공정제어용 범용 Operator Interface 개발

°이재만, 김정훈, 채영도
한국 전자 통신 연구소

Development of a General Purpose Operator Interface for a Process Control System

Jae Man Lee, Cheong Hoon Kim, Young Do Chae
Electronics and Telecommunications Research Institute

Abstract

This paper describes the development of general purpose operator interface which uses a color graphics terminal with a touch-sensitive screen as the control console. Operators interact with a process through a collection of application-dependent displays generated interactively by users familiar with the physical process. The use of real-time operating system(iRMX-86) and multitasking results in a straightforward and reliable development which may easily be extended to support multiple devices of varying types in the control console.

1. 서 론

Physical process의 제어를 위한 operator interface는 operator에게 공정의 현 상태에 대한 정보를 제공해 주고 operator가 공정을 제어할 수 있도록 고안되어야 한다. 현존하는 operator interface로는 재래적인 hard-wired control room으로 부터 가장 최근에 개발된 컴퓨터 시스템과 연결된 compact한 control station까지 여러 계층의 다양한 형태가 있다. 기존의 hard-wired된 control room에서는 operator가 넓은 범위에 걸쳐 설치된 스위치, 버튼, 다이알, 그리고 meter들을 사용하여 physical process를 조작한다. 그후 컴퓨터를 기본으로 하는 제어시스템이 나옴에 따라 상태표시와 제어를 쉽게 할 수 있는 CRT터미날과 프린터가 제어실에 등장하게 되었다. 최근엔 공정전반에 걸쳐 compact하게 상호 작용하는 operator interface를 위해 color graphics 터미날이 널리 사용되고 있으며 operator는 변화하는 display를 통해 공정을 감시하고 keyboard, light pen, 또는 다른 입력 device를 이용하여 제어명령을 보낸다.

현재 있는 대부분의 operator interface는 single process 제어용으로 설계되었기 때문에 flexible하지 못하며, color graphics control console은 일반적으로 각각의 display와 command를 위해 특별한 목적을 가진 programming이 필요하다. 그러한 시스템을 요구사항의 변경을 처리하거나 다른 종류의 physcial process에 적용시키기 위해 수정하는 일은 매우 어렵고 비용이 많이 든다. Honeywell의 TDC 2000, Beckman의 MV 8000, 그리고 IBM의 Advanced Control System과 같은 종합 공정제어 시스템들은 약간의 확장으로 이 문제를 완화시켰다. 이 시스템들은 규모가 큰 공정제어용으로 개발되었으며 color graphics control console을 사용하지만 special purpose 시스템과는 달리 추가적인

programming이 필요없이 templates로부터 display를 구성할 수 있도록 해준다. 하지만 이것도 공정제어 시스템내에 embedded되며 특정 vendor의 control philosophy와 장비에 의해 제한된다.

본 연구는 이러한 문제를 해결하기위해 어떤 physical 공정에서도 사용할 수 있는 general purpose operator interface를 개발하였다. 이것은 공정제어 S/W와 operator interface S/W 사이에 process state를 나타내는 table과 명령을 전달하는 mailbox를 통으로써 physical 공정에 독립적인 범용성을 구현하였으며, 시스템에 color graphics monitor와 touch sensitive screen을 연결하여 operator가 physical process와 interactive하게 communication할 수 있도록 하였다. Graphics terminal의 display는 static background와 dynamic한 공정정보를 나타내는 object로 이루어지며 operator는 touch sensitive screen을 통해 current display를 바꾸거나 process control S/W에 command를 보낼 수 있다. General purpose operator interface S/W는 operation을 위해 memory를 공유할 수 있는 multitasking environment를 필요로 하며 시스템을 구성하는 task는 항상 memory에 존재하고 task가 사용하는 table과 communication mailbox는 shared memory에 저장된다. General purpose operator interface는 Structured Analysis and Design 기법을 사용하여 설계하였으며, iRMX 86을 O.S.로 사용하는 OEM system에서 C language로 개발하였다.

2. 설계 개요

요구 분석의 결과로 operator interface가 갖추어야 할 기능을 구현하기 위해 본 연구에서는 Yourdon사에서 제안한 설계기법을 따랐다. 아래 그림은 modeling 결과로 나온 DFD(Data Flow Diagram) 이다.

