UNITED ARAB REPUBLIC

THE INSTITUTE OF NATIONAL PLANNING

Memo No. 876

PROCESS CONTROL SERIES

PROCESS CONTROL SOFTWARE

By

ALWALID ELSHAFEI

OPERATIONS RESEARCH GROUP

March 1969

CAIRO

3, MOHAMED MAZHAR - ZAMALEK

INTRODUCTION.

Process environment dictates the design of process computer software. This environment may involve chemical synthesis, metal rolling, steam power production, or any process susceptible to computer control.

Functions of a process computer include such things as reading sensors, setting control devices, and logging operational data while operating as a standard data processor. Design of the software which directs the computer's operation is dictated by the specific requirements of the process involved.

The early expectation that process control computers would require little software support (because they would be programmed once and for all to do a specific job) was very quickly replaced by the realization that not only was all of the sophistication of conventional software required but, in addition, the real-time aspects of the process control application also had to be supported by software. This realization has led to a complete time-shared executive system with an extensive library of subroutines for communication between the computer and process. During process free-time, these executives permit use of the computer for off-line tasks under the control of a subroutine called the off-line monitor to distinguish it from the executive itself. The library of subroutines represents, in a sense, a language for process control, allowing programming of quite extensive real-time control systems. Several software designs requirements are common to all systems, they are:

1. Computer programs must react quickly to process events. Alarms and corrective controls must be given with little delay.
2. Programs must be scheduled so that computing time does not exceed available real time.
3. Input and output devices must be allowed to operate at (or near) their maximum speeds.

4. Actions which occur at definite times or within definite time periods must be scheduled according to a real-time clock.
5. Information must be gathered from the process as it becomes available and transmitted to the process as it is needed.

Process control software may best be understood by considering the control problem itself and the software and hardware needs it generates. A process computer system in a chemical plant gathers temperatures, pressures, and flow rates from the process. A raw material composition and marketing requirements change, programs calculate and send out changes in plant operating conditions. When emergencies occur - such as extreme temperatures or pressures - visual and audible alarms and corrective actions are sent to the plant. Process information, recorded each hour, helps determine long range operating strategy.

In a metal rolling process, information is gathered beforehand from laboratory analyses, physical measurements, and customer requirements. As a rectangular slab of metal is converted to a flexible strip, its position and temperature are recorded through sensors. Data are combined by the program, and corrections in roll force, speed, and temperature are calculated and sent to the mill. Alarms are generated when schedules or design requirements cannot be met or when equipment fails.

When starting up or shutting down a steam power plant, the computer system gathers turbine speed, motor position, temperatures, and pressures. As turbine speed is increased, the program scans eccentricity and vibration. If vibration exceeds an upper limit, turbine speed is held constant or decreased, or the turbine may be shut off. Visual alarms are sent to the operators to inform them of the emergency condition. Once conditions are steady, the system monitors temperatures and vibration and prints them each hour.

CONTROL LEVELS

Since the objective of the control systems is to successfully operate the process despite the presence of many disturbances, it is appropriate to partition the disturbances according to their relative frequency and consider the control of the system in the presence of each of these disturbances separately. The resulting control system hierarchical in form, is shown in Fig. 1

The highest frequency disturbances which must be considered are physical upsets which cause process variables to deviate from their "reference" values. For example, flows, temperatures, pressures, and the like will not long stay constant without the continual intervention of a control of some kind. Such control is usually fast and, if accomplished by a digital computer, termed direct digital control (DDC), and is generally regulatory in nature. Often this first level of control is accomplished by analog controllers with the digital computer supplying only the higher levels of control (such a control system is termed a supervisory control system). This is especially true in existing plants which already have analog control and add the supervisory control in order to increase production, cut costs, etc. The operator communication task in Fig. 1 is concerned mainly with this first level of control. One of the most important tasks at this level is alarming in case any process variable exceeds prescribed safe limits. This involves informing the process operator and then either taking appropriate action or turning control over to the operator. The computer is especially efficient in alarming and displaying overloads and is a tremendous assist to the human operators.

