論文 85-22-5-6

工場 自動化응용을 爲한 Satellite Processor System 硏究

(A Study on Satellite Processor System for Factory Automation)

金 鍾 晋*, 朴 贊 益*, 金 明 桓*, 朴 圭 晧*

(Jong Jin Kim, Chan Ik Park, Myung Hwan Kim and Kyu Ho Park)

要 約

本 論文에서는 工場 自動化 分野에 適合한 컴퓨터 시스템에 關하여 硏究하였고, 基礎的인 satellite processor 시스템을 構成하였다. 全體 시스템은 하나의 運營體制에 의하여 制御되어 階層的 構造를 가진다.

기존의 UNIX 시스템과 完全 互換性을 가지도록 하기 위하여 UNIX 運營體制를 修正 補完한 SPUNIX 運營體制를 構成하였다. 또한 應用 動作에 近接하는 動作을 하게 하기 위하여 特別한 프로그램— 이것을 co-process라 부르기로 한다— 과 이 co-process와 相互 協力하여 전체 動作을 形成하는 satellite processor kernel 等이 構成되었다.

많은 數의 satellite processor가 存在하기 때문에 信賴度, 擴張性, 同時性(concurrency) 等의 特性이 나타난다.

Abstract

For the application of factory automation, a UNIX system with satellite processor, which can be categorized into master / slave hierarchical structure, is studied and implemented.

The kernel part of UNIX is modified for the master/slave structure, but it is designed fully compatible with the existing UNIX systems. A special user process is created to make easy to down load developed program from host to satellite processor with the concept of co-process. Also satellite processor can send messages and request certain services from host computer. The design principles considered here are reliability, expanaibility, and concurrency.

I. 序 論

컴퓨터 시스템의 應用分野는 一槪的인 用度뿐 아니라 工場 自動化, 事務 自動化, 家庭 自動化 等 거의 미치지 않는 領域이 없으리만치 넓어졌다. 이렇게 各各 應用分野가 變化함에 따라 그에 最適한 컴퓨터 시스템의 하드웨어나 소프트웨어 形態도 또한 變化하여야 하는 것은 當然한 歸結이라 생각된다.

本 論文에서는 이 중 工場 自動化에 最適合한 컴퓨터 시스템에 관하여 基礎的인 硏究를 satellite processor 槪念을 도입하여 遂行하였다.

이러한 工場 自動化의 全型的인 作業 形態는 大型

*正會員, 韓國科學技術院 電氣 및 電子工學科
(Dept. of Elec. Eng., KAIST)

接受日字 : 1985年 3月 20日

컴퓨터의 處理能力을 必要로 하지 않는, 여러 종류의 比較的 간단한 일(task)들, 즉

① 주어진 時間內에 處理되어야 하는 工程

② 많은 量의 데이터를 外部로 부터 入出力장치(I/O)를 通하여 收集하는 作業

③ 간단한 control algorithm에 의한 데이터 處理 作業

④ 外部 工程 以外에는 相互聯關이 거의 없는 作業

等의 特性을 가지는 일들과, 가끔 大型 컴퓨터의 處理能力, 補助 記憶 장치들을 必要로 하는 일들로 構成된다.

그러므로 자기에게 주어진 단순한 作業만을 專擔하는 小型 컴퓨터를 slave로 하고, 하나의 大型 컴퓨터를 master로 하는 master/slave 階層 構造가 이러한 工場 自動化 分野에서는 應用 構造와 그에 對應되는 시스템 構造가 一致한다는 則面에서 가장 適合하다. 이 같은 形態의 시스템은 그림 1에 圖示한 것과 같다.