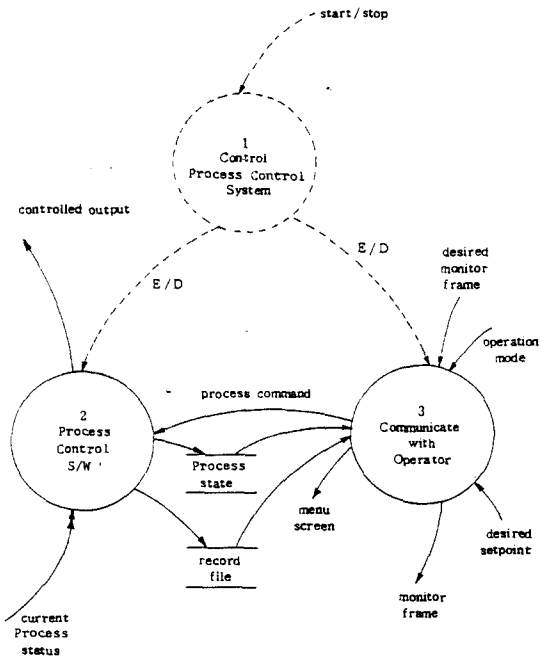

그림 1. 공정정보 처리 시스템의 DFD

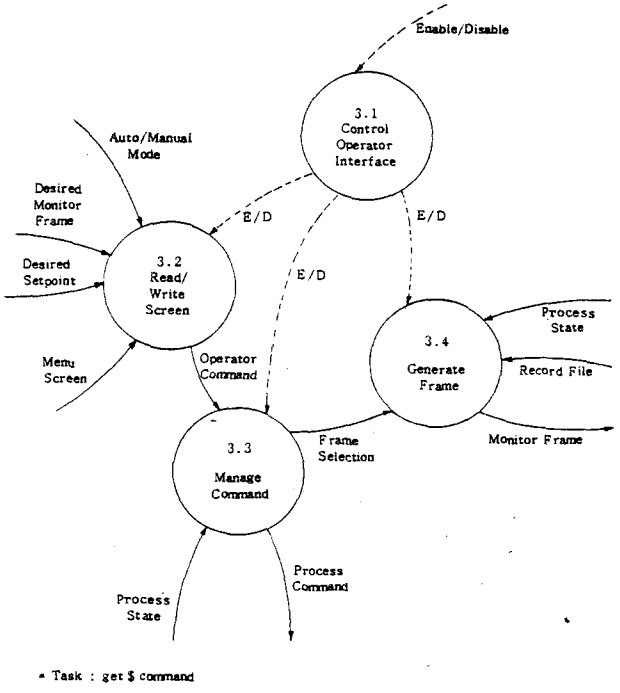

그림 2. Communicate with operator

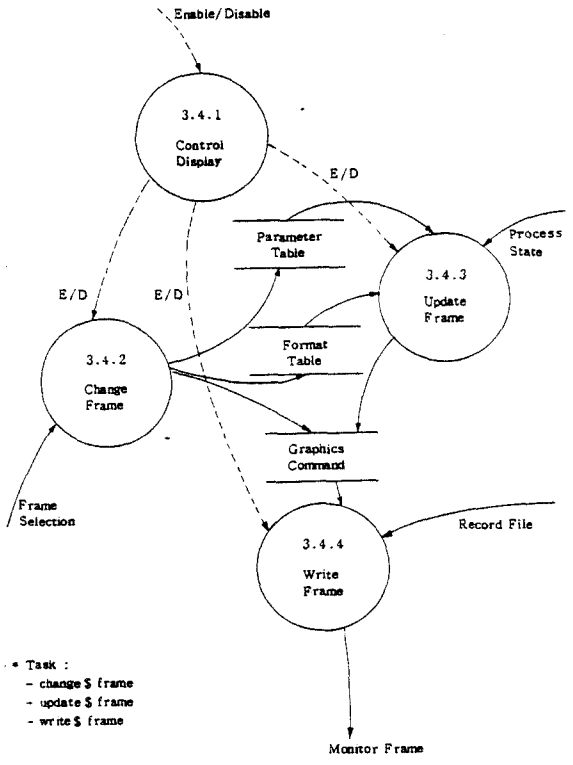

그림 3. Generate frame

Functional modeling을 한 결과 operator interface는 task, table, 그리고 mailbox라는 세개의 기본적인 system components로 구성하였다. Task들은 table 형태의 정보를 공유하고 mailbox를 통해 서로 message를 주고 받으며 concurrent하게 동작한다. Table은 shared memory내에 구축되며 임의의 어떤 시간에 한 task만이 table의 내용을 access할 수 있도록 task간에 mutual exclusion이 이루어져야 한다. 따라서 shared structure의 사용을 원하는 다른 task는 사용중인 task가 data structure의 access를 포기할 때까지 기다려야만 하고 task가 비어있는 mailbox로부터 message를 받으려고 하면 그 task는 다른 task가 mailbox에 message를 보낼 때까지 기다려야만 한다.

