ISBN: 978-3-95450-254-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS1991-S08NC09

A Program Development Tool for KEK VME-MAP Control System

H.Nakagawa, T.Katoh, E.Kadokura, A.Akiyama,
K.Nigorikawa and K.Ishii
KEK National Laboratory for High Energy Physics

Abstract

The control system for KEK 12 GeV Proton Synchrotron has been replaced with a distributed VME-bus based microcomputer system and a MAP local area network. In order to simplify programming for network application tasks, a set of a preprocessor for a PASCAL compiler and a network communication server has been developed.

Application programs for accelerator control system have blocks with similar codes; sending, waiting for, receiving, analyzing messages, etc. The preprocessor called "OBJP" incorporates such common codes into the source code written by an application programmer.

In case of a simple program, the size of the source code is reduced by one tenth of a full coding.

I Introduction

The present control system for the KEK 12GeV PS has been modified by using VME-bus based computers and MAP local area network. On these computers, application tasks work under the VERSAdos; real-time multi-tasking operating systems. In this case, tasks in one group run on some computers and they communicate with each other by network. Then, most important factor of such programs is that any programmer can write the communication function of applications easily. By reason of these thoughts, the network support programming tool 'OBJP' has been created.

The 'OBJP' is preprocessor of PASCAL compiler, but the source file of OBJP programming seems to be a new language system like PASCAL.[1] And it also seems like the object oriented programming, but 'OBJP' is not complete object oriented programming. So the 'OBJP' is a communicatable multi-tasking program development tool. But we think that both the 'OBJP' applications and object oriented programming found on a same basic idea; each function is isolated and communicate with each other by message.

Let's show the 'OBJP' programming, and the configuration of the VME-MAP computer system for the 12GeV PS control.

II System configuration

This control system consists of two major devices; 26 VME-bus computers and a MAP local area network.[2] All the VME-computers are linked together in the same level. Their physical connection is bus style, but logical connection is ring style by token-passing. All computers are equal to each other in the logical ring.

Each computer dedicates to each specialized function. There are computers of one network manager, one program development, four console and many device controllers as shown in fig.1.

The network manager is "UNIX", which checks computer condition and makes logging of its information. The computer "ROLA" offers multi-user programming environment, and sends applications to other computers at rise-up of the computer. The console computer works for human interface. The device controllers control the power-supplies and monitor their status.

Video information signals are also transmitted through the MAP network cable.

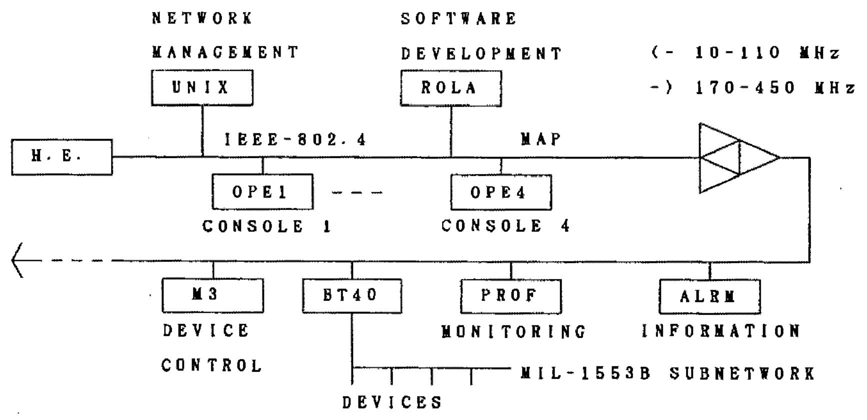

Fig.1 A scheme of MAP-VME control system.
The computers are connected with the MAP local area network.
The head end remodulator(H.E.) repeats signaling from the reverse channel on the forward channel.
Each box shows a computer and ID named after the function
There are some broad-band amplifiers.

