UDC 681. 3. 06

FACOM U-200 PROCESS ORIENTED PROGRAMMING SYSTEM (POPS)

Tsuneyoshi Oboshi
Yasuyuki Kohara

I. INTRODUCTION

It has been some years since the term "software crisis" appeared in the software field. During this time, the changes in the environment where the price of hardware dropped, personnel costs increased, higher level functions were required for systems and applications were made more universal, caused the software crisis to get worse and the situation to make more urgent. The field of computer application to process control can not be exception either. The situation where systems were made only to work by frantic efforts has not been supported by either users or manufacturers. Demands became especially strong for easy-to-use programming systems with assured realiability prepared in the planning stage. Such standard programming systems not only increase the efficiency of system design and compilation but also improve the overall system reliability and ensure the maintenance of systems which are especially interesting for users.

Fuji Electric last year announced a new computer system for process control, the FACOM U-200, and reported an outline of the hardware and software(1). This article will describe the POPS (Process Oriented Programming System) which has been developed for the FACOM U-200.

II. DEVELOPMENT POLICIES AND AIMS

Among the programming systems, the programming language occupies an important position. In the last few years, the development of languages for processes has been pursued actively in various fields.(2) The results have not always satisfied the original expectations but most of the problems in programming languages for processes have been clarified and future trends have been tried to be set. Various types of process languages have already been reported and in the field of Problem Oriented Languages (POL), it is expected that languages with various characteristics will be developed in the future. However, no matter in what directions process languages develop, the importance of the group of process subroutines when the program written in the language is executed does not change. As the program description system becomes more macro (this does not mean macro-language), the functions and efficiency of the process subroutines and packages have a greater influence on the system as a whole. Even though the expression in the program is the same, the operation and efficiency in the execution of the object program depends to a great extent on the process subroutines and packages.

From the above points, one of the main points stressed in the development of the FACOM U-200 POPS was the preparation of process subroutines and packages and further the following points were considered:

1) Reflection of past experience in the specifications

Fuji Electric has been working with computer control systems for above the past 10 years. The fields of application have been wide ranging including electric power, iron and steel, chemicals and water treatment and considerable help has been obtained from users concerning processes. The users have also given their evaluations and opinions concerning software systems. The work of investigating specifications for the FACOM U-200 process subroutines and packages was started from investigations and adjustments of requirements for a large number of application systems. As will be described later, the large number of specifications compiled were not necessarily special and were ordinary but were considered as sufficient for applications over a wide field. From another viewpoint, even when the application field differs and the structure and functions of the system appear to differ, the specifications for process subroutines and packages which form the core of the software system will be common over a wide range and also such a system of subroutines and packages will make the programming system easier to use.

2) Efforts toward modulation

The advantages of software modulation need not be described here. There is the opinion concerning standardized software that "functionally, everything is possible which is good but such software is large and not easy to use". In the last few years, there has been a remarkable drop in the value of hardware. However, no matter how cheap the memory equipment becomes, it is not permissible for the required memory equipment capacity to be increased indiscriminately by the use of standardized software from the user's standpoint.*

On the process subroutines and packages for the FACOM U-200, the software necessary for executing certain functions was modulated and several types of modules differing functionally in their defense ranges are provided. By selecting the optimum module for the system applied, it is possible to keep down the size of the required memory capacity. As a simple example, an explanation will be given of the access subroutine of the file of three dimensional structure. The three dimensional file can be located in either the main or auxiliary memories. The three dimensional file access subroutine must have access to both the main and auxiliary memories. However, depending on the system, the three dimensional file is located only in the main memory and in such cases, the part of the subroutine which accesses to the auxiliary memory becomes useless speaking in the extreme case. Therefore, three types of subroutines are provided: one for only the main memory, one for only the auxiliary memory and one for use with both. Any of these can be selected optionally in accordance with the requirements of the application system. Naturally, calling procedure of each subroutine is the same and when describing the application program, there is no need to be concerned about which subroutine is included in the system. The factors in preparing more than one module are as follows:

(1) Differences between input/output equipment making up the system
(2) Type of RTC input/output device
(3) Differences in memory media
(4) Differences in file structure
(5) Range of processing functions

*Note: Attention must be paid to the following points:
① Even if the required main memory capacity is increased to some degree by the use of standard subroutines and packages, considering the current prices of hardware and software, the use of standard routines is profitable as long as the capacity does not become extremely big and this trend should become stronger in the future. From considerations of software reliability, maintenance, etc., the amount of increase in the price of the memory equipment should be sufficiently compensated for by the results obtained.
② One of the interesting points in the memory equipment from the standpoint of the user is that some parts of the main and auxiliary memories are occupied by the system program, so that what amount of region remains for the use of the user. The problem is to what range can be called as the system program. If the process programming system is complete, large memory equipment is naturally needed. However, compilation and testing of application programs become easier and the application program itself should be smaller. Therefore, considering only the size of the system program has no meaning in many cases. It is important how much the functions for processes which application programs require can be covered.

(6) Differences in data formats
(7) The size of multiple use area in the main memory

3) A complete table look-up system

The system which gives the detailed specifications of the application systems in tables has already been used for a long time and its effects have been fully confirmed. In this case, a further advance has been made and all of the table groups are controlled together in files. In this way, system editing, correction, follow-up, etc. are easy and it is possible to apply problem oriented language directly with this system.

4) Provision of file subroutines

One of the features of the process system is that many tasks with different priority levels have random access to the common file groups. During processing in the application program, access to the data file which occupies a rather large part has a big influence on the application system as a whole because of the efficiency and reliability of the file access subroutine.

There is no established theory concerning the file structure and access method in the process system and various theories are given, but in the FACOM U-200 POPS, Fuji Electric adjusted various types of file with which it has had experience so far and prepared access subroutine to each of them. In the design of the application system, the most basic problems are the structure of the process data file and its access method. Determining whether the designed data file is appropriate or not is connected with the efficiency during system execution, and the ease of system compilation, testing and maintenance. The file subroutines prepared in this system are considered to almost completely cover the files required in the process system. If these subroutines are used effectively, system design, compilation, testing and maintenance should be easy.

5) Emphasis on character disply equipment

Several years have passed since display equipment appeared in process control systems. The advantages of display equipment have been widely recognized from the first and they are the main method of man/machine interface. At present, the process operator's console (POC) can not be considered without display equipment. However, display equipment has considerable adaptability and since these devices are closely related to software, they play a major role in the software. If emphasis is placed on the dialogue functions of display equipment, various types of use almost without limit can be considered but amount of work of software compilation for these uses becomes also tremendous.

In the FACOM U-200 POPS, these factors have been considered and the subroutine package CSP (Character display Subroutine Package) is developed. In the CSP specifications, Fuji Electric has included the concepts related to the modes of use which have been experienced so far and also related to the tomorrow's use of display equipment. In the CSP, special emphasis is placed on the dialogue function and the application of various types other than POC has also been considered.

