SPECIFICATION, DESIGN AND MODULA-2 IMPLEMENTATION OF A LOW COST INDUSTRIAL CONTROL SYSTEM

D. Garcia, H. Lopez, J. Tuya and A. Diez

Department of Electrical, Electronic, Computers and Systems Engineering, University of Oviedo, Carretera de Castiello s/n, 33204-Gijón, Spain

Abstract. The purpose of this paper is to show the application of modern software engineering techniques to the construction of a small/medium size package for real time control. This package, denoted by SCM (acronym of "Sistema de Control Modular"), has been written in Modula-2 and presents all advantages offered by this high-level structured programming language. It has been implemented on a PC-AT compatible computer.

Keywords. Software Engineering; Real time control; Modula-2 language; Concurrency; Computer-aided training

INTRODUCTION

Real-time control of industrial processes, specially in a research environment, requires the use of a software that meets some special characteristics such as modularity, flexibility, robustness, etc. Additionally, it is fundamental to have an open package capable of being extended. Consequently, it becomes rather difficult to find an adequate software package commercially available, providing free access to control variables in real time by their symbolic name, a friendly user inteface and several simulation and control structures.

This paper describes the most important features of the control package SCM, "Sistema de Control Modular" (Modular Control System) for process control in real time. The package has been totally developed in The Electrical, Electronic, Computers and Systems Department of Oviedo University and implemented in Modula-2 on a PC-AT compatible.

SPECIFICATIONS PHASE

Real Time Specifications

In addition to the basic programming requirements of all real time systems (readability, maintainability, simplicity, portability), it is important to indicate some specific characteristics of the SCM system as a real time control system:

1) A software organization in multiprocess is essential in order to describe real time systems. Then, the language must offer constructions to describe, create, intercomunicate, etc, concurrent processes.

2) Direct low level devices programming in order to control I/O non standard devices (A/D-D/A converters, I/O digital lines, etc).

3) To obtain a very high standard of reliability, the real time software must be able to recover error conditions. This reaction must be capable of restoring the system to a consistent state, repair any damage, and continue running normally.

4) In other applications, efficiency is to some extent subordinated to flexibility. However, in real time systems the former is very important, because the objetive is to guarantee the time response of the system whenever a change occurs in the controlled plant.

Specification of SCM-System Functions

In this section, the applications which have been taken as prior ojetives to be verified by the SCM system are defined:

1) Real time industrial processes control.

2) Investigation and development of new control and supervision methods for industrial processes.

3) Personal training for using real time control systems.

4) Obtaining of a cheap and trustworthy product, capable of enlargement.

Next, the user and process interfaces are specified in order to reach the previous objetives.

The User Interface

The user interface is a very important part of a real time control system. It is interesting to subdivide funcionally this interface in two modes, namely, "menu mode" and "command mode".
Subsequently, we define the facilities that both modes must incoporate for the interaction with the user.

Menu mode. In this operating mode, the interface helps the user (by menus,

indicating default values, etc) to introduce the basic structure and properties of the controller and other operative details previous to the control of the process. As regards this, its facility to be used is very important.

Once the user has entered the data, an initialization data base is created. This must contain the following values:

- Controller(s) parameters
- Estimator parameters
- Controller on/off
- Indentification on/off
- Process inputs/outputs
- Parameters for numeric or graphic representations
- Limits and actual values of the variables
- Refresh ratio of numeric and graphic representations

The introduction of data by the user is simplified with several techniques:

1) The most important technique is the utilization of a tree structure consisting on a main menu and several submenus for selection, visualization and modification of some parameters. The options are selected with the cursor movement keys. In fig. 1, the options of the main menu are shown.

2) When the user starts the SCM system all parameters take fixed default values. These values are modified in order to agree with the configuration file. This file preserves all parameter modifications. Then the operator can often configurate the system with a few modifications.

   Furthermore, the terminal shows information about the current state of the variables. The user can select with the cursor keys the desired item and with just one strike of a key, he can deposit the default value contained in the variable. Later, the cursor is moved to the next item.

3) Numerical data introduced are checked. If they are not correct, the user will be required to re-entry the information. Each entry can be modified in free format.

4) Several helps are incorporated to each option. This simplifies data introduction for the inexpert users.