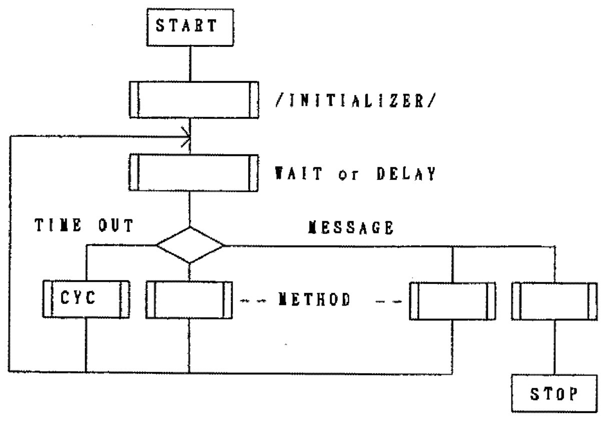

Fig.2 Typical flow chart of resident tasks for control. Similar codes for initialize part, wait for events and event select appear in many programs.

III "OBJP" programming

The OBJP preprocessor program has been created as a result of a research of program design of multi-task system. We notice the appearance of similar codes in many programs.

Fig.2 shows that a flow chart of a program which runs on multi-task OS and residents on memories. In the chart, both "WAIT or DELAY" part and message select part are required from all our tasks. And almost all of our tasks must send messages.

Our programmers are not proper in programming, then good programming environment is required. So, we avoid the method by editor, so that the method by means of automatical formation has been chosen. For this purpose, the preprocessor called "OBJP" has been developed. The OBJP makes a PASCAL source as shown in fig.3.

Simple program of OBJP source is shown in fig.4, . It works really in our system for relay-board control. This program is simpler than one in which all possible items relating to the network communication functions are written.

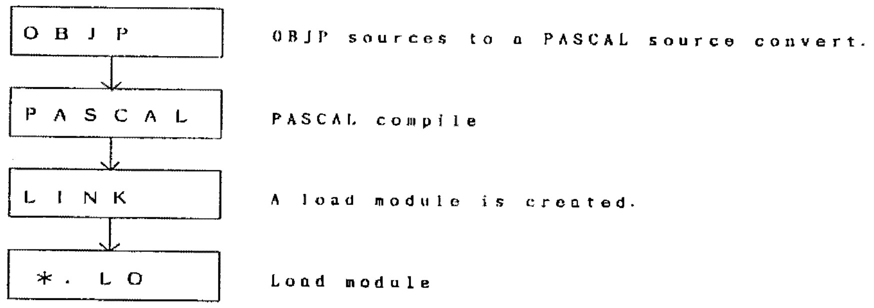
O B J P
OBJP sources to a PASCAL source convert.
P A S C A L
PASCAL compile
L I N K
A load module is created.
*. L O
Load module

Fig.3 Program development flow.
The OBJP is a preprocessor of PASCAL compiler.

```
SWITCH = W
   ( Graphics and Delayw       )

program SOUNDPRO ( Objective pascal )  ;

/COMMON/
const
TASK    = 'WAAA'       ;
TDELAY  = 000          ;  ( msec )

/LOCAL/
const
   DDTT  = 500    ;  ( msec )
   DMADA= 0500    ;  ( msec )
   DELAY = 21     ;  ( Delay for RMS )

type
   MSDATA = string[ 255 ] ;

var
   ADDR [ origin 16#DFF502 ] : word ;
   ADDS [ origin 16#DFF500 ] : word ;
   I : integer          ;
   J : integer          ;
   K : integer          ;
   L : integer          ;
   N : integer          ;
   ERROR : integer      ;
   MM : MSDATA          ;
   S_NAME : CCCC        ;

/EXTERNAL/
0.OBJP.MTOBIN.TX

/INITIALIZER/
L := 16#DFF400       ;
N := 16#400          ;
S_NAME := 'ROUT'     ;
```

```
ERROR := MakeMMIO( S_NAME , L , N )    ;

/METHOD/
TERM:
          1:DCLK <:= 'MESS TERM WAAA' ;

OHHI:
          MM := MESSAGE          ;
          I  := MTOBIN ( MM ) ;
          J  := MTOBIN ( MM ) ;
          K  := 1                ;
          if I <> 0 then
              for N := 1 to I do
                  K := K * 2      ;

          for L := 1 to J do
            begin
            ADDR  := K          ;    ( relay on )
            ERROR := TRAP1X(DELAY,DDTT);(wait )
            ADDR  := 0          ;    ( relay off)
            ERROR := TRAP1X(DELAY,DMADA);(wait)
            end                 ;

REST:
          MM := MESSAGE          ;
          I  := MTOBIN ( MM ) ;
          K  := 1                ;
          if I <> 0 then
              for N := 1 to I do
                  K := K * 2      ;

         ADDS  := K          ;    ( relay on )
         ERROR := TRAP1X(DELAY,DDTT);(wait )
         ADDS  := 0          ;    ( relay off)
         ERROR := TRAP1X(DELAY,DMADA);(wait)

/END/
```

Fig.4 Sample of a OBJP source is shown.
A program for driving a relay-board is shown as simple example.