The second set of disturbances of importance are less frequent in nature and are caused by the changing mode of operation of the process. For example, a power system has drastically different load requirements during day and night hours and the operation of the process differs

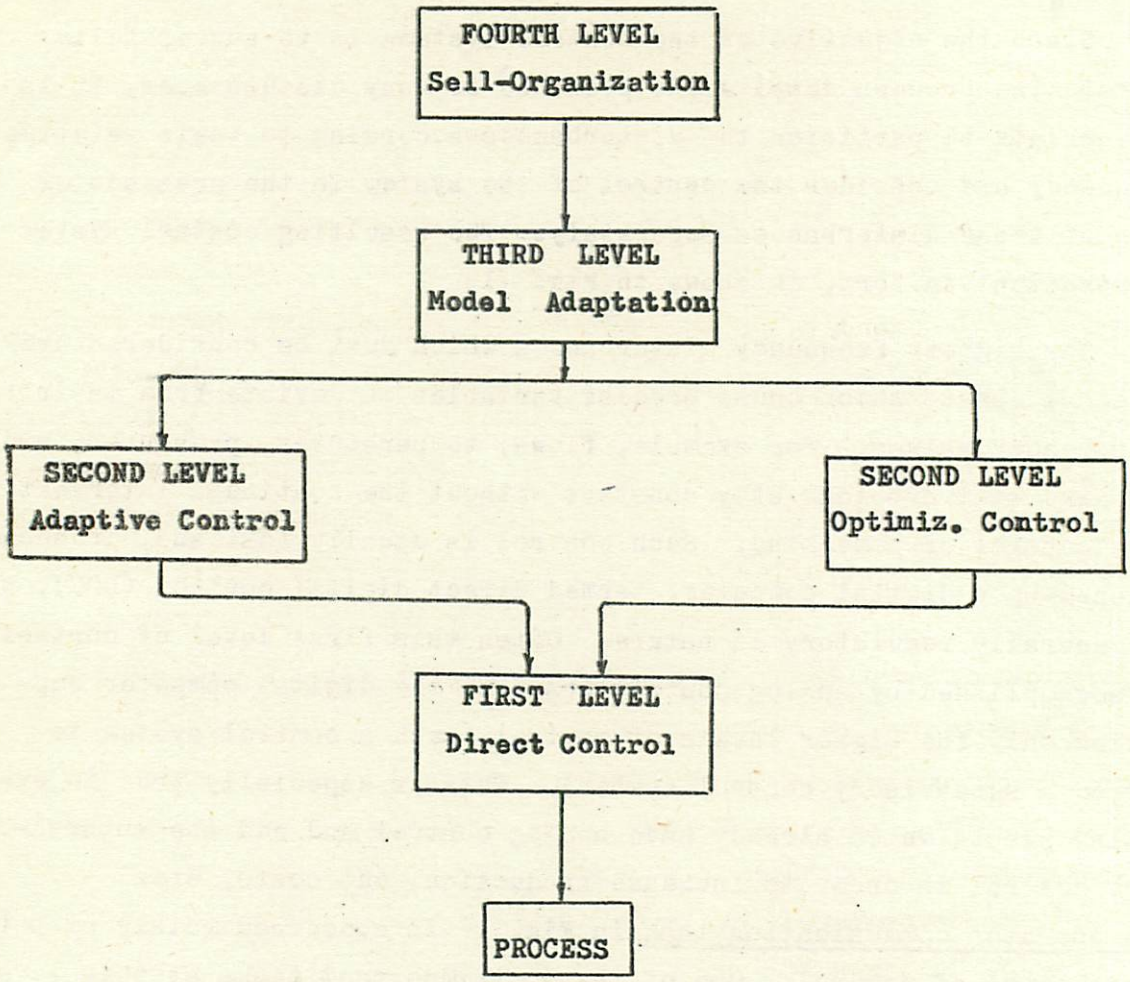

FIG.1 DIFFERENT CONTROL LEVELS

considerably at high and low loads. The second level control system which compensates for these disturbances has two functions. First, as the disturbance changes, it may be desirable to change the operating points (reference values for process flows, pressures, etc.) in order to make the operation as economically attractive as possible. Such a control system supplies new reference points to the first level analog controllers or digital controllers and is termed an optimizing controller. Secondly, on the same level, the dynamics of the process itself may change as the operating conditions of the process change, requiring modification of the first level controllers if adequate dynamic performance is to be retained. Adaptive control is implemented on this level to update or adapt the parameters of the first level direct digital controllers. This is most easily done if the first level of control is DDC rather than analog control since the controller parameters are then merely numbers stored in machine memory.

The third class of disturbance is even slower in frequency and consists of changes in the parameters of the process itself. This may be due to seasonal changes, aging, corrosion, and many other factors.

If the optimizing control is to be effective, it must have an accurate mathematical model and consequently the objective of the third level controller is to make use of process data to update or adapt the parameters of the mathematical model of the process itself.