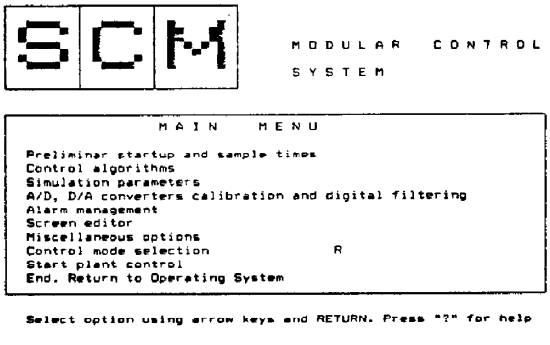

Fig. 1. Main menu of the SCM system.

The main menu is presented after the system starts-up. It shows the following options:

1) Sample times and variable dimensions. Intersample times for digital filtering must be specified too.

2) Control algorithms. The algorithm used in the real time phase is selected and initialized.

3) Plant simulators. It allows to initialize the parameters of several simulation structures.

4) Converter calibration. For each input and oputput, the limits in the real process variables and digital filters, can be specified. The calibration can also be carried out in order to adjust the reference levels (zero and full scale) in the converter, sensors and actuators.

5) Alarms. Basically, it consist of a list of variables which have to be checked each sampling in order to maintain them within the desired limits. The cause of the activation of an alarm can be indicated here i.e. crossing a threshold, occurrence number, etc. The action produced by the alarm (screen mesagge, alarm beep, etc) is also indicated. With that purpose, some of the output digital channels can be assigned.

6) Screen editor. Prior to the real time running, the screen interface between process and user will be defined. In these screens, the user will be able to see the values of the variables in real time, as well as the trend curves and histograms. See fig 2.

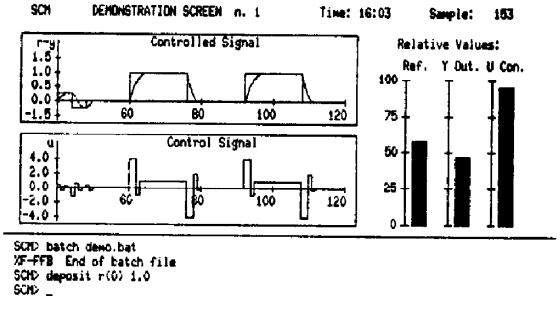

Fig. 2. Graphic interface during control.

All variable disposition, windows, screens, graphics, etc, is configurable for the user.

- Windows: the screen consists of a main window. Within it, other windows can be defined in any position and with any size.

- Variables: Any position of a window can be reserved to show the present value of one variable. This value will be modified in real time in conjuntion with the graphic

representations.

- Dialogue zone: It will contain a reduced number of lines reserved to the command interpreter in real time.

Command mode. After the inicialization of all necessary parameters in the previous mode, the system goes to the command mode.

This mode consists, basically, of a command interpreter. It allows the user to change in real time (during the execution and without interrupting the normal control cycle) any parameter specified in the previous mode. The commands accepted by the interpreter have the following format:

command [subcom] [param 1, param 2, ...]

The command is introduced in response to prompt SCM>

```
SCM> DEPOSIT R(1) 5.0
```

The previous order is equivalent to

```
SCM> DEPOSIT
_Variable name: R(1)
_Value: 5.0
```

This allows to assist non-experimented users in the system parameter introduction. The introduction of commands is simplified through a set of techniques:

- Any non-ambiguous short form of the command is accepted by the interpreter.

- The incorporation of the "ayuda" (help) command, which shows the actuation possibilities of the system to the user.

- The new parameter values introduced by the user are only accepted if they are within the prefixed limits for them in the menu mode and the variable table.

- Dump of messages when an error happens indicating its cause. The system has a facility for doing reports of all commands processed. In this way, when the system fails, the cause is easily investigated.

  Every subcommand and parameter is checked in agreement with the sintaxis defined for it. A sequence of inputs could be:

```
SCM> DEPOSIT R 5.0
%V-VD1 - The dimension of variable is
1 "R"
SCM> DEPOSIT R(5)
%V-VFR Index of variable out of range
"R(5)"
SCM> DEP R(1)
_Value: 5.0
SCM>
```

- Input and output can be easily redirectioned (to terminal, file, etc). Therefore, all inputs to the interpreter can be recorded in a file which can later be run in batch mode. Furthermore, the standard error and alarm outputs are completely differentiated. This allows, for example, to send the alarm outputs to file and terminal at the same time, in order to generate reports.

- Operating in batch mode, with execution of the commands contained in a file. The execution instant of each command can be specified in an absolute form (time measured from beginning) or relative (time measured from the execution of the last command). With that purpose, some additional commands are included, whose use is not allowed in interactive mode. An other command is "verificacion" (verification). It allows the visualization by the terminal of every command accepted by the batch processor.

