DESIGN OF AN INTERACTIVE SYSTEM FOR PROCESSING PICTURES

by

Margaret Anita Turek

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREES OF

BACHELOR OF SCIENCE

and

MASTER OF SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 1974
(JUNE)

Signature of Author _____
Department of Electrical Engineering, February 8, 1974

Certified by _____ / Thesis Supervisor

Accepted by _____
Chairman, Departmental Committee on Graduate Students

DESIGN OF AN INTERACTIVE SYSTEM FOR PROCESSING PICTURES

by

Margaret Anita Turek

Submitted to the Department of Electrical Engineering on February 8, 1974, in partial fulfillment of the requirements for the Degrees of Bachelor of Science and Master of Science

ABSTRACT

The design described here is part of a large computer system, based on a PDP-11/40 computer, for processing pictures in digital form. The function of the software modules described here is that of communicating with the user and carrying out his commands. Using these modules a user may initiate predefined processes or define new processes for the system.

THESIS SUPERVISOR: Donald E. Troxel
TITLE: Associate Professor of Electrical Engineering

Acknowledgements

I would like to thank Charles Lynn for his valuable help with the entire project.

TABLE OF CONTENTS

	Page
Chapter 1 -- Introduction	6
1.1 An Overview of the System	6
1.2 Structure of the Software System	7
Chapter 2 -- A Detailed Description of MENU and PROCES Tasks	12
2.1 The Structure of Information	12
2.2 Initializing the MENU Task	13
2.3 Using the MENU Task	13
2.4 Using the PROCES Task	15
2.5 An Example of a Process Specification	20
Chapter 3 -- Design Considerations	29
Appendix -- File and Data Set Formats	33
A.1 The HELP.ASC File	33
A.2 The MENU.ASC File	33
A.3 The PTLIST.ASC File	33
A.4 The NAME.DES File	34
A.5 The NAME.SYM File	34
A.6 Task Control Block Variables Visible to the User	36
A.7 Data Types	37
A.8 The NAME.TCB File	39
A.9 Task Program Format	41
References	43

LIST OF FIGURES

		Page
1.1	The structure of an active process	10
2.1	A dialogue with MENU and PROCES tasks	21
2.2	Schematic structure of process NEWPRO	24
2.3	First version of NEWPRO.SYM	25
2.4	Second version of NEWPRO.SYM	27
A.1	NAME.TCB file format	40

Chapter 1 - Introduction

1.1 An Overview of the System

The design described here is an integral part of a large multiprocessing picture handling system. The system is being designed for the Associated Press to handle digital transmission of news pictures. It consists of a PDP-11/40 computer, a number of peripheral devices for reading, writing, displaying, storing and transmitting pictures and a software system that will communicate with the user and supervise the activities of the entire network.

The hardware of the system consists of numerous devices. The user communicates with the system through a VT05 terminal, which consists of a keyboard and a CRT display. The system's main secondary storage device is a disk. Additionally, data can also be stored on Dectape or magnetic tape. The picture transmitters and receivers used have recently been developed and are based on laser scanning. These machines, called Laserphoto, are capable of scanning 100 lines per inch and offer substantial improvements in the cost/performance ratio. The TV monitor used for displaying pictures has a semiconductor memory capable of storing a full frame consisting of 256 by 256 picture elements, with four bits of memory allocated to each element. A line printer and a paper tape reader and punch are also available.

The bulk of the system's operations will consist of receiving and transmitting pictures which will be accomplished

automatically. The PDP-11/40 computer, located in New York, will communicate via long distance telephone lines with transmitters and receivers located in several other cities. It will, therefore, be possible to transmit a picture from any location to the main computer which in turn will automatically transmit it to all interested parties. Additionally, a user of the system communicating through a VT05 terminal will be able to issue commands to display pictures on the TV monitor, crop, enlarge, reduce, perform various tone scale manipulations, add captions, and transmit pictures.[1] The list of operations on pictures is not completely defined at this point, and it is not necessary to do so. The system is designed in such a way that it can be initialized and used with a minimal number of procedures. Other procedures can be defined and added to the system as need arises without interrupting its operation.

1.2 Structure of the Software System

The software system will consist of the standard PDP-11 monitor, a special supervisor, known as the AP supervisor, and various tasks and processes. The AP supervisor will perform many monitor functions such as free storage management, scheduling of processes, communication among tasks as well as communication between processes and the system, and trap handling. By using traps, a process can request services from the supervisor such as I/O operations or storage

allocation, and signal "error" and "process completed" conditions. The AP supervisor is necessary since the above functions are not available under the standard PDP-11 monitor. Whenever possible the standard DEC disk operating system monitor is used.