Operator Interface를 구성하는 task로는 get_command, change_frame, update_frame, 그리고 write_frame이 있다. Mailbox로는 process_command mailbox, display_command mailbox, 그리고 graphics_command mailbox가 있으며, table에는 state table, parameter table, 그리고 format table이 있다. Operator interface가 동작하기 위해선 operator interface와 공정제어 S/W사이의 상호 연결의 기본이 되는 state table과 process_command mailbox만 사용 가능하면 된다.

3. 구 현

Operator Interface는 4개의 task와 mailbox및 table 들로 구성된다. 각각의 specification은 다음과 같다.

3.1 Mailbox

가. Graphics_command

Graphics_command mailbox는 write_frame task로 가는 graphics command를 받아들이는 곳이다. Graphics command로는 BACK, TREND, LINE, BLOCK, STEXT, 그리고 PCIR이 있다. Change_frame task는 새로운 frame을 color monitor에 loading하기위해 BACK과 TREND command를 보내며, update_frame은 process parameter 정보를 화면에 update하기 위해 LINE, BLOCK, STEXT, 그리고 PCIR 등의 command를 보낸다.

나. Display_command

Display_command mailbox는 get_command task가 operator로 부터 frame변경에 관한 명령을 받아들여 그 명령을 보내는 곳이다. Display command의 종류로는 BACK과 TREND가 있다.

다. Process_command

Process_command mailbox는 operator interface가 공정제어 S/W와의 communication을 위한 창구로 쓰인다. Get_command task는 touch sensitive screen을 통해 받아들인 operator의 명령을 이 mailbox로 보낸다. Process

command로는 process start, stop, setpoint 변경, 그리고 auto/manual mode 변경이 있다.

3.2 Table 구성

가. State Table

State table은 update flag와 공정제어 S/W에서 결정되는 current process parameter의 값으로 이루어진다. Update flag는 공정제어 S/W가 parameter의 값을 바꿀때마다 set된다. Update_frame task는 공정의 현재 상태를 반영하기 위해 화면에서 parameter와 연관된 object를 update 시킨 것인지를 결정하는데 이 flag를 사용한다. Operator interface는 state table 내의 process parameter를 읽을 수만 있으며 항상 parameter table에 포함된 parameter directory를 통해 간접적으로 access한다. 이러한 간접적인 access방법은 operator interface를 state table layout의 변화 등으로부터 독립시켜 준다.

나. Parameter Table

Parameter table은 Local Parameter Table, Global Parameter Table, 그리고 parameter table들을 가리키는 Parameter Directory로 구성된다. Parameter table과 state table 사이의 관계는 아래 그림과 같다.

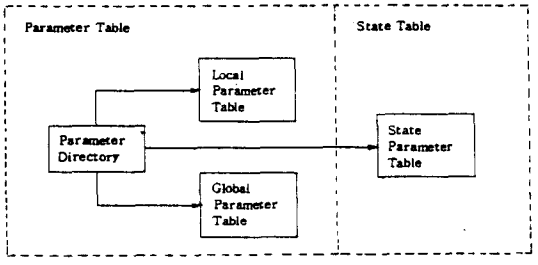

그림 4. Parameter와 state table의 연관 관계

Operator interface 내부에서 사용되는 parameter는 Local Parameter Table과 Global Parameter Table에 위치하게 되는데 각각의 구조는 state parameter table과 같다. Global Parameter Table에는 모든 frame에 공통으로 쓰이는 변수들이 들어 있고 Local Parameter Table에는 각 frame에서 자체적으로 쓰는 변수들이 존재한다. Operator interface는 항상 Parameter Directory를 통하여 위 변수들을 access한다. Parameter들은 update_frame task의 동작을 통해 화면에 여러가지 형태로 display된다.

다. Format Table

Format table은 Format Table Control Information, Object Directory, Object Number List, 그리고 Object Table로 구성된다. Object란 화면에 동적으로 나타나는 그림의 단위로서 현재 정의된 object의 종류로는 TIME, REAL, LEVEL, PIECIR, OPMODE, 그리고 ALARM이 있다. 각 object는 적어도 한 parameter와 연관되며 그 object가 화면의 어느곳에 위치하는지에 관한 정보를 포함한다. 또한 object의 update 여부를 나타내는 update flag, 색깔, mode, 그리고 attribute parameter number에 관한 정보도 Object Table에 포함된다. 다음 그림은 format table의 구조를 나타낸 것이다.