- It is possible to change and deposit values in the variables, with allowed access. The input will be carried out with the complete name of the variable (and its index if it is a vector or matrix). SCM allows variables of up to three dimensions can be easily interpreted as polynomial matrices, etc.

- Finally, we shall remark the data adquisition commands, input-output redirectioning, initialization of graphic mode as well as those used for saving and restoring the state of the full control data base.

The Process Interface

Some other digital auxiliary channel is desirable, in addition to several analog I/O channels, for the link between the system and the plant. These auxiliary channels can be used to handle on/off actuators, external alarm signals activation, etc.

Furthermore, the conexion software with such I/O devices, has to be hierarchically structured in several levels. Then, it will be easy to change the existing hardware, modifying the minimum amount of code and always in the lower level modules.

In addition to the real time control of a plant, the SCM system offers the following possibilities:

- Control, in real or scaled time of a simulated plant by an analog computer.

- Control of a simulated plant in another digital computer in real or scaled time.

- Control of a linear process simulated internally in the package.

- Control of a complex mathematical model, external to the package and linked to it.

When the package is working with simulated plants it is possible to emulate the AD/DA converters with a variable word lenght. This, in addition to the utilization of the "Decimals" module (for

decimal aritmethic), allows an easy simulation of the problems originated by the quantization and round-off phenomenous.

Additionally, antialiasing filters will be included in each AD converter. Spike-filters are also supplied, both in the control of a real or a simulated plant.

DESIGN PHASE

Design techniques for real time systems

Once the specitication set has been defined, it is necessary to establish the system structure and its modules. The use of an adequate high level language is useful in this phase.

In this way, the design will be made in a top-down mode, by consecutive debugging. Based on the desired system features, a model was defined in the specification phase. This model will be progressively detailed in the design phase in the form of modules, the interface between them, the functions assigned to each of them and the algorithms incorporated in these modules.

Finally, another usual technique for the design of this type of systems is the use of redundant constructions. This will allow the detection of most of the errors during the compilation phase, and the rest, in the execution phase.

Structure of the Control System

The solution adopted for the SCM system is the typical two task structure (background and foreground). Each of these tasks is composed of several modules, which are procedures that run sequentially. It can be seen in fig. 3.

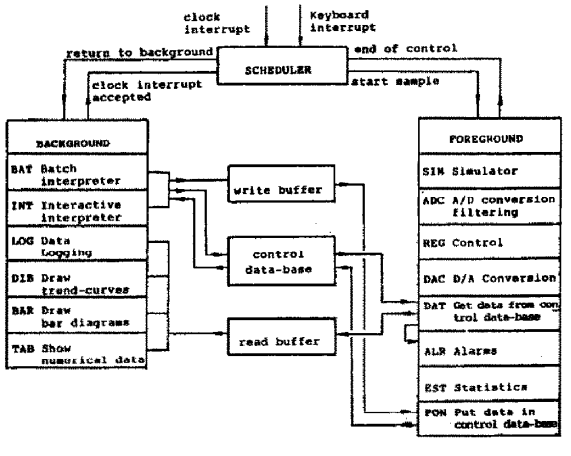

Fig. 3. Control system structure.

Using Modula-2, both the background and the foreground are declared as coroutines by the standard sentence NEWPROCESS. There is a third task, the scheduler, installed by the IOTRANSFER sentence as a clock interrupt service routine.

The control flow can be seen in the structure diagram. After the scheduler has been installed, it gives up the control to background task. In response to the clock interrupt, the control is transferred to the foreground. When this finishes, the background takes over control again.

The modules incorporated to the SCM system can be classified in the following groups:

Control modules. The number and characteristics of the control modules are continuously growing through the implementation of several algorithms by the investigators who are using the package. These algorithms are implemented in modules with structural dependence on the REG process. The following are included as standard algorithms:

- Independent PID regulators. They can be calculated by optimization techniques for the control of several loops or as back-up regulator in case of estimation failure, in adaptive control.
- Astrom-Wittenmark structure for regulators calculated by pole assignment.
- Adaptive and Self-Tuning controlers.

Inferface modules. These modules allow the communication between the different tasks. The data flow depends on the structure used, grouped around a data base. It contains the control variables block and two intermediate buffers for data transfer between background and foreground. This transfer is realized in different forms:

- The interpreters, using the adequate commands, can read and write directly in the control data base.
- The interpreters can write in the intermediate writing buffer too. This is used in order to change a determined number of variables simultaneously (PON process).
- The exhibition and data adquisition procedures read these from the intermediate reading buffer. The data have been transferred to it from the data base control with the DAT process. In this way a simultaneous access to all desired variables is guaranteed after a sampling and the associated control action.