6) Provision of debugging aid routine

Along with the tracer, snap shot dump and the dummy functions for I/O, the following points have been considered concerning the debugging aid routine:

(1) Debugging data input/output function

To increase the efficiency of the debugging work, it is necessary that it be easy to feed the debugging data into the file and dump the results from the file. In the FACOM U-200 POPS, file utilities are used since it is easy to use at the time of debugging.

(2) Improved document characteristics of debugging results

The output list from the debugging aid routine is such that it can be used directly as the program test results. For example, there are various data formats which can be used for data expression on the output list and it is easy to see. In particular in the output list of the file utility, comments can be added in any optional form and if a suitable comment is given, the results can be checked by persons without a deep knowledge of the program contents.

(3) On-line use also possible

It takes time to clarify the factors when an accident occurs during system tests, test runs and actual operation. With the debugging aid routine used in this system, the accident analysis can be performed without any effects on the execution of the on-line program.

The above paragraphs have described the concepts used in the development of the process subroutines and packages which form the core of the FACOM U-200 POPS. These subroutines and packages can all be used for programs written in FORTRAN too. In the FACOM U-200 POPS, the programming languages include the macro-language FPL (Fuji Process Language) in addition to FORTRAN. The macro-type language is also desirable as a method for developing problem oriented language. However, when compared with such languages as the FIF (Fill in the Form) and the interpreter type, the delay in the use of this language is considered to be due to the fact that it is difficult generally to select the language for the results expanded from macro and therefore, a suitable processer (macro-generator) for expansion of the macro-instructions was not developed. However, in the FACOM U-200 assembler (FASP), a rather high level macro-generator (hereafter referred to as MACGEN) is employed for a computer system of this scale. MACGEN has a large number of assembly instructions for control of the generated statements and various types of instruction groups can be formed in accordance with the parameter specifications. FPL is a language developed in the basis of MACGEN. Because MACGEN is based on the assembler, it is limited to the type of statement description but it is suitable for the development of problem oriented language because of the following points:

(1) If the language specifications are decided, development and compilation can be performed easily in a short period.
(2) The required instructions can easily be added.
(3) Since it is easy to form different instruction groups from the same instructions, suitable languages can be formed for each application field.
(4) The efficiency of instruction groups formed is good when compared with compiler level language.

The FPL consists of a language part of universal procedures similar to FORTRAN statements and a part of problem oriented language applied for each application. Naturally, it is possible to express the application program only in the procedural language part but it is desirable to make use of the problem orientated language part and add the necessary instructions. Generally, program coding and debugging are easier and the efficiency of the instruction groups formed is better if the functions are larger with one macro-instruction (literally macro-like).

FPL is used actively in POPS for system editing, FORTRAN subroutines, part of thc packages, etc.

III. POPS SUBROUTINES AND PACKAGES

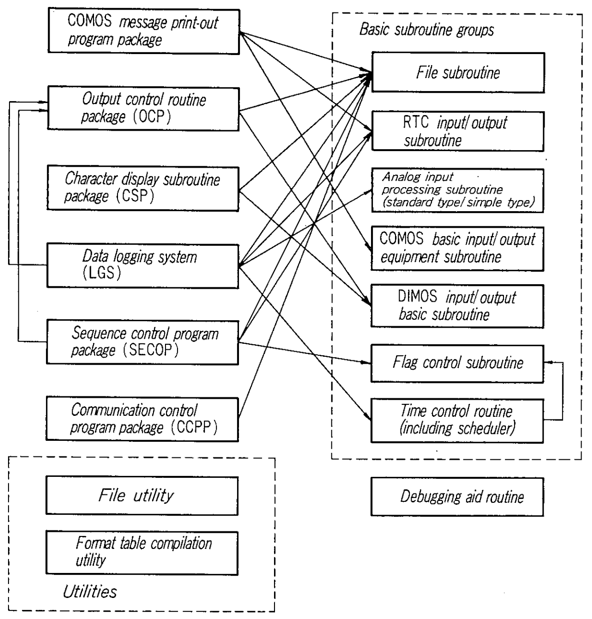
Fig. 1 shows the FACOM U-200 POPS subroutines and packages. The following sections give an outline of the main subroutines and packages.

1. File Subroutine

(1) Basic routine of auxiliary memory access

The 4 types of subroutines shown in *Table 1* are functional expansions of the COMOS/DIMOS auxiliary memory access service routine to facilitate use from the application program. With the N words transfer subroutine (RDWN/WDWN), transfer is performed via the buffer region on the main memory. This buffer region is controlled so that the file contents will not be disturbed even if there is random access from many tasks with different priority levels for files on the auxiliary memory. Therefore, it is possible in the application program to access the files on the auxiliary memory without the problems of interference between levels of tasks and so that programming is easy. The size of the buffer region can be specified for each application system and if sufficient size can be assured, the number of accesses of the auxiliary memory will be reduced and this is also an advantage from the standpoint of processing speed.

(2) File control subroutine


COMOS message print-out program package
Output control routine package (OCP)
Character display subroutine package (CSP)
Data logging system (LGS)
Sequence control program package (SECOP)
Communication control program package (CCPP)
Basic subroutine groups
File subroutine
RTC input/output subroutine
Analog input processing subroutine (standard type/simple type)
COMOS basic input/output equipment subroutine
DIMOS input/output basic subroutine
Flag control subroutine
Time control routine (including scheduler)
File utility
Format table compilation utility
Utilities
Debugging aid routine

Note: COMOS: Core Monitor System
DIMOS: Disk Monitor System

Fig. 1 Construction of POPS subroutines and packages

The basic routines shown in *Table 1* perform access without any involvement of the file structure of the region concerned. The file control subroutines perform access under the premises of the various types of file structures shown in *Table 2*. These files can be located in either the main or auxiliary memories except for some cases. With the file control subroutines, auxiliary memory access is performed by the N words transfer subroutine.

The detailed specifications such as location of each file in the memory equipment are specified in the file table on the main memory, the location, etc. of each file can be changed by changing the contents of the file table. The file control subroutines are used in various types of packages as is evident from *Fig. 1*.

2. RTC Input/Output Subroutines

Many types of devices can be installed in the RTC (Real Time Controller),[1] so that many exterior input/output signals can be handled. Access to these devices is easy by means of the subroutines shown in *Table 3*. Access to the various RTC devices is according to the address (PDN: Physical Device Number) determined by the hardware in each case. However, generally the PDN does not maintain continuity even for the same type of device and they naturally differ for each system. In the RTC input/output subroutines, the system designer defines the Logical Device Number (LDN) logically and access is possible in accordance with the LDN.

3. Analog Input Data Processing Subroutines

The subroutines shown in *Table 4* perform conversion, statistical processing, monitoring, etc. of analog input data. These subroutines include standard types which handle floating decimal point data and simple types which handle fixed decimal point (integer) data. The standard subroutine groups are used with the data logging system which is described later.