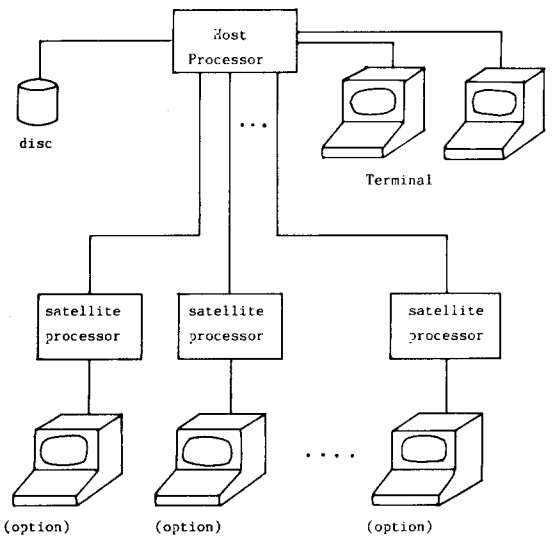

그림 1. 主從關係를 가지는 satellite system
Fig. 1. Master/slave hierarchy system with satellite processors.

Ⅱ. 本 論

1. Satellite processor(以下 SP) 概念

SP 시스템 概念은 원래 Bell 연구소에서 特殊目的에의 應用을 위한 미니컴퓨터와 마이크로컴퓨터의 數가 증가함에 따라 適切한 소프트웨어를 提供하기 위해 생겨났으며, 이 SP 概念은 대형 컴퓨터 시스템의 長點을 많은 마이크로프로세서에 賦與하여 特殊目的 마이크로컴퓨터가 實時間 應答 및 柔軟性을 維持하면서 主 컴퓨터의 파일 시스템, 소프트웨어 tool 및 peripheral등에 接近할 수 있도록 하는 것이다.[1] 즉, 主 컴퓨터의 運營體制를 여러개의 SP로 擴張하는 것이라 할 수 있다.

이와같은 SP 시스템에서는 SP에서 遂行中인 user program과 host processor(以下 HP)의 運營體制와의 通信 역시 system call에 의해 하지만 SP의 system call을 trap exception에 의해 處理하여 system call number 및 arguments등을 HP로 傳達한다는 點이 일반적인 컴퓨터 시스템과의 差異點이다.

이와 같은 概念을 그림으로 나타내면 그림 2와 같다.

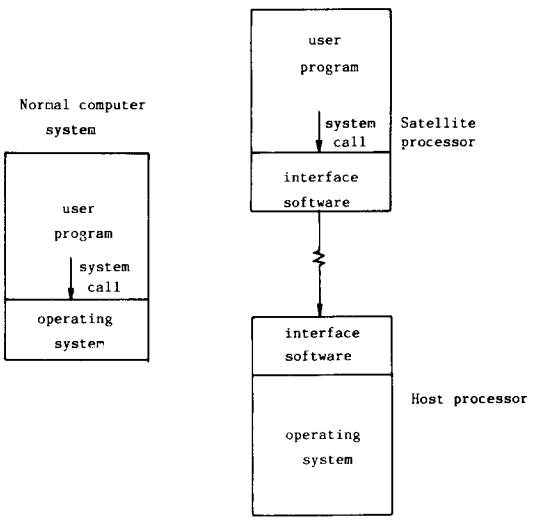

그림 2. Satellite processor 概念
Fig. 2. Satellite processor concept.

2. 하드웨어 構造

本 시스템의 構造는 그림 3에서 보는 바와 같이 여러 資原(resource)을 갖고 있는 主 컴퓨터와 몇 個의 SP들로써 이루어져 있다. 現在 이 시스템에서는 主 컴퓨터로 MC68000 CPU를 사용한 UNIX 시스템을 쓰고 있으며, SP는 同質性(homogeneity)를 높히기 위해 主 컴퓨터와 같은 CPU를 가진 single board 컴퓨터를 使用하였다.

그림 3과 같이 HP와 SP간에 主從 構造를 이룸으로써 I/O를 위한 일이나 시스템 관리는 HP가 擔當하고, 빠른 應答을 要求하는 계산을 위한 일은 SP가 擔當함으로써 實時間 應答을 要求하는 시스템에 應用할 수 있게 하였다.