Handling devices modules. These modules control the devices involved in inteface operations with the process (ADC for the A/D converter, DAC for the D/A one) or with the user (drivers for keyboard, video, printers, etc).

Utility modules. These modules are related to drawing and graphic exhibitions (DIB, TAB and BAR), data adquisition (LOG), alarms (ALR) and statistics (EST).

Simulation processes modules. The package has several options which allow the simulation of all types of plants, both linear and non-linear. This utility is

very often used when the package acts as a trainer, a didactic tool or in the development and implementation phases of the new control algorithms. The whole set of limits and dimensions of control variables can be changed for customized versions modifying the definition module "ControlCom.def" and recompiling the imported modules. The different options are:

1) Discrete time linear SISO or MIMO model: this can be originated from identification or discretization of a previous continuous model. A help screen can be seen in fig. 4.

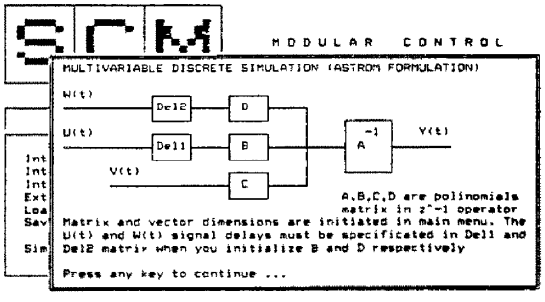

Fig. 4. Help screen of discrete time model simulation

2) Continuous linear SISO or MIMO models: with the same structure as the previous, but in the Laplace domain operator s.

3) Non linear model: the equations of a non linear model can be specified by the SimEstad module. The user must specify the system equations, compiling and linking the module with the rest of the package.

4) External simulation: The same previous model can be used from another computer performing as a plant. The link between both computers is made by serial line RS-232C.

The User Interface

First of all, two parts have to be differenciated. There are related to menu and command phases.

These parts are completely isolated, except for the specific control variables, which are initialized by menus.

Menu mode. Before the main menu initialization, every variable is read from a description file, which preserves the changes previously made. In this way, full and easily modifiable menus are presented to the user.

The menu system is organized in a tree structure. Each menu is contained in a module and the different options modify variables or activate submenus, which are only called by their father menu.

ESMmenu and WM are included as common modules, both, with great scope. These are the menu system kernel. The first carries out all I/O operations, while the second accounts for the window handling.

Command mode. When the SCM system is controlling the plant, the other code block is executed. Now, the dialogue with the user is carried out by a command interpreter. This solution has been adopted, instead of using menus, because the number of operations to be done and parameters to be adjusted is relatively reduced. Furthermore, as the dialogue with the operator occurs, a graphic screen is shown and it is necessary to use as much as possible of the screen surface for such a purpose. This implies the use of a reduced number of lines (four normally) for the dialogue with the user.

The interpreter kernel is the IntCom module, which accepts a line as input, parses it in the different commands and subcommands and processes it.

Once the command is parsed, the interpreter checks whether the command is on the command table and if its sintax is correct. See fig. 5.

```
"SCM > " C "!"
! !    >1
1 BAT*CH   "_Batch file: "     A "batch.bat" >3
1 DEP*OSIT "_Variable name: " A ""
           "_New value: "      R ""      >24
1 EX*AMINE "_Variable name: " A ""      >20
1 END      "_Process name: "   C ""
  2 BAT*CH          >13
  2 CON*TROL        >16
```

Fig. 5. Extract from the command table.

The access to variables is also supervised by a table. See fig. 6. In this table the owner module, type of variable, dimension, limits, protection, etc, are defined. The "examine" and "deposit" commands accede to the "Variab" module, which supervises access to the variable, detects error ranges, non-existant variables, forbidden access, etc.

MODULE	NAME	TYPE	LIMITE		DIM.	INDEX_RANGE		[PROT]
SimCom	cd	R	-1E20	1E20	3	0,3	0,3 0,7	N
SimCom	dd	R	-1E20	1E20	3	0,3	0,3 0,7	N
SimCom	dimad	E	0	7	2	0,3	0,3	L
SimCom	dimbd	E	0	7	2	0,3	0,3	L
SimCom	dimcd	E	0	7	2	0,3	0,3	L
SimCom	dimdd	E	0	7	2	0,3	0,3	L
ControlCom	e	R	-1E20	1E20	1	0,7		L
ControlCom	facel	R	-1E-8	1E10	0			N
ControlCom	hma	E	0	32000	0			L
Variab	nul	R	0.0	0.0	0			N
ControlCom	r	R	-1E20	1E20	1	0.7		N
ControlCom	u	R	-1E20	1E20	1	0,7		L
ControlCom	v	R	-1E20	1E20	1	0,7		N
ControlCom	w	R	-1E20	1E20	1	0,7		N
ControlCom	y	R	-1E20	1E20	1	0,7		L

Fig. 6. Extract from the variables table.