4. COMOS Basic Input/Output Equipment Subroutine

Table 5 gives an outline of the COMOS basic input/output equipment subroutines. These subroutines are used with the COMOS which has major restrictions in respect to memory capacity and therefore, consideration has been given to suppressing drastically the required memory capacity of the input/output buffer region, etc.

5. DIMOS Basic Input/Output Subroutines

The subroutine groups shown in *Table 6* are used as the basic subroutines for the CSP and OCP packages described later. In the DIMOS, the input/output buffer region can be set on the auxiliary memory too and the operational efficiency of the computer is raised by means of the "leave-behind" control. Operation of all of the output devices simultaneously is possible.

6. Flag Control Subroutines

There are many ON/OFF data (flags) transferred

Table 1 **Basic subroutines for drum/disk memory**

Name of subroutine	Entry name	Function
(1) Detached type record transmission subroutine	STDRR/STDRW	Transmission demand is added to the queue and fed back to call program. Transmission is performed by separate task in demand sequence. After transmission is completed, task given by parameter starts.
(2) Wait type record transmission subroutine	TRDWR/TRDWW	There is queuing in the wait until completion of demand transmission. The call origin task must be a task which can be processed in parallel.
(3) Completion feedback type record transmission subroutine	TRDRR/TRDRW	There is queuing by occupying the level until the completion of the demanded transmission. The call source task need not be one which can be processed in parallel.
(4) N words transmission subroutine	RDWN/WDWN	There is access to an optional N words in the auxiliary memory via the main memory buffer. If the specified range is on the buffer, there is no access to the auxiliary memory.

Record=32 words (drum)/128 words (disk)

Table 2 **Data file format and control subroutines**

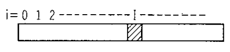
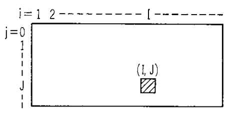
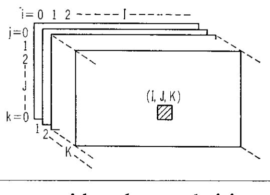
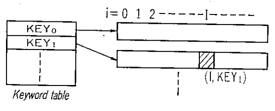
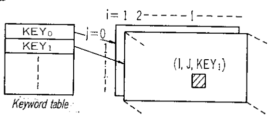
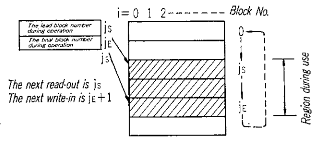
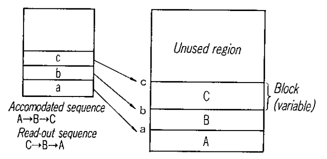
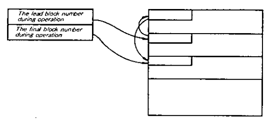
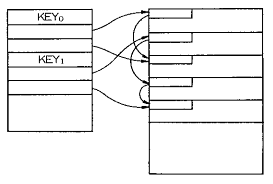
File name	Structure	Subroutine	
		Entry name	Function
(1) Pure one dimensional file		RFIL1/WFIL1	Performs read-out and write-in of the pure one dimensional file
(2) Pure two dimensional file		RFIL2/WFIL2	Performs read-out and write-in of the pure two dimensional file
(3) Pure three dimensional file		RFIL3/WFIL3	Performs read-out and write-in of the pure three dimensional file
(4) One dimensional file with key	It can be considered as subsitituted by a keyword defined by the user in place of the j of the pure two dimensional file 	RFILK1/WFILK1	Performs read-out and write-in of the one dimensional file with key
		KREG/KCAN /KCHK/KCLR	Performs recording, deletion, etc. of keywords for keyword tables of one and two dimensional files with keys
(5) Two dimensional file with key	The k of the pure three dimensional file can be considered as substituted by the keyword 	RFILK2/WFILK2	Performs read-out and write-in of the two dimensional file with key
(6) Cyclic file		RFILC/WFILC	Performs read-out and write-in of cyclic file
(7) Queue one dimensional file	File with record sequence control added to one dimensional file with key	RFILQ1/WFILQ1	Performs read-out and write-in of the queue one dimensional file
		KREGQ/KCANQ /KCHKQ/KCLRQ	Performs recording, deletion, etc. of keywords for keyword tables of queue one and two dimensional files
(8) Queue two dimensional file	File with record sequence control added to two dimensional file with key	RFILQ2/WFILQ2	Performs read-out and write-in of the queue two dimensional file
(9) Stack file		RFILS/WFILS	Performs read-out and write-in of stack file
		ERSSTK/SERCHU	Performs deletion, etc. of stack file block
(10) Link file		RFILL/WFILL	Performs read-out and write-in of link file
		LOFF	Performs breakage of links
(11) Link file with key		RFILLK/WFILLK	Performs read-out and write-in of link file with key
		LOFFK	Performs breakage of links

Table 3 **RTC input/output subroutines**

Subroutine name	Entry name	Function
(1) ON/OFF input read-in subroutine	DIN	Read-in of ON/OFF input of n bits
(2) BCD input read-in subroutine	XCI	Conversion to data type specified by read-in of n columns of BCD (binary-coded decimals) input
(3) ON/OFF output subroutine	DON	Performs ON/OFF output of n bits
(4) Numeric display subroutine	XCO	Converts integer or floating decimal point type data into format of specified numeric display device and outputs them
(5) ON/OFF control output subroutine	DOC	Performs ON/OFF control output of 1 bit
(6) Analog input read-in subroutine	AIN/AIL/AINL	Reads in analog input of n points
	AIAUT	Decides optimum gain of amplifier for analog input of 1 point and reads in
(7) Pulse input read-in subroutine	RPI	Reads out pulse input integrated values n points
	RCPI	Read-in of pulse input integrated values and clearing of integration region and pulse counter
	IPI	Clearing of pulse input integration region of n points and pulse counter
(8) Analog output subroutine	AO	Output of data given in analog output of n points
(9) Pulse output subroutine	PO	Output of data given in specified pulse output device
	POCHK	Check of output conditions
	POCNL	Cancel of output requests
(10) Real time clock read-in /correction subroutine	CLKA	Read-in of time point from clock device and conversion to specified format or correction of time

Note: refer to the various package sections concerning the RTC connected typewriter and communication lines

Table 4 **Analog input data processing subroutines**

Subroutine name	Entry name	Function
(1) Signal validity check subroutine	HDCK	Checks validity of analog input values
(2) Scale conversion subroutine	SLCVT/SLCVTR	Conversion to engineering units
(3) Thermocouple scale conversion subroutine	ICCVT/CACVT /CCCVT/CRCCVT	Conversion to engineering units by polynomial approximation of thermocouple input
(4) Upper/lower limit check subroutine	HLCHCK	Checking of upper and lower limits of input values converted into engineering units
(5) Temperature and pressure compensation subroutine	TPCOM	Performs compensation calculation by temperature and pressure
(6) Digital filter subroutine	DFLT	Performs filtering by a primary delay filter
(7) Statistical data processing subroutine	AVREG/SUM /MAXMIN	Processes statistical data such as average values, integrated values, and maximum and minimum values
(8) Conversion factor check subroutine	TREND	Checks difference between previous and present values
(9) Substitute data processing subroutine	SUBST	Processing of substitute values at time of check out by HDCK

among the programs. These flags are especially important in systems which consist mainly of logical processing such as sequence control. In the POPS, the subroutines shown in *Table 7* are used so that it is easy to handle flag regions (1 flag corresponds to 1 bit) used in common for each task.