The highest level of control compensates for the lowest frequency disturbance of all, changing process structure (in contrast to changing process parameters). This fourth level of control is seldom automated but rather supplies appropriate information to management and operators of the process so that they can make the proper decision as to structural changes in the overall control system and process (Fig. 1); This level might include emergency restart of the process,

emergency shut down, etc.

Observe that this breakdown of the problem results in hierarchical control system in which each level of control effects directly the level of control directly below it and in turn is controlled by the level immediately above it. This structure is attractive not only from a conceptual viewpoint, but also because each level can be designed somewhat independantly of each other level, thereby permitting the design of a complex control system by means of building blocks which are easily changed and improved upon as the appropriate technology becomes available.

From a software point of view, this division of a large task into many independent subtasks is very desirable since the programming of each subtask may be done independently and is then easily documented and updated at later times. However, the various subtasks are not necessarily performed at the same rate or even in the same sequence. In fact, various tasks may be performed under emergency conditions or upon demand of management or the process operator. Consequently, the software must permit the programmer to efficiently control the sequencing of these tasks and to change the sequence easily when it is desired. Of particular interest is the observation that the computational load is approximately the same for each level of control. That is, going up in level decreases the frequency at which the computation must be performed, but the complexity of computation increases with the result that the load (product of computation time and frequency) remains about constant.

The design requirements of the controlled process systems dictate the following program design characteristics:

- Since both time and memory are limited, programs share both.
- Some programs voluntarily give up time and space to other programs when it is necessary to wait for input or other actions.

- Some programs involuntarily give time to other programs which operate input/output devices, allowing the devices to operate at or near rated speeds.
- A real-time clock is used to schedule actions which must occur at definite times or time intervals, and to generate a time-of-day display. There must be a provision for resetting the clock through an external device.
- Core memory is fast but expensive, and bulk memory (drum or disc) is slow but relatively cheap. Programs in the system arrange for sharing the limited core memory between the programs in bulk sotrage.

PROGRAMMING FOR DIFFERENT CONTROL LEVELS:

There is a significant difference in the programming requirements of the first and higher levels of control. The simple algorithms of first level control are applied to processes in which there are many variables, all of which are regulated in a similar fashion. Consequently the computational load is very repetitive in nature and efficient programming and core allocation are necessary. On the other hand, the higher levels of control are relatively complex and nonrepetitive in nature and it is more advantageous here to use an algorithmic language in order to provide flexibility and good documentation.

The extensive input-output facilities of a process control computer may be needed by any one or all of the levels of control, since all make use of process data during their operation. Consequently, many decisions must be left to the programmer to make for each particular installation. For example, rate of scan of analog inputs, type of scan (sequential or random), changes in scan rate, strategy in case of input or output error detection, etc.., all vary from application to application and in fact from one level of control to another witin a given application. Thus any software system necessarily must permit the programmer to communicate easily with all of the hardware in the computer and cannot incorporate arbitrary decisions about these problems in a single executive.

Most of the software systems currently available for process control are designed to support the higher levels of control (the so-called supervisory control levels) rather than the first level, and are in the form of an executive system with real-time Fortran as the basic language. Two factors permit such executives to be used with supervisory control systems. First, the speed of machines has significantly increased so that Fortran-level programs can compete in terms of speed with machine-language programs in the earlier machines. Secondly, secondary storage has become readily available, permitting large executive systems and

efficient storage and saving of programs outside of core. The problem of servicing the first level of control can be solved without sacrificing the executive if a two-computer system is used: one computer doing primarily first level control (DDC) and the second, under control of the executive system, performing superivsory tasks as well as backup of the DDC computer.

CONTRAN LANGUAGE:

One exception to the real-time Fortran approach to process control languages is the CONTRAN system being developed by Honeywell. This language is an outgrowth of the Consequent procedure language developed by Fitzwater and Schweppe. Their language called TASK 64, is a modification of ALGOL 60 to include task processing statements and consequent procedures (procedures which are initiated when prescribed conditions are fulfilled). Such a language operates within an executive system as does Fortran, but goes a step further than the subroutine library approach. Control of sequence of various portions of the control program is obtained by the specification of set of Boolean variables or switches for each program so that the program will be executed when, and only if, these conditions are fulfilled. Thus these conditions, rather than the order of program statements or routines, determine the sequence of their operation. Control of interrupts and communicationbetween the process and the computer through input-output devices are obtained through subroutine calls as in real-time Fortran.

CONTROL PROGRAMS :

Program sequence control (PSC) - controls the sequencing and initiates the loading and execution of user-specified process core loads.
Master Interrupt Control (MIC) - automatically determines the type of each interrupt as it is recognized and transfers control to the proper interrupt servicing routine.