The Process Interface

Basically, each process is a main module, which has access to several calculus algorithms and general handling device modules situated on a second level. The third level is composed of specific drivers depending on the corresponding device.

For example, the present configuration of the system has a Metrabyte Corporation DASH-16/16F board,

having 16 channels of analog inputs, 2 of analog outputs, 4 of digital inputs and 4 of digital outputs. Additionally, it has an internal clock, DMA possibility, etc.

IMPLEMENTATION PHASE

For developing the data and flow control structures of the programs, it is necessary to select the implementation language.

In this sense, there are several high level programming languages adequate for real time systems implementation. These languages verify the previously specified conditions. Ada, Modula-2, Concurrent Pascal, etc, offer modern solutions to the implementation of this kind of systems.

Modula-2 has been used because of it is a small and compact language, which makes it specially adequate for small and medium size systems like the SCM system, developed on a PC-AT compatible under DOS operating system. There are several Modula-2 commercial compilers for PC computers.

Also, low level routines may be efficiently implemented without sacrificing the benefits of a high level modular programming language. Modula-2 allows true modular programming with strong type checking while incorporating the flexibility of routines for data transfer between variables of different types, interrupt handling and access to underlying hardware.

Modula-2 includes data abstraction mechanisms and, particularly, the concept of modularity, very useful in this phase:

- It helps the depuration of errors in the program. After the tests, the restricted access to modules prevents any environmental error from disturbing its operation. The system can be constructed by debugging and installing each module separately.

- It helps the maintenance of program. Unlike a procedure or a function, which cannot interact with their environment modifying non-local variables, a module is a close item (except the exported and imported objets). Thus, a module can easily be modified. If the interface does not change, the modifications will not affect other modules.

- It helps the separation of software blocks with different functions, and, particularly, the ones that will be required for the different operative modes of the control system.

- Modula-2 offers a systematic way of expressing the relationship between different modules and their variables, in such a way that it is easy to modify some modules in order to incorporate new characteristics.

The compiler used is the Logitech Modula-2/86 package, an effective development tool providing debugging facilities (post-mortem and run-time) that supports up to 1 Mb of physical address space and the creation of overlays for larger systems. It is a low cost compiler for PC-Compatible computers, which only needs 256 Kb of RAM memory.

CONCLUSIONS

In this paper, a set of basic requirements for the design and implementation of real time control systems has been presented. The application of these guide-lines is shown on the SCM package implemented in Modula-2 on the PC-AT. Then, the utility of this language for implementing small/medium size control systems is shown.

One of the aims of the SCM system, besides industrial control, has been the possibility of easy incorporation and test of new algorithms. This allows to improve some of the present applications, research for example. Other uses of SCM are the process simulation and the training of personnel in the field of process control.

REFERENCES

Clarke, D. W. (1983) "PID algorithms and their computer implementation", OUEL Report 1482/83.

De Keyser, R. M. C. (1986) "Personal computer training for software adaptive control" JOURNAL A, Vol.27, No.3, pp.155-161.

De La Puente, J. A. and A. Crespo (1984) "Programación sistemática para control industrial", II Simposio Nacional sobre Automática en la Industria, pp.315-321.

Hoare (1974) "Monitors and operating system concepts", ACM, Vol.17, pp.549-557.

Hoare (1978) "Communicating sequential processes" ACM, Vol.31, pp.666-677.

Lamber, R. (1980) "Design methods for computer controlled real time systems" 6th IFAC/IFIP Conference on Digital Applications of Process Control, Pergamon Press, pp.25-34.

Macleod, I. M. (1982) "The application of modern software engineering techniques in the development of a process control package", SOCOCO'82, Madrid.

Plessmann, K. W. and H. Zoller (1986) "High level language programming in the field of process control", IECON'86, pp.665-670.

Tuffs, P. S. and D. W. Clarke (1985) "FAUST: A software package for self-tuning control", IEE CONTROL'85, pp.186-191.

Wirth, N. (1982) "Programming in Modula-2", Srpinger Verlag, Third Edition.