7. Time Control Routines

The time control routines consist of the time data calculation routines and schedulers shown in *Table 8*. The schedulers are started by periodic interrupt signals from the RTC and they start the tasks periodically on the basis of time points of the RTC clocks or at specified times.

8. COMOS Message Print-Out Program Package (MPRINT)

In small scale systems centered on COMOS, there are packages which print-out on a typewriter connected to the RTC. The output format and data are specified in the table. In addition to message output, report type print-out is also possible. For the MPRINT, use with the COMOS has been taken into consideration so that the routines are compact and the size of the various tables prepared by the

Table 5 **Standard I/O subroutines for COMOS**

Subroutine name	Entry name	Function
(1) Equipment assign and release subroutine	OPEN/CLOSE	Performs input/output equipment assign and release
(2) Keyboard input read-in subroutine	KB	Reads in input from typewriter keyboard
(3) Typewriter print-out subroutine	TW	Performs print-out to typewriter
(4) Paper tape reading subroutine	PTR	Reads in contents of paper tape
(5) Paper tape punching subroutine	PTP	Punches output on paper tape
(6) Card reading subroutine	CDR	Reads in data of 1 card from card reader
(7) Card punching subroutine	CDP	Punches output on card
(8) Character display subroutine	WDISP	Performs display on screen of character display equipment
(9) Character display read-in subroutine	RDISP	Reads in data from screen of character display equipment
(10) Code conversion subroutine	CONV	Conversion between external/internal code

Table 6 **I/O basic subroutines for DIMOS**

Subroutine name	Entry name	Function
(1) Typewriter output request subroutine	WTW	Performs output requests* to typewriter (including RTC connection)
(2) Line printer output request subroutine	WLP	Performs output requests* to line printer
(3) Paper tape puncher output request subroutine	WPTP	Performs output requests* to paper tape puncher
(4) Card puncher output request subroutine	WCP	Performs output requests* to card puncher
(5) Input/output buffer control subroutine	OPNFIL/CLSFIL /PUTFIL/GETFIL /CANFIL	Reads in and out input/output data to input/output buffer region in auxiliary memory
(6) Code conversion subroutine	CVIN/CVOUT	Code conversion between internal and external codes

*In the subroutine, output data series are accomodated in buffer region of auxiliary memory.
Output to equipments is performed by a "leave-behind" control system using a separate task.

Table 7 **Flag control subroutines**

Subroutine name	Entry name	Function
(1) Flag ON/OFF subroutine	FON1/FOFF1/FSET /FONN/FOFFN /FONR/FOFFR	ON or OFF of flags of 1 bit or n bits
(2) Flag check subroutine	FCHK/FCHKR /FSCN/FSCNR /FSRC/FSRCR	Checks conditions of flags of 1 bit or n bits. Initially at ON bit notifies detected flag and turns OFF if necessary. Checking sequence is specified according to parameters
(3) Logic operation subroutine between flags	FANDN/FORN /FEORN/FAND /FOR/FEOR	Performs logic operations (AND, OR, NOR) between specified flags
(4) Flag transfer subroutine	FTRS/FTRSR	Transfers flags of n bits to other specified flag region and if necessary, resets to original flag region

user can be made as small as possible. *Table 9* gives an outline of the programs for MPRINT.

9. Output Control Routine Packages (OCP)

The OCP is a package which processes print outputs (messages and reports) and displays for man/machine interface, and paper tape and card outputs for off-line system interface. In process systems, such output functions: (1) increase output data capacity, (2) diversify output equipment and (3) widen the range of output formats. They also play a big role in system compilation. The OCP has the following features so that it is easy to provide complete man/machine interface functions.

(1) The output data and formats are specified by tables. Utilities as described later are provided for compilation, recording and control of these tables.
(2) A sufficient number of devices for the process can be handled (for example, 32 typewriters and display devices).
(3) Output requests are received by calling the out-

Table 8 **Time schedulers and subroutines**

Subroutine name	Entry name	Function
(1) 1 second scheduler	RTS1	Scheduler of 1~9 second starting tasks
(2) 10 second scheduler	RTS10	Scheduler of 10~50 second starting tasks
(3) 1 minute scheduler	RTS1M	Scheduler for tasks starting after more than 1 minute or at fixed time
(4) Table conversion subroutine for schedulers	TIM00/TIM10 /TIM60	Performs table changes for each scheduler
(5) Data change routine	DATEX	Performs date changes at specified times
(6) Time calculation subroutine	TIMCB	Performs operations between time data

Table 9 **Message print program package for COMOS**

Name of subroutine or program	Entry name	Function
(1) Output request receipt subroutine	MSGRQ	Links output request to receiving queue
(2) Subroutine to change table to original conditions	MSGCL	Concerts linked table to original conditions
(3) Character output program (task)	MPRINT	Picks out output request received by MSGRQ, performs format expansion of output data and gives out to specified device
(4) RTC connected typewriter print-out subroutine	OPNTW/CLSTW /PUTTW/TWN/TWC	Prints out output to RTC connected typewriter
(5) Data conversion subroutine	DCOVT1	Expands fixed point data of single or multiple accuracy to specified format
	TITLE	Expands print-out title to output format
	DATE	Expands dates and times to output format
	TAGNO	Expands tag No. to output format

put request reception subroutine with specifying the table No.

(4) Since one part of the output format and data given by the table can be changed during program execution, precise format control is possible.

(5) Substitute output functions during output equipment breakdowns are provided. The types of the output equipment can be different. This substitute processing system can also be changed by instruction of the operator.

(6) Since the package can be selected to suit the system on the basis of the equipment components and processing functions of the various systems, the memory capacity can be limited.

(7) The received output requests are processed by "leave-behind" control to increase the computer utilization efficiency. Naturally, the devices can be operated simultaneously.

Fig. 2 shows an outline of OCP operation. *Table 10* shows the main subroutines used from the user in the OCP.

10. Character Display Subroutine Package (CSP)

The importance of the display equipment in the process system and the problems as viewed from the software side are mentioned in section II 5). In the POPS, the CSP package has the following functions and features:

(1) Wide ranging display functions include screen color control, pattern display, point graph and rod graph displays.

(2) The protection field on the screen can be used effectively.