IV Servers

The simple description about the OBJP applications is supported by communication server tasks. They work on all computers and communicate with each other. Simplified diagram is shown in fig.5.

The task '&SRV' receives the request for message send, and it passes the message to task 'C37X' which controls communication board. The task 'NANN' is an agency for system call on remote node. Then a message from an application is placed in a receive task's communication buffer. There are more servers for network management or file manipulation.

At OBJP level programming, any programmers do not have to know the server's functions.

V Standard human interface

Touch screens and CRT displays are used for the human interface of the accelerator control. The touch screen system has been managed by a special task called 'GPAN'. Application programmer need not write a part of program for controlling the touch screen.

All commands for the control has been written in files for the GPAN. When the surface of screens is touched, the GPAN sends a message to a task of either device control or information display. Scheme of this relation is shown in fig.6. Data structure of the input file is standardized as shown in fig.7.

When the GPAN is used, programmers need not write command request function of

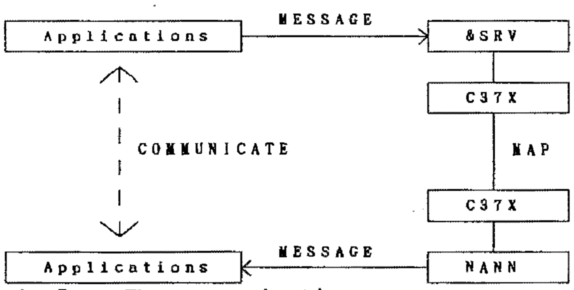

Fig.5 The communication servers.
Applications which is made by the OBJP communicate with each other through communication servers.
When MAP protocol requires connections, the applications do not have connections. The server only has ability of the connection.

programs. That simplifies programming.

VI Typical application task

When the GPAN sends a message to a task, the message brings about cascade shower of messages among some tasks. A sample is shown in fig.8, where many tasks work for controlling a beam slow extraction of both EP1 line and EP2 line. Each task has only one function, so that it is very simple.

In this case, the one task corresponds to one device similar to an object of the object oriented programming. Because one device replacement corresponds to one task

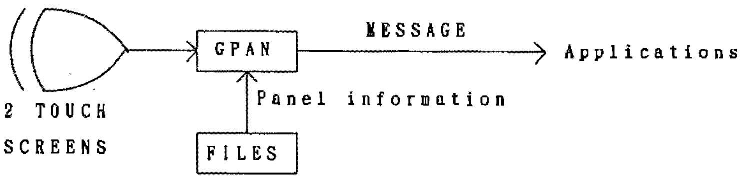

Fig.6 Touch screen management is done by special task, called 'GPAN'. The 'GPAN' reads "Control Information Files" and rearranges buttons of the touch screen.
If the button displayed on CRT is tòuched, the 'GPAN' sends a message to the target task.

```
type

 button = record
  number  : word;         { Button location XY  }
  corrv   : word;         { Button change style }
  ccodep  : word;         { Color code : normal }
  ccoder  : word;         { Color code : pushed }
  infop   : string[27];   { Letter : normal     }
  infor   : string[27];   { Letter : pushed     }
  taskname: string[8];    { Receive task name   }
  commmess: string[30];   { Message 1           }
  tasknamr: string[8];    { Receive task name   }
  commmesr: string[30];   { Message 2           }
  keykey  : word;         { Keybord or 2nd panel}
  nextssf : string[8];    { Next screen         }
 end;
```

Fig.7 A control data structure for our standard touch screen.
Data size per a touch button is about 200 bytes. The data are edited by a program for the exclusive use of data making.

replacement, good program maintainability can be kept.

The sample is complex case. One program is send to two computers for both EP1 and EP2. But in many case, a pair of tasks works one job; one task controls devices or takes data and another task displays the status on a CRT display. The structure depends on a programmer's conception.