The entities executed under this supervisor are called processes. They generally represent complete operations on pictures such as reading a picture into memory, transforming it in some way, and writing it on a secondary storage device. The components of a process are called tasks. A typical process might consist of a control task, an input task, a transform task and an output task. Each task is a program written in pure code, i.e. in such a way that no data is stored in the program itself and therefore several processes can timeshare one copy of a task program in core. Any data or pointers to data belonging to a particular instance of a task is stored in the task's task control block, the TCB. Each active process, therefore, will correspond to a number of task control blocks in core containing all the information unique to that process.

Data being operated on by a process, usually picture lines, will be passed between the TCB's of a process by the supervisor. The TCB's of a process contain pointers telling the supervisor in what order the tasks contained in the process will operate on data. The T.DAD pointer in a TCB points to the task's father, the task from which the current

task expects data. The T.BRO pointer indicates the task's brother, the task that is to receive the same data as the current task. The T.SON pointer indicates which task is to receive the current task's output data. When a process is being activated, the supervisor receives a pointer to the TCB block, a single block of memory containing a TCB for each task in the process. Using the information contained in the TCB block the process can be executed. When a process is completed, the supervisor notes this and purges the process by deleting the TCB block from its list of active processes and returning it to free storage.

The relationships between tasks, task control blocks, and some variables in the TCB's are illustrated in Figure 1.1. TCB's one, two and three make up one process consisting of tasks one, two and three. TCB's four and five represent a second process consisting of tasks one and four. Task one, therefore, is shared by both processes.

Two tasks, the MENU task and the PROCES task, serve special purposes even though they do follow the above rules. The MENU task always displays the menu, a list of processes currently defined and available for activation. The menu display also indicates what parameters must be provided to activate each process. Given a user request for process activation consisting of the process name and parameters, it is the job of MENU task to convert this request into a TCB block and pass it to the supervisor for execution.