(3) With the CSP, the same functions can be achieved by the software for display equipment with no scroll up mode display function by means of the hardware.

(4) As can be seen in *Fig. 3*, insertion on the small screen (subpicture) of the display screen is possible. It is also possible to delete only part of the small screen during display and restore it to the main screen.

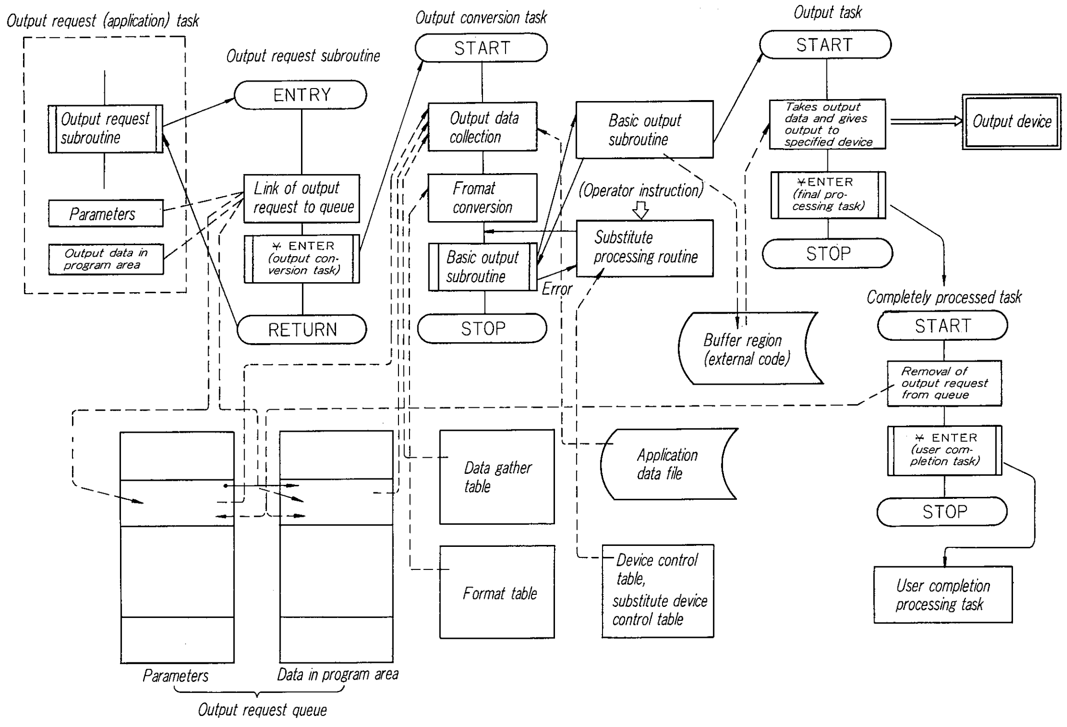
(5) Priority levels can be set on the screen during display. When there is a display request for high level screens, the low level screen during display is shunted and is restored when there is a deletion of the high level screen.

(6) The dialogue subroutines as shown in *Table 12* are provided.

(7) Interaction functions such as light-pen, function key and operator ID card can be used very effectively.

(8) Since there are standard programs for the standard dialogue functions, the users need only compile the tables.

(9) The detailed specifications of the screen display format and data, etc. are specified by the tables. As in the case of the OCP, utilities are pro-



Output request (application) task
Output request subroutine
Parameters
Output data in program area
Output request subroutine
ENTRY
Link of output request to queue
ENTER (output conversion task)
RETURN
Output conversion task
START
Output data collection
Fromat conversion
Basic output subroutine
STOP
Basic output subroutine
(Operator instruction)
Substitute processing routine
Error
Output task
START
Takes output data and gives output to specified device
Output device
ENTER (final processing task)
STOP
Buffer region (external code)
Completely processed task
START
Removal of output request from queue
ENTER (user completion task)
STOP
User completion processing task
Data gather table
Format table
Application data file
Device control table, substitute device control table
Parameters
Data in program area
Output request queue

Fig. 2 **Operating process of output control routine package**

Table 10 **Subroutines of output control package**

Subroutine name	Entry name	Function
(1) Logging output request subroutine	LGOUT	Receives logging output request and adds it to the queue
(2) Message output request subroutine	MSOUT	Receives message output request and adds it to the queue
(3) Logging output request cancel subroutine	LGCAN	Cancels the received logging output request
(4) Message output request cancel subroutine	MSCAN	Cancels the received message output request
(5) Total screen display request subroutine	CRTAL	Receives total screen display request and adds it to the queue
(6) Partial screen display request subroutine	CRTPT	Receives partial screen display request and adds it to the queue
(7) Subroutine for screen read-out during display	CRTRD	Reads out screen contents during display. Used for screen shut and restoration
(8) Display deletion subroutine	CRTOF	Deletion of screen during display
(9) Error data read-out subroutine	ERPAD	Notification of accident in output device
(10) Substitute processing methode indicate subroutine	EXCHS	Indication of substitute processing method
(11) Display accident recovery notification subroutine	RCOVR	Notification of accident recovery in device

vided for compilation, recording and control of these tables.

(10) For convenience in debugging of tables and programs compiled by the user, print-outs of CSP control table contents and display contents (including data related to color control and protection fields) are possible.

Table 11 shows the main CSP subroutines and *Fig. 4* application examples.

11. Data Logging System (LGS)

The LGS is mainly for continuous processes and is a package which performs field data read-in and processing, accommodation in files and message output, log sheet print-outs and various types of report