VII Sample of remaking

As hardwares of accelerators are replaced often, control program for a replaced hardware must be also replaced. For example a sound information unit has been replaced recently, as it is hopeful that useful messages for operators are announced. The step of the replacement is shown in fig.9.

In this case, there are 3 stages. First, a relay-board on VME-bus drives talker units which can talk only one message of 8 seconds. It is very simple, but not flexible.

We introduce a new alarm message system. We have made the talker of both a D/A board and its drive program. As the source program of OBJP is simple and many functions does not appear in the source codes, the simple modification has been made.

The third stage in fig.9 is on testing now. It is a logic table based information system. We think that it is similar to knowledge base system. This program calculates input states relation and it sends a message when the condition becomes equal states written in its table. In this case, the new talker system can use without modification when numbers of information are increased.

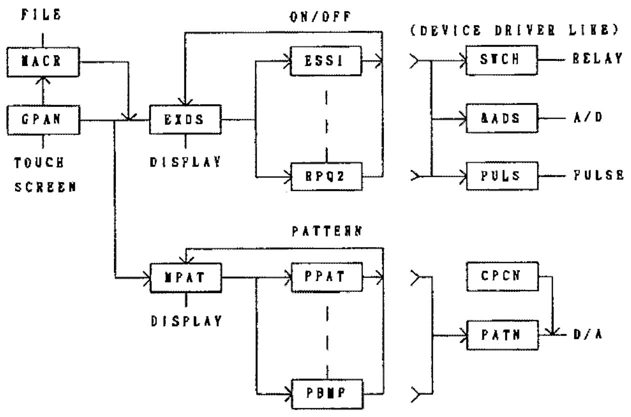

Fig.8 The OBJP applications work communicating with many tasks. This is a sample of beam extraction control system. Each box means a task with a special duty. These tasks work on some computers.

VIII Conclusion

In order to make up the new system by a few persons , a simple programming method is required to make many programs. We have made the OBJP for simple programming as the result of research of multi-tasking system.

In case of making program with "small and many" tasks, the one task corresponds to one device, so that it is similar to an object of the object oriented programming. But this programming method is not so easy for many programmers. They makes complex programs which have all functions in one task. "The all in one type programming" seems to be high performance, but debugging process is very complex. The "simple and many" program's debugging is simple, because the function of a task is simple. Total performance of the "simple and many style programming" becomes higher than the "all in one type programming", because of the simple debugging and its portability to other jobs.

IX Acknowledgement

The authors wish to acknowledge the continual and helpful cooperation of the 12GeV PS members. We wish to thank Dr. T.Kawakubo for his useful programs and his suggestions for debugging of the OBJP .

X References

[1] H.Nakagawa, "OBJP," KEK Report 90-20, Feb. 1991, (in Japanese).

[2] T.Katoh, H.Nakagawa, K.Ishii and E.Kadokura, "VME-COMPUTER BASED CONTROL SYSTEM FOR THE KEK PROTON SYNCHROTRON," Europhysics Conference on Control Systems for Experimental Physics, Villas-sur-Ollon, Switzerland, 1987, and KEK Preprint 87-91.

```
@ 1ST STAGE

INPUT --> RELAY DRIVE PROGRAM --> RELAY --> BUZZER OR
                                            FIXED TALK UNIT
                    |
                    | A FEW MINUTES
@ 2ND STAGE         |
                    V
INPUT --> BIT TO TERMS TRANSLATOR --> TALK PROGRAM --> D/A
                                           |
                                           | SAME PROGRAM
@ 3RD STAGE                                |
                                           V
INPUT --> INFORMATION BY LOGIC TABLE --> TALK PROGRAM --> D/A
                              |
                              |--> INFORMATION DISPLAY --> DISPLAY
                              |
                               --> LOGGING PROGRAM --> PAPER OR FILE
```

Fig.9 A sample of task replacement.
A new talk information task for alarm announcements is introduced in the control system.
Stage 1 : fixed talker units were used for sound information.
Stage 2 : a new talker has been introduced into the system.
The relay drive program has been modified for new system.
Stage 3 : a new information system(now testing).
The program is like a core of any knowledge base system.
The new talk task can be used without modification.