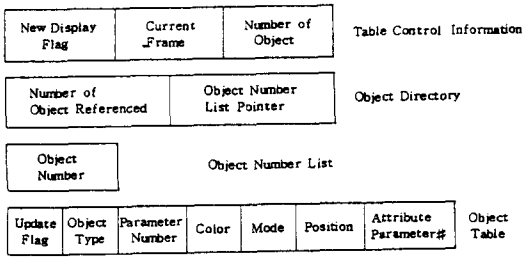

그림 5. Format table의 구조

Format table에서는 현재 display에 사용되는 각 variable parameter가 그 parameter를 사용한 object로 pointer를 통해 연결되어 있다. 따라서 한 parameter가 여러개의 다른 object에 기여하거나 여러 parameter가 한 object에 기여할 수도 있다. 이러한 pointer 들은 update_frame task가 화면의 object를 update 시키는데 사용한다. Update_frame task는 parameter가 update되면 그에 해당하는 object의

update flag를 setting하고, flag가 mark된 각 object에 대해서 write_frame task로 graphics command를 보내 그 object가 화면에 다시 그려지도록 한다. 또한 format table은 new display flag를 가지고 있는데, 이것 change_frame task에 의해 화면이 새로운 그림으로 바뀔때 setting된다. New display flag가 mark되어 있으면 update_frame task는 화면상의 모든 object를 update 시킨다. 다음 그림은 format table과 parameter table의 관계를 나타낸 것이다.

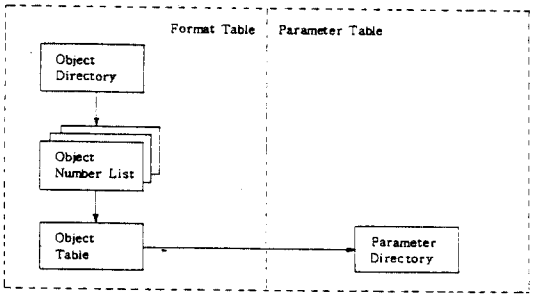

그림 6. Format table interconnection diagram

3.3 Task Specification

가. Get_command

Get_command task는 touch-sensitive screen으로 system operation에 대한 menu를 제공하고 operator가 입력하는 command를 sensing하여 operator가 쉽게 제어 시스템을 동작시킬 수 있도록 해준다. Operator가 입력시킬 수 있는 command로는 공정제어 S/W에 보내는 것으로 process start, stop, mode(start-up, shutdown, auto/manual) 변경, 그리고 setpoint 변경이 있으며, change_frame task로 보내는 것으로는 화면선택과 시간 setting 등이 있다. 공정제어에 관한 명령은 process_command mailbox에 보내고, 화면선택에 관한 명령은 display_command mailbox로 보낸다.

나. Change_frame

Change_frame task는 display_command mailbox를 통해 get_command task로 부터 background와 trend command를 받아들인다. Command message가 도착하면 먼저 format table의 new_display flag를 setting하고 frame의 고유번호를 current_frame에 기록한다. 그 다음 Object Table에 있는 모든 object의 update flag를 setting하고 write_frame task로 background 또는 trend command를 보낸다. Wait 상태에서 command message를 기다리다가 display_command mailbox에 message가 도착하면 깨어나서 위의 일을 다시 수행한다.

다. Update_frame

Update_frame task는 5초 간격으로 주기적으로 반복 수행되며 하는 일은 다음과 같다. Parameter Directory를 통해 process state parameter, global parameter, 그리고 local parameter를 scan하여 update된 parameter가 있으면 그에 해당되는 object의 update flag를 setting한다. Object Table에서 update flag가 mark된 object에 대해 object type 별로 다음과 같은 기능을 수행한다.

1)TIME: 시간을 관장하는 task가 분단위로 시간을 갱신시켜 놓으면 그 결과를 text command를 사용하여 메세지를 write_frame task로 보내 화면에 항상 현재시간이 표시되도록 한다.

2)REAL: Parameter table의 parameter 값이 update되면 그 값을 ascii string으로 바꿔서 write_frame task로 text command를 보내 화면에 새로운 값이 표시되도록 한다.