Interval Timer Control (ITC) - provides a programmed real-time clock, a timer for TSC, nine programmed interval timers, and control for two machine-interval timers.

Time-Sharing Control (TSC) - Controls the timesharing of variable core between process and nonprocess core loads.

Error Alert Control (EAC) - provides the following functions whenever an error occurs:

1) Optionally saves core for future reference,
2) Optionally branches to a user's program for further error analysis.
3) Prints an error message.
4) executes a specified recovery procedure.

Communications Control (COMC) - controls communication with the PSC and the I/O programs.

Bulk Transfer Driver (BTD)

controls transfer between bulk storage and core

SEQUENCE CONTROL:

Statements which permit the programmer to control the order in which tasks are performed interrupts serviced, and off-line jobs permitted. Such control is important, since the various levels of control are necessarily carried out in sequence rather than in parallel and the order is critical. For example, a sequence of tasks might be to collect certain data, use a statistical identification routine to determine

parameters of the process, and finally to use an adaptive routine to change the controller parameters. An optimizing routine too large for core can be executer in parts if the programmer has control over the sequence of programs.

Program Sequence Control (PSC) is a control program that handles the flow of control from the mainline core load to the next. PSC functions are initiated by execution of PSC CALL statements in the user's program. The specific functions of PSC are:

1- Execute the next sequential mainline core load. The new core load overlays the one that contained the call.
2- Save the mainline core load in progress (on disk) and load a special core load for execution.
3- Restores the core load that was saved in item 2 and continue execution from where it left off(the statement following the CALL SPECL).
4- Queue mainline core loads associated with interrupts whose occurrence has been recorded.
5- Execute the highest priority mainline core load listed in the core load queue.
6- Insert mainline core load entries into or delete them from the core load queue.

For PSC to perform the above functions, a CALL statement must be executed for each one. The specific CALL statements and their parameters are described below.

Commands giving this type of control can be categorized in three groups used to:

1- CALL the next mainline core load to be executed.
2- SAVE the present mainline core load (or disk) and CALL a special mainline core load for execution.

3- RESERVE and CONTINUE execution of the saved mainline core load

consequently, those commands can be classified as follows:

1- CALL STATEMENTS, including:

- Normal Call - CALL CHAIN (NAME), specifying next program to be executed.
- Special Call - CALL SPECL (NAME)
- Return Saved Mainline - CALL BACK

2- QUEUING STATEMENTS, including:

- Insert Into Queue - CALL QUEUE, entering program in a waiting queue.
- Delete From Queue - CALL UNQUEUE, removing program from a waiting queue.
- Execute Highest Priority Core Load - CALL VIAQ
- Queue Core Load If Indicator Is ON - CALL QIFON
- Clear Recorded Interrupts - CALL CLEAR

3- SHARING STATEMENTS, including:

- SHARE, indicating availability of free time in which non-process programs may be executed under the control of the off-line monitor.

Such statements may be freely inbedded within process programs written in FORTRAN. Through use of these commands within programs, the programmer can control the frequency and order in which the various levels of control are performed. Even when various levels are not performed on a regular basis (for example, when a certain function is performed only upon operator demand), these commands permit control over the sequence. In response to an operator-initiated interrupt, the interrupt subroutine can decode the request and call for the

appropriate program to be entered in the queue and then executed when it has the highest priority. Of equal importance is the ease by which sequence is changed as the process control problem changes with time.

INTERRUPT CONTROL:

This includes the control of the routines which service interrupts and the control of the interrupts themselves. For example, during certain routines it may be advantageous to delay the serviving of interrupts to minimize exchanges of programs or to prevent certain interrupts entirely (as when a routine cannot be used recursively and may be called from more than one level.

The program in charge of such control is the MASTER INTERRUPT CONTROL program (MIC). It controls the servicing of interrupts, an interrupt may occur at any time but it will not be recognized by MIC unless the interrupt is on a level that is not marked and is of higher priority than the present level of machine operation. The user-assigned interrupts can be delayed from being recognized by masking the level to which they are assigned. The servicing of process and COUNT subroutions can also be delayed by recording their occurrence. There are, basically, two types of interrupts:

EXTERNAL INTERRUPTS:

are those associated with the process and programmed interrupt features. They are serviced, or recorded, by one of four types of user - written routines:

1- Skeleton Interrupt Routine
2- Mainline Interrupt Routine
3- Interrupt Core Load
4- Mainline Core Load