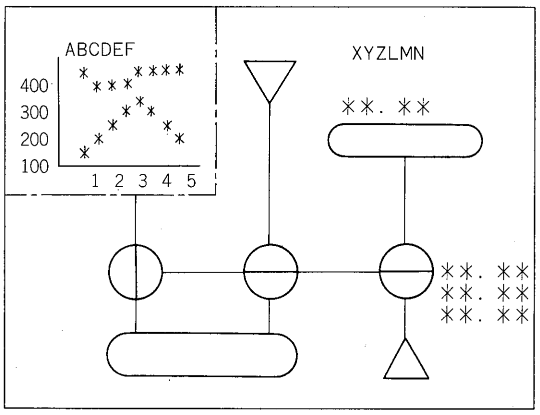


Fig. 3 **Subpicture**

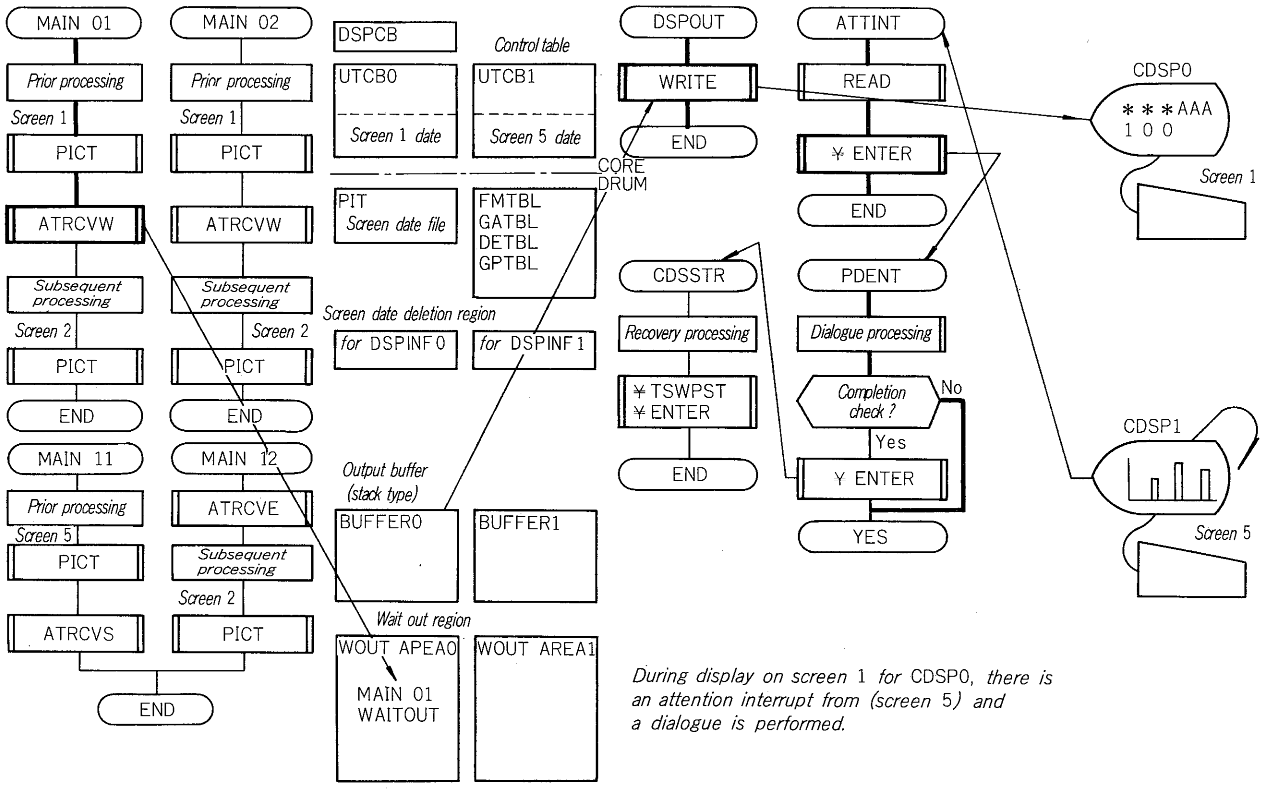


Fig. 4 Example of use of CSP

***Table 11* Subroutines of CSP**

Subroutine name	Entry name	Function
(1) Screen display subroutine	PICT	Displays total screen (master picture)
	SPICT	Displays small screen (subpicture)
(2) Message data display subroutine	DMSG	Message data are displayed at set position on screen
(3) Graph display subroutine	GRAPH	Received data are expanded to graph pattern and displayed
(4) Scroll-up mode display subroutine	SCROLL	Display in scroll-up mode
(5) Total screen release subroutine	PDLT	Total screen with no dialogue is released
(6) Small screen cancelation subroutine	SPDLT	Small screen is deleted
(7) Wait subroutine for attention interrupt for dialogue	ATRCV	"Attention" information is read in
	ATRCVW	Queuing until completion of dialogue
	ATRCVS	Preparation for dialogue
	ATRCVE	Checks equipment conditions at completion of dialogue
(8) Interaction subroutines	FLREAD	Reads in address on screen of light pen or code of function key
	CSREAD	Reads in contents of SEND botton or ID card
(9) Dialogue processing release subroutine	DLGEND	Performs dialogue stop or completion processing
(10) Debugging subroutine	CDSDMP	Starts analysis task of CSP control table

compilations for operator guidance. The tasks making up the LGS are controlled by the schedulers described in section 7. They perform data processing using the subroutine groups of sections 2 and 3 and perform print out by means of the OCP. In the LGS, macro-language can be used effectively in compilation of various types of tables and program. All tables can be written in macro-instructions without concerning the table construction. For example, simple compilation is possible even in such cases as in field data read-in processing programs where only the subroutine name used in the system is written in

Table 12 **LGS subroutines and programs**

Name of subroutine or program	Entry name	Function
(1) Field data processing program	DTAPM	Reads in analog input, converts to linearity and engineering units, checks upper and lower limits, etc. User describes by macro-language. Segmentation possible in accordance with size of multiple region
(2) Report processing program	PLOG	Performs report output data processing. Requests output in OCP
(3) Trend log start/stop subroutine	TLGOPN/TLGCLS	Starts or stops trend log for specified group
(4) Trend log point record /delete subroutine	TLGENT/TLGDLT	Records specified process variable or calculated variable to group specified in trend log or deletes from the group
(5) Trend log output program	TLGTSK	Gives trend log output for requested group
(6) Alarm summary output program	ALSUM	Prints out or displays in display equipment input checked out from results of validity check or alarm check
(7) Message request subroutine	MREQ	Gives out output request for alarm message, etc.
(8) Tag No. conversion subroutine	TAGNO	Converts tag No. to the data base No. corresponding to the input point or its reciprocal
(10) Scan ON/OFF subroutine	SCNON/SCNOFF	Instructs ON/OFF of scan at specified input
(11) Data base read-out subroutine	RWTBL	Reads out data base linked to specified input point

Table 13 **SECOP subroutines**

Subroutine name	Entry name	Function
(1) Staritng subroutine	SCSTAT	Starts sequence control from specified step
(2) Initial subroutine	SCINTL	Initializes control table
(3) Condition skip subroutine	SCSKIP	Neglects step completion conditions and advances to next step
(4) Stop subroutine	SCKILL	Stops control
(5) Step advance subroutine	SNOSI/SNOSE /SNOTS/SNOTSF /SNODC	Advances sequence step by specified method
(6) Joint resources control subroutine	SCCUTU	Controls resources used in common for several sequences
(7) Logic decision subroutine	SAND/SOR/SPAT	Checks establishment of sequence step completion conditions
(8) ON/OFF data conversion subroutine	SDIB/SDIBN	Reads in digital input from exterior and converts to ON/OFF data (work memory) for logical decision
	SAICH/AICL/AIE	Reads in analog input from exterior, performs upper/lower limit check, etc. at AD conversion value base and converts to ON/OFF data
	SFCHON/SFCHOF	Checks specified flag in flag region and sets results in work memory
	SIDCH/SIDCL/SIDE	Checks upper/lower limits etc. for engineering unit data in file and converts to ON/OFF data
	SVH/SVL/SVE	Obtains difference between former and present value for engineering unit data in file, checks upper/lower limits, etc. for this difference and converts to ON/OFF data
(9) ON/OFF data operation subroutine	SWOPEN/SWSPUT /SWRPUT/SWCLOS	Reads out and writes in ON/OFF data for logical decision in work memory
(10) Message output request subroutine	MSSG	Requests output of messages for operator guide

the macro-instructions. It is also simple to divide programs in cases where there are limits on the main memory region and to add processing specific to the user.