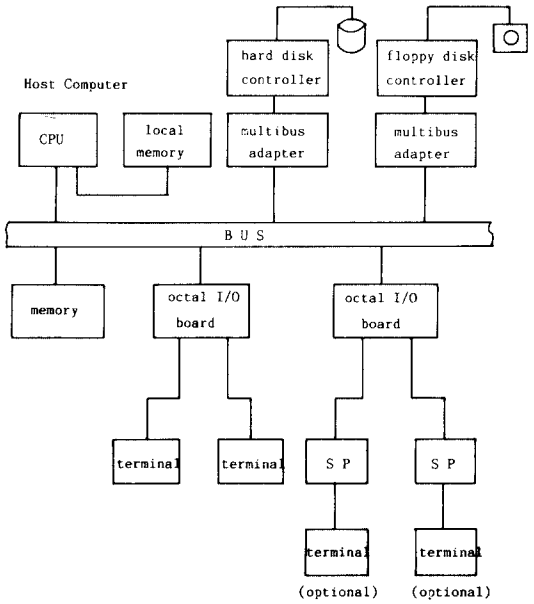

그림 3. 시스템의 構造
Fig. 3. System architecture.

3. 소프트웨어 構造

앞 節에서 說明한 하드웨어를 管理하기 위하여 必要한 소프트웨어의 構成은 그림 4 에 圖示한 바와 같이 主要 役割에 따라 SPUNIX kernel, Co-process, Satellite processor kernel로 크게 區分할 수 있으며 各各의 技能은 다음과 같다.

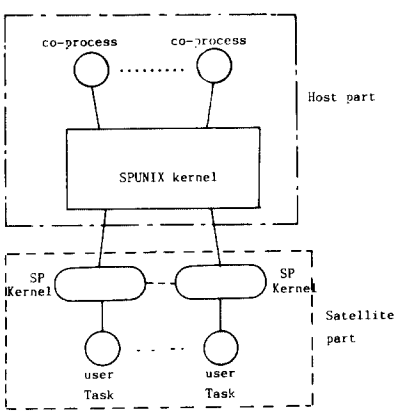

그림 4. 소프트웨어의 槪略的 構造
Fig. 4. Overall configuration of software block.

4. SPUNIX Kernel

SPUNIX kernel은 앞에서 說明된 하드웨어를 管理하는 運營體制(operating system)로서, 基本的으로 UNIX kernel을 修正하여 構成된 것이다. SPUNIX kernel을 構成하기 위하여 UNIX kernel에 添加되는 部分은 役割에 따라 다음 두 部分으로 나뉜다.

i) Satellite processor의 管理

SPUNIX kernel에서는 satellite processor를 하나의 資原(resource)으로서 認識하여 管理를 하는데, 그에 reguest-SP( ), release-SP( ) 動作(routine)들이 必要하다. 이러한 動作들에 의해 運營되는 global 데이터 構造들에는, SPUNIX kernel內에 존재하는 SP table, RSP table, PROC table 等이 있다. 各 table들의 相互關係는 그림 5 에 表示하였다.

SP table의 한 entry는 시스템에 登錄된 satellite processor 중의 하나와 一對一 對應 關係를 形成하면서 그 satellite processor의 狀態를 記錄한다. 즉, SP table의 한 entry에는 SP- id와 SP-stat가 있어, 各各 satellite processor 固有의 番號(identification number)와 現在의 satellite processor 狀態(busy 혹은 free)를 表示한다.

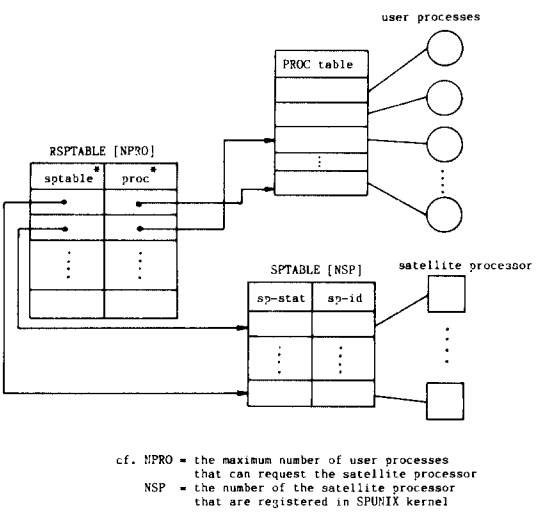

그림 5. Satellite processor 運營을 爲한 SPUNIX kernel 內의 global 데이터 構造
Fig. 5. SPUNIX kernel data structure for managing satellite processors.