3)LEVEL: 막대 그래프나 tank의 수위표시 등에 사용되는 object이다. 먼저 현재의 level을 나타내는 parameter와 full 상태에 해당하는 값 그리고 전체 block의 위치로 부터 원하는 level의 높이를 계산해 낸다. 현재의 값을 pvalue, full일 때의 값을 fvalue, 전체 block의 좌표를 (x1, y1) (x2, y2)라 하면 원하는 값 Y는

Y = (pvalue/fvalue) * (y2 - y1) + y1

이다. 그 다음 (x1, Y) (x2, y2)에 해당하는

block을 지우도록 write_frame task에 명령을 보내고, (x1, y1) (x2, Y)에 원하는 색을 칠하는 명령을 보낸다. 이때 사용되는 graphics command 이름은 BLOCK이며 color가 0이면 지워지고 1부터 7까지는 각각 해당되는 색이 display된다.

4)PIECIR: Process parameter의 퍼센트 값을 화면에 pie circle로 나타낸다. PCIR graphics command를 사용한다.

5)OPMODE: Operation mode의 변화에 따라 현재의 mode가 화면에 표시되도록 한다. Mode의 종류에는 process start-up, control, shutdown, 그리고 control auto/manual이 있다.

6)ALARM: Alarm이 발생하면 화면에 ALARM이라는 자막이 빨간색 reverse mode로 display되도록 하고 어떤 종류의 alarm이 발생했는지를 나타내 준다.

라. Write_frame

Write_frame task는 graphics_command mailbox를 통해 화면에 display할 내용에 관한 명령을 받아들인다. Command message가 도착하면 wait 상태에서 깨어나 해당하는 그림을 화면에 그리고 다시 wait 상태가 된다. Graphics command의 종류와 argument 및 기능은 다음과 같다.

1)BACK: 화면의 고유번호에 따라 새로운 화면의 밑그림을 그린다.

2)TREND: trend의 고유번호와 시간 간격을 받아서 process의 trend를 그린다.

3)STEXT: 화면상의 위치와 색깔, 그리고 string pointer를 받아서 text를 화면에 써 넣는다.

4)BLOCK: Block의 위치와 색깔에 따라 화면에 해당되는 부분을 그린다.

5)PCIR: 퍼센트 값, 위치, 그리고 색깔에 따라 pie circle을 그린다.

4. 결 론

본 연구에서 개발된 범용 operator interface를 보일러 시뮬레이터에 적용시킨 결과 보일러 상태의 monitoring에 매우 우수한 성능을 확인하였다. 보일러 제어의 각 동작 mode에서 그 mode를 monitoring 하기에 가장 적합한 그림을 선택하여 color monitor를 관찰하면 보일러에서 sensing한 파라메타 들과 control output 값이 real-time으로 display 됨을 볼 수 있었다. 또한 범용으로 개발되었기 때문에 보일러가 아닌 다른 공정에 적용하여도 table의 값만 바꿔주면 operator interface는 그대로 사용할 수 있으며 graphics 입출력 장치도 필요에 따라 다른것으로 바꿀 수 있다.

현재 operator interface가 가지고 있는 reporting 기능은 monitor frame의 hard copy가 있으며 앞으로 printer를 이용한 reporting 기능이 추가될 예정이다. 그리고 operator가 좀더 쉽게 공정을 제어할 수 있도록 기능을 보완하고 분산제어 시스템에도 적용될 수 있도록 하는것이 앞으로의 연구과제이다.

참고 문헌

[1] Paul T. Ward, Stephen J. Mellor, Structured Development for Real-Time Systems, Volume 1, 2, Yourdon Press, 1985.

[2] Neil B. Corrigan, J. Denbigh Starkey, "A Concurrent General Purpose Operator Interface," IEEE Trans. Software Eng., Vol. SE-10, No. 6, pp.738-748, Nov., 1984.

[3] J. A. Calvo and R. T. Jones, "A more efficient control room operator-process interface," IEEE Trans. Power App. Syst., pp.2509-2516, Nov.-Dec. 1971.

[4] A. Uyetani, K. Yoshizaki, and K. Nagakawa, "Standardized operator interfaces for distributed control systems," Instrum. Technol., pp.43-47, Dec. 1978.

[5] R. Dallimonti, "Human factors in control center design," Instrum. Technol., pp.39-44, May 1976.

[6] iRMX86 Programming Technigues, Intel Corp., 1982.

* 본 연구는 과기처 특정연구 개발사업의 결과임.