大32MB)	8M 注)
ハードディスク	40MB(最大80MB)	270MB(最大1.89GB)	—	—
適応領域	開発システム	システム組み込み	システム組み込み 開発システム	機器組み込み

注)外部に1Mのバッテリー・バックアップ・メモリを持つ

*Genaration Method for Delivery Lisp System
†HIROYUKI, Sugiyama
‡NTT Human Interface Laboratories

'boot' の定義から、'system' の追加・削除を行うことにより、容易に異なる構成のシステムを生成できる。

VME ボード用の 'boot' の定義を以下に示す。

```
(defboot :vme
  (:memory "system\\site\\memory.tbl")
  (:site "system\\site\\site.tbl")
  (:startup-message
   "ELIS CommonLisp for ELIS-VMEboard")
  (:features
   ("ELIS" "KEYWORD")
   ("ELIS-8200" "KEYWORD")
   ("VME" "KEYWORD") )
  (:systems
   (:primitive)
   (:define-fun)
   (:fas-loader)
   (:definition)
   (:varray :vnumber :type :char
    :control :verror :predicate
    :hash :list :package :symbol
    :sequence :string :realtime-kernel
    :misc )))
```

起動方式として lboot 方式、boot 方式の 2 つの方式がある。lboot 方式は、システム生成のための起動方式で、指定された 'boot' に記述されている順序に従い 'system' をロードし、システムを起動する。起動後の主記憶のイメージを出力し 'world' ファイルを作成する。boot 方式では、この 'world' ファイルをロードし、システムを起動する。

ワークステーション上でシステム生成時に、デリバリーシステムの主記憶サイズと 'boot' を指定することでデリバリー用システムの生成を行うことができる。

4 VME ボード用システムの実現

ワークステーション上で、ネットワークから、ウィンドウシステムまですべての開発環境を含んだ最大構成においては制御記憶 56Kword、主記憶 8.2MB を必要とする。

VME ボード用システムでは、ボードサイズの制限により制御記憶が 32Kword に、さらにハードディスクの代わりに用いるバッテリー・バックアップ・メモリの容量が 1Mbyte のために 'world' のサイズは 1Mbyte に制限される。また、1Mbyte のうち AP の使用部分を考慮するとシステムで利用できる領域は 800Kbyte 程度である。これらの条件と、VME ボードの適応領域 (産業機器組み込みで、24 時間連続運転、停電自動復旧が要求される用途) から、以下の観点からモジュールの選択を行った。

表 2: 主記憶使用量

開発環境	VME ボード用システム	
	compaction 無し	compaction 有り
8435KB(8.2MB)	936KB	837KB

- 入出力機能 (reader,printer,stream 等) の削除
- 算術演算では fixnum 以外データ型を削除
- シーケンス、配列の非破壊版関数の削除
- ロジック、オブジェクト指向機能の削除
- 開発環境 (エディタ、コンパイラ等) の削除

この結果、CommonLisp の機能をほぼ保ちながら、制御記憶は 32Kword と約半減、主記憶は 936KB と 1/10 程度に削減できた。

さらに、主記憶使用量を削減するために、以下に示すように実行環境として必要な最低限の情報以外は極力削除し、表 2 に示す様に、目標をほぼ達成できた。

- デバッグ用情報の削除
- マクロ定義の削除
- 内部シンボルの関数,defstruct 定義等を削除
- 内部シンボルのシンボル名を空文字列に変更
- 内部シンボルの unintern

5 まとめ

多様な形態の LISP システムを構築するためのシステム生成機構を実現、それを VME ボードに適応した。このような柔軟なシステム生成が可能になったことにより、従来リソースの制限により LISP を用いることが困難であった領域においても LISP が利用できる。

謝辞

本システムの実現にあたり、御指導いただいた日比野靖グループリーダをはじめとする、グループの方々に感謝いたします。

参 考 文 献

1) 鈴木他: 新 ELIS システム概念, 1989 年信学会秋季全国大会, D-137, 1989

2) 渋谷、小平、鈴木: 機器組み込み用 ELIS-VME ボードのハードウェア, 情報処理学会第 42 回全国大会, 1991

3) 川村:1 ボード ELIS/PC システム, 情報処理学会第 42 回全国大会, 1991