RSP table은 satellite processor를 要請한 user process에 對한 情報를 貯藏한다. 즉, RSP table의 한 entry에는 SP table pointer와 PROC table pointer가 있어, user process와 그에 連結되는 satellite processor를 結合시키는 橋梁 役割을 한다.

Request-SP( )와 release-SP( ) routine의 概要는 다음과 같다.

```
request-SP( );
{
set up process-id into RSP table entry;
Loop: search free satellite processor;
      if found
        arrange SP table;
        return allocated satellite processor
        number(SP-id);
      else
        sleep until available satellite
        processor;
goto Loop;
} ;
release-SP( ) ;
{
arrange RSP table;
arrange SP table;
wake up appropriate user processes that wait
for satellite processor allocation;
if error
  return error value;
} ;
```

ii) Satellite processor kernel과의 通信 channel 形成

Satellite processor가 하드웨어的으로 I/O board를 通하여 시스템 버스(Bus)와 interface되어 있으므로, 그 I/O board를 管理하는 driver routine이 必要하다. 이런 I/O board는 基本的으로 tty interface와 同一하므로, 添加되는 driver routine은 tty driver routine과 많은 部分을 共有한다. 하지만 이 通信 channel을 管掌하게 되는 user process는 다른 tty channel을 access하는 user process보다 더 높은 우선순위를 가지게 하여 satellite processor의 要請에 對한 Host 시스템의 應答 時間을 短縮하였다.

5. Co-process

Co-process는 satellite processor kernel과의 協力을 通하여 다음과 같은 主要 動作을 遂行한다.

① Satellite processor로의 作業(test) 割當
② 그 作業이 要請하는, host 시스템이 行하여야 하는 動作에 對한 應答
③ 그 作業에 對한 制御

이같은 動作을 通하여 satellite processor로 割當된 作業(task)이 host 시스템에서 提供되는 여러 周邊 裝置들을 使用하며 遂行된다.

Co-process와 satellite processor kernel과의 協力은 서로간에 message를 交換함으로써 이루어지는데, 이러한 message 形態는 하드웨어 構造가 星相(star) 形態를 띠므로 매우 간단히 構成될 수 있다. 이러한 message 形態에 관하여는 3.3節에서 자세히 論議하기로 한다.

6. 通信(communication) message 形態

HP와 SP間의 데이터 傳送은 適當한 header 및 checksum code를 包含한 protocol을 使用하게 되는데 本 시스템에서는 모토롤라社가 開發한 S-record를 使用하였다.[6] 이 S-record는 傳送過程이 visible할 뿐 아니라 editing 및 decoding하기도 쉬우며 旣存의 시스템을 活用하기 쉽기 때문에 本 시스템에서도 이를 導入하였다. 이 S-record는 record type, record length, memory address, code/data 및 checksum 등의 여러 field로 構成된 character string인데, 各 byte는 2-character hexadecimal number로 encode되어 있으며 그 構成은 그림 6(a)와 같다.