Table 12 gives an outline of the LGS subroutines and programs.

12. Sequence Control Program Package (SECOP)

The sequence control definition is not always clear for process computer systems. There is a hardware base sequence control which makes possible processing by means of a sequencer such as the USC-4000[3], on the other hand there is a sequence control for the system as a whole which is performed including data logging functions, various types of control functions and man/machine interface functions as in the case of large-scale batch processes. The SECOP is a subsystem which forms one of the computer control systems with evaluation functions through process data finishing and processing.

The SECOP offers the subroutine groups required for sequence control so that various types of sequence control can be performed. The user can combine these subroutines based on the control system.

Table 13 shows an outline of the SECOP subroutines.

Table 14 **Communication control program package**

Name of subroutine or program	Entry name	Function
(1) Communication control macro-instruction	RTCC	Gives out telegram transmission and reception requests. Execution is by DSPS or DSPR
(2) Telegram control macro-instruction	MCR	Telegram in transmission buffer is transferred to line buffer or telegram is transmitted from line buffer to reception buffer
(3) Contension process program	CCPS/CCPR	Telegram is transmitted or received by contension process
(4) Poling/selection process program	CCP	Telegram is transmitted or received by poling/selection process
(5) Transmission telegram program	MCPS	Telegram is taken from transmission telegram file, accomodated in transmission buffer and starts transmission process program
(6) Reception telegram program	MCPR	Telegram is removed from reception buffer, accomodated in reception telegram file and starts user telegram
(7) Telegram monitor print program	MMPS/MMPR	Telegram is taken from transmission telegram file or reception telegram file and monitored and printed
(8) Test program	TSTP	Hardware test is performed by terminal repeat in own bureau
(9) Device specific program	DSPS/DSPR	Reception and transmission by IOC-G
(10) Monitoring program	CTOP/DEMP	Time monitoring and device monitoring
(11) Interface macro-instruction	ISRC	Initial set of transmission telegram file or buffer
	OPNC	Line open processing
	CLSC	Line closed processing
	MPUT	Transmitted telegram recording
	MGET/MSTP	Obtaining received telegram
	TSTC	Own bureau terminal repeat test

13. Communication Control Program Package (CCPP)

The IOC-G are devices for communication control in the RTC. The IOC-G perform data transfer by the interlace mode. The CCPP is a package for control of the communication lines connected to the IOC-G. It consists of the subroutines and tasks shown in *Table 14*. The CCPP has the following functions and features:

(1) There are programs for the contention and polling/selection processes.

(2) Even when the same contention process is adopted, there are small differences in the detailed parts of the monitoring sequence, etc. but in the CCPP program, there is flexibility in respect to these differences.

(3) The interface macro for use of the CCPP from the application program has been considered so that the user does not need to worry about the special characteristics of the communication lines.

(4) Monitoring and printing of the transmitted and received telegrams is possible.

(5) A hardware check of control circuit is possible by means of return tests in each bureau.

(6) Control of more than one line is also possible.

14. Debugging Aid Routine

The following routines are used for debugging of the U-200 in the POPS:

1) Dummy RTC routine

These routines are software substitutes of the functions of each RTC device and the receive and transmit input/output data in a memory. In addition to effective debugging with the debugging machine, they can also be used in cases where no input signal is obtained and external input/output equipment can not be used. The calling procedure for the dummy RTC routines is the same as that for the subroutines with access to the original RTC so that no special considerations are necessary during program compilation. The dummy RTC routines print out access devices, data, etc. The data for these print-outs are stored once in the auxiliary memory and printed out by a separate task. In particular, since the dummy functions can be effective only when a specified task (more than one task can be specified) has access to a specified device (more than one is possible), they do not have major effects on system processing during system test runs or even during actual operation and checks of abnormal locations are possible. Print-out functions with the dummy RTC routines are also effective when there is actual access to the RTC, they can also be employed for tests of input signals from the exterior.

2) Tracers

The tracers are used for debugging of programs mainly written in assembler FASP and macro-FPL language. The tracer is called from the user program as a subroutine and it has the following functions:

(1) The contents of the register at the time of calling and the contents of the specified region are

printed out.

(2) The output can be in the hexadecimal, decimal, floating decimal point, character and bit string forms.

(3) For easy viewing of the trace list, any title in the list can be given out.

(4) The trace functions can be generated by switching on the flag in the control table.

(5) Trace functions can be generated only for specified multiple tasks.

(6) The trace list print-out is separated from the call source task and is given out by the "leave-behind" system.

(7) Because of (5) and (6), these routines can be used for checking of part of the program during both test runs and actual operation. If sufficient consideration is given during construction, these functions can show exceptional results in tracing the cause during accidents in the computer control system covering the plant as a whole.

The file utility described later can be used for input/output of the debugging data.

IV. FPL (FUJI PROCESS LANGUAGE)

1. Features of the U-200 Macro-Generator (MACGEN)

The following features of MACGFN are utilized in the FPL:

(1) Assembly language statements which differ according to the presence of macro-instruction operators, numbers and values can be formed.

(2) Part of the macro-instruction parameters can be incorporated in the statements formed.

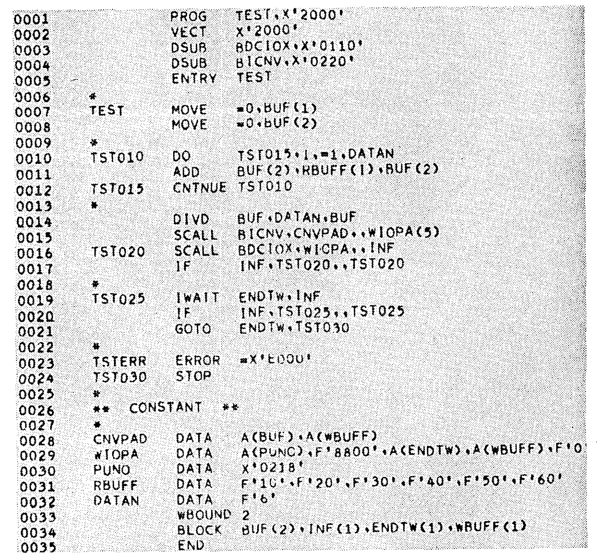
(3) Statements can be formed which differ according to the times when the macro-instructions are given in the programs.

(4) Information can be received or transmitted between macro definitions and other macro-definitions.

(5) Macro-instructions can be used within macro-definitions.