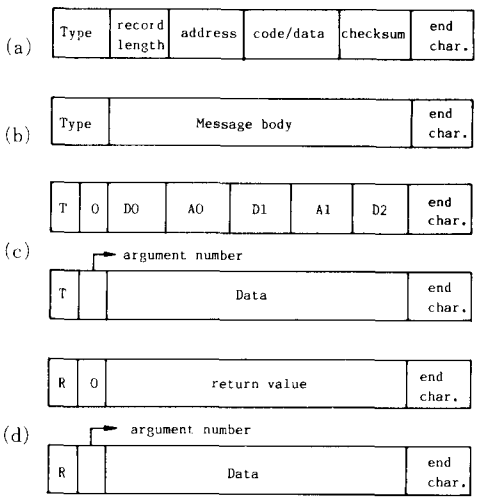

그림 6. (a) S-record protocol
(b) Message 傳達을 위한 通信 protocol
(c) T- record protocol
(d) R-record protocol

Fig. 6. (a) S-record protocol.
(b) Communication protocol for message.
(c) T-record protocol.
(d) R-record protocol.

System call message를 轉送할 때는 text program

의 境遇보다 轉送되어야 할 데이터量이 많지 않으므로 더 간단한 protocol을 使用했는데, (그림6(b) 참조)
UNIX 시스템에서 system call을 遂行하기 위한 情報로서는 system call종류를 구분할 수 있는 情報 및 各 system call 遂行에 必要한 argument 등을 포함하여야 하며(그림6(c) 참조, 여기에서 MC68000 CPU의 레지스터인 D0 가 system call 종류를 나타내는 情報를 포함하고 있으며 레지스터 A0, D1, A1, D2등이 각종 arguments등을 포함하게 된다.) 이런 情報를 包含한 system call message를 받으면 그에 대한 遂行을 한 후, 그 結果를 그림 6(d)와 같은 形態로 message를 構成하여 SP로 보내주게 된다.

7. Satellite kernel

Satellite kernel은 主 컴퓨터로 부터 task를 download하며 遂行시키는데 필요한 program이며 이 satellite kernel의 構成은 하드웨어 및 소프트웨어 인터럽트를 處理하는 IH(interrupt handler), HP와 SP간의 通信을 擔當하는 IPC(interprocessor comunication) handler, system call 요청을 擔當하는 SCH(system call handler), SP의 local mode로 user program을 debugging하며 修正하기 위한 debugging package 및 kernel program의 일부 subroutine을 user가 使用하기 便하게 만든 library package 등으로 構成되어 있으며 그 各各에 대한 說明은 다음과 같다.

i) IH(interrupt handler)

IH는 SP의 local device나 소프트웨어에 의한 asynchronous interrupt 요청을 現在 遂行中인 user task와는 관계없이 實時間으로 처리하기 위한 routine이다. IH의 주된 任務는 HP로부터의 text의 downloading, system call 요구에 대한 HP의 遂行結果인 reply message를 받는 일 및 control signal을 받는 일들을 確實히 해주기 위하여 IPC handler를 도와주게 된다. 물론 이러한 情報를 받는 對象은 HP뿐만아니라 local terminal 및 SP의 ABORT button도 이에 該當된다.

ii) IPC(interprocessor communication) handler

SP와 HP間의 通信을 擔當하는 routine으로서 이는 SP 및 HP 양쪽에 모두 implement되어 있으며 다음과 같은 機能을 가진 routine 등으로 構成된다.

• text program의 loading - SP에서 遂行시키고자 하는 text program을 loading하는 機能.

• 制御信號의 傳達 - 遂行始作信號, 遂行抛棄信號, loading 再試圖信號 및 memory 診斷信號등을 傳達하는 機能.

• 데이터 傳達 - system call request message 및 그 system call에 대한 結果인 reply message 등의 데이터를 傳達하는 機能.

이러한 機能들로 因해 主 컴퓨터는 SP에서 task를 遂行시키도록 制御할 수 있게 된다.

iii) SCH(system call handler)

User task가 system environment에 接近하기 위해 system call을 利用하게 되는데, 이러한 system call이 SP로 하여금 主 컴퓨터에 있는 파일 시스템에 接近하거나 I/O peripheral을 利用할 수 있게 한다.

現在 UNIX 시스템에서는 50餘 가지의 system call이 있으며 이들은 I/O에 關한 것, 프로세서에 關한것, system element에 關한 것 및 파일 시스템에 關한 것들이 있으나 本 시스템에서 現在 가장 關心의 對象이 되는 것은 主 컴퓨터의 파일 시스템에의 接近을 可能하게 하는 creat, open, read, write, close, exit 및 lseek등의 system call이며, 現在 이들이 구성되어 있다.

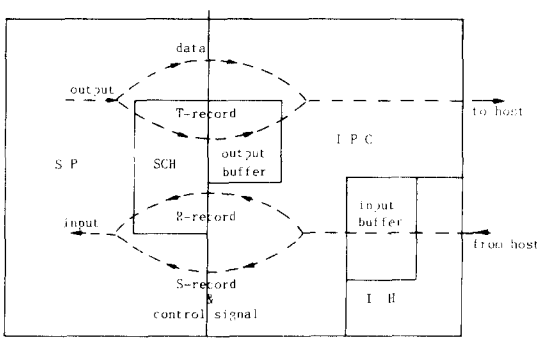

그림 7. 소프트웨어의 견지에서 본 data flow
Fig. 7. Date flow from the viewpoint of software.

Ⅲ. 結 論

工場 自動化를 爲한 컴퓨터 시스템 開發의 一環으로 host 시스템과 수개의 satellite processor로 이루어지는 主從(master/slave) 關係를 가지는 multi-CPU 시스템을 構成하였다. 이 시스템은 基本的으로 實際應用 分野의 作業形態와 一致하는 構造를 가지고 있으므로, 理想的으로는 — 어떤 故障도 없을 경우 — 工場 自動化 應用 分野에서 最適合한 構造라 할 수 있다. 그러나 實際로는 하드웨어가 直相 構造로 이루어져 있으므로 全體 시스템의 信賴度(reliability) 問題가 發生한다.

그리고, satellite processor 部分을 module化 함으

로써 全體 시스템의 擴張및 維持를 容易하게 할 수 있으며, host 시스템에서의 일(task)들을 satellite processor가 分擔하므로써 host 시스템의 process switching 같은 overhead를 減少시켜 全體 시스템 結果率(performance)를 向上시킬 수 있다.

하지만 앞서 조금 言及하였던 fault-tolerant 問題点은 host 시스템 部分, 즉 master 部分과 satellite processor 部分, 즉 slave 部分에서 各各 다른 形態를 띤다. 먼저 master 部分에서의 fault-to lerant 問題는 가장 간단히 duplication 方法으로써 克服될 수 있고, slave部分에서는 그 satellite processor가 遂行하는 일이 독특한(unique) 것이면 master 部分과 같이 duplication 形態로써, 一繫的인 일이 되는 경우엔 master 部分에서 slave 部分으로 dynamic하게 일을 割當시킬 수 있게 하면 解決될 수 있다. 또한 全體 시스템에서 master 部分이 主要 bottleneck으로 나타난다. 즉, 많은 satellite processor에서 要請하는 動作을 정해진 時間內에 master 部分이 遂行할 수 없게되는 데, 이런 問題는 master 部分에서 行해야 하는 動作들을 技能別로 여러 CPU로써 構成한다면 解決될 수 있다.

參 考 文 獻

[1] H. Lycklama et. al., *A Minicomputer Satellite Processor Systems.* BSTJ, pp. 2103-2113, 1978.

[2] A.B. Barak et. al., *UNIX with Satellite Processor,* Software Practice and Experience, pp. 383-392, 1980.

[3] B.C. Wonsiewicz et. al., *Microcomputer Control of Apparatus, Machinery, and Experiment.* BSTJ, pp.2209-2232, 1978.

[4] C.I. Park, *A Study on UNIX System with Satellite Processors.* KAIST, 1985.

[5] J.J. Kim, *A Study on the Satellite Processor for a Master - Slave Multiprocessor System.* KAIST, 1985.

[6] MC68000 Educational Board User's Manual, Motorola, 1982.

[7] D.A. Mellichamp, *Real-Time Computing, Van Nostrand Reinhold Co.*, 1983.