Table 15 Basic instruction set of FPL

Type	Instruction
(1) Symbol definition	EQU
(2) Compilation of constants	DATA, DBDATA, CHARCT
(3) Region assurance and boundary array	AREA, PRGCOM, SYSCOM, WBOUND
(4) Execution instructon	MOVE, ADD, SUB, MULT, DIVD, ADDD, SUBD, LAND, LOR, LEOR, SETB, RSTB, TSTB, BREAD, BWRITE, WCNVSD
(5) Control variable operation	SETX, STRX, ADDX, SUBX, MULTX, DIVDX, IFX, GOTOX, DO·········CONTNUE
(6) Program control and coupling	PROG, VECT, NOVECT, ENTRY, DSUB, SCALL, BLINK, COMRT, RETURN, END
(7) Control service	ENTER, STIMER, STOP, WAIT, SMODE, SLEVEL, ENQ, DEQ, ERROR, TIME etc.



```
                PROG    TEST,X'2000'
                VECT    X'2000'
                DSUB    BDCIOX,X'0110'
                DSUB    BICNV,X'0220'
                ENTRY   TEST
        *
        TEST    MOVE    =0,BUF(1)
                MOVE    =0,BUF(2)
        *
        TST010  DO      TST015,I,=1,DATAN
                ADD     BUF(2),RBUFF(I),BUF(2)
        TST015  CNTNUE  TST010
        *
                DIVD    BUF,DATAN,BUF
                SCALL   BICNV,CNVPAD,,WIOPA(5)
        TST020  SCALL   BDCIOX,WIOPA,,INF
                IF      INF,TST020,,TST020
        *
        TST025  IWAIT   ENDTW,INF
                IF      INF,TST025,,TST025
                GOTO    ENDTW,TST030
        *
        TSTERR  ERROR   =X'E000'
        TST030  STOP
        *
        **  CONSTANT  **
        *
        CNVPAD  DATA    A(BUF),A(WBUFF)
        WIOPA   DATA    A(PUNO),F'8800',A(ENDTW),A(WBUFF),F'0
        PUNO    DATA    X'0218'
        RBUFF   DATA    F'10',F'20',F'30',F'40',F'50',F'60'
        DATAN   DATA    F'6'
                WBOUND  2
                BLOCK   BUF(2),INF(1),ENDTW(1),WBUFF(1)
                END
```

Fig. 5 Coding example with FPL

2. FPL Instruction Sets

Table 15 shows the basic FPL instruction sets. *Fig. 5* is an example of a simple program using the FPL. As was described previously, the FPL can easily change or add instruction sets so that instructions systems suitable for describing the system software can be formed for each application system.

V. UTILITIES

1. File Utility

The file utility has a high rate of use during the debugging, system test and test run as well as in system maintenance after actual operation. The functions and simplicity of use of the file utilities have an important influence on the operation. *Table 16* gives an outline of the file utilities.

(1) Basic file utility

In the utility which specifies the region in the auxiliary memory by an absolute address, easy use is stressed and the control statement is very simple.

(2) Debugging aid file utility

This utility specifies the region in the files by means of parameters decided from the structure of each file as shown in *Table 2* so that input and output of data are easy during program debugging.

The features are as follows:

(1) There are many formats for expression of input/output data.

(2) Input data monitor can be given out.

(3) Explanatory sentences can be inserted in the output list.

(4) It is possible to dump the file data on paper tape, etc. and data used for single program debugging can also be used in system tests, etc.

2. Format Table Compilation Utility

When using the CSP or OCP, the user must prepare groups of tables specifying the detailed input/

Table 16 **File utility**

Classification	Program name	Entry name	Function
Basic file utility	Memory contents dumping program	DUMPS/DUMPP	Dumps data in main or auxiliary memories to system list or paper tape
	Hexadecimal loader	HDL	Hexadecimal data punched on paper tape is read into main or auxiliary memory
	Hexadecimal data comparison program	COMPS	Contents of paper tape and main or auxiliary memory compared
	Memory contents conversion program	ST/STN	Contents of main or auxiliary memory converted by given data
Debugging aid file utility	File accomodation program	FLST	Given data accomodated in specified file. Monitor output of input data possible
	File dumping program	FLDUMP	Data in specified file dumped in system list or on paper tape
	File restoration program	LOADF	Contents of paper tape punched by FLDUMP are restored to file
	File comparison program	FLCOMP	Comparison of file and paper tape contents
	File table conversion program	FLCONT	Change in file table

Table 17 **Format table utility for OCP and CSP**

Utility name	Entry name	Function
(1) Format statement translation program	FOTLAT	Format statement punched on paper tape or card is converted into object code
(2) Gathering statement translation program	GATLAT	Gathering statement punched on paper tape or cards is converted into object code
(3) Translation results dump program	FGDUMP	Object code compiled by FOTLAT or GATLAT is punched out on paper tape
(4) OCP format table recording program	FORLOD	FOTLAT output recorded as format table for OCP
(5) OCP gather table recording program	GATLOD	GATLAT output recorded as gather table for OCP
(6) OCP format table printing program	FORLST/MASLST	Contents of recorded format table are printed
(7) OCP gather table printing program	GATLST	Contents of recorded gather table are printed
(8) CSP screen table recording program	PICTLD	Outputs of FOTLAT and GATLAT are recorded in CSP screen table
(9) CSP screen table printing program	GFLST	Contents of recorded screen table are printed
(10) CSP screen table deleting program	PITDLT	Specified screen table is deleted

output specifications. *Table 17* shows the utilities for compilation, recording and control of these tables. Instruction groups which describe the format table (specifying the input/output format) and the gather table (specifying the location of the input/output data) which are important for the user are prepared and the object codes of the table can easily be compiled by means of the translation programs (FOTLAT and GATLAT).

VI. CONCLUSION

There are many new topics concerning the hardware of computer control systems. In the past, the progress and changes in hardware technology have been remarkable and such conditions will continue in the future. However in the software field, there have also been some developments. There have been many developments and proposals concerning OS and languages and standardization activities have considerable both in Japan and abroad. However, from the standpoint of the design and compilation of actual application systems, the majority of efforts must have been devoted to the realization of new possibilities resulting from the progress in hardware.

In the FACOM U-200 POPS, the subroutine groups, packages and utilities which form the heart of the process control system as outlined in the previous sections have been developed over the last one and one half years on the basis of Fuji Electric experience and in consideration of the future software requirements. The requirements for programming systems are limitless and efforts must continue in the future, although the foundations have already been laid. POPS is already in practical use. In relation to the actual application systems, there will no doubt still be demands for various types of improvements. Criticisms and opinions of users are welcome.

References:

(1) Oboshi, T. et al.: FACOM U-200 computer system, Fuji Electric Review, **20** (3), 1974.

(2) Japan Electronic Industry Development Association: Trends in Standardization of Industrial Computer Software (No. 3), 49-A-83, 1974.

(3) Shijima et al.: Universal Sequencer USC-4000: Fuji Journal, **45** (8), 1972.