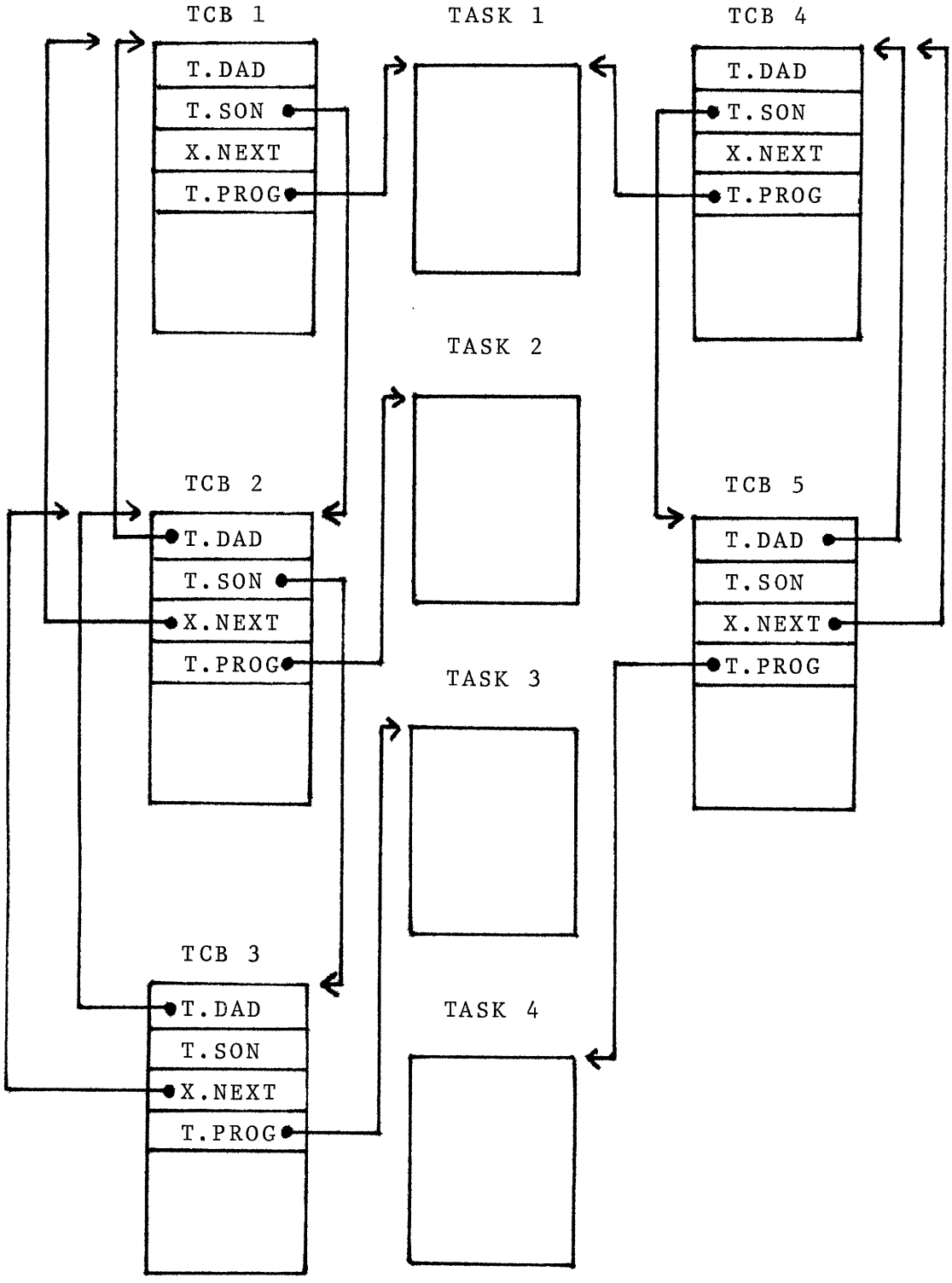

Figure 1.1 - The structure of an active process

The main job of PROCES task is that of expanding the capabilities of the system by adding new tasks and processes. A program that is to become a new task must follow certain rules and must be previously written and assembled elsewhere. After it is added to the system it can be used as part of a new process. The creation of a new process involves specifying exactly how a number of existing tasks should interact. This description is checked, processed and stored by PROCES task. When that is completed, the new process will appear on the menu display and can be activated just like any other process. The number of processes and tasks that can be defined in this manner is limited only by the amount of storage space available on disk. The design and use of these two tasks is the topic of this thesis.

Chapter 2 - A Detailed Description of MENU and PROCES Tasks

2.1 The Structure of Information

All the information concerning currently available tasks and processes is contained in a number of files residing permanently on disk. These files can be accessed through MENU task, but can only be altered by using PROCES task. This set of files consists of the following:

MENU.ASC - contains the menu display

PTLIST.ASC - an ASCII file containing a list of all processes and tasks; it is referenced by PROCES task to determine whether or not a given process or task exists

HELP.ASC - file introducing a new user to the use of the system

NAME.DES - the user must create a NAME.DES file for each new process or task he defines; it should contain a short description of the process or task

NAME.BIN - file containing the actual assembled task program

NAME.SYM - file describing a process to PROCES task

NAME.TCB - file containing information about process NAME; used by the MENU task to activate the process; created by PROCES task by transforming the NAME.SYM file

Formats of all of the above files are described in detail in the appendix.

2.2 Initializing the MENU Task

The MENU task is designed in such a way that no initialization procedure is necessary. The following files are assumed to exist on disk: ASCII files HELP.ASC, MENU.ASC and PTLIST.ASC and assembled tasks MENU2 and PROCES. MENU1 should be loaded and execution started at relative location zero. Current contents of the MENU.ASC file will be displayed and MENU will wait for a request from the user. The user can now add new tasks and define new processes by using PROCES task. MENU task can be initialized in this manner both as a minimal system described above and after several tasks and processes have been added to it, provided that the files on disk created by PROCES task have not been altered or deleted.

2.3 Using the MENU Task

The purpose of the MENU task is to provide the user with information about the system and to activate processes. Following every user request, MENU task will display the menu, execute the request, display an error message if necessary, and then wait for another request. User requests will appear on the general communications area of the screen. All displays will appear on another area of the screen, especially reserved for that purpose, and will remain there until a new display is shown. The menu is a list of processes currently available for activation. Each line of the menu display will have the following format:

NAME $P_1,P_2,\ldots,P_M$; $Q_1=V_1$, $Q_2=V_2,\ldots,Q_N=V_N$

where NAME is the name of a process, P's are descriptive names of required parameters (parameters that must be specified to activate the process), Q's are descriptive names of optional parameters, and V's are default values of optional parameters.

The following commands are available to the user under MENU task:

HELP

HELP is a request to display the HELP.ASC file, a file that lists and describes the commands available under MENU.

HELP NAME

This is a request to display the NAME.DES file, a file containing a description of a process named NAME.

DISPLAY NAME.EXT

This is a general request to display an ASCII file named NAME.EXT residing on disk.

NAME ARGUMENT-LIST

This is the basic command format for activating a process. NAME is the name of a process listed on the menu display and must be followed by at least one space. ARGUMENT-LIST contains values for required and optional parameters. Required parameter values must appear first as a string of values separated by commas. The values must appear in the order specified on the menu display. A semi-

colon indicates the end of the required parameter string. Optional parameters follow, in the order specified on the menu, as another string of values separated by commas and followed by a carriage return. Any number of optional parameters and any number of commas at the end of the line may be omitted. Default values will be used in place of the missing optional parameters.

In addition to activating processes through the user terminal, processes can also be activated internally by other tasks. This is accomplished by creating another instance of the MENU2 task. A single TCB must be created and the following TCB variables must be set:

T.PROG - pointer to the program control block (PCB) of the MENU2 task

X.LNTH - number of words allocated for the TCB (must be a multiple of four)

T.NARG - number of arguments to the task (must be set to one)

T.PAR1 - a pointer to a user line, an ASCII file formatted exactly as a line buffer containing a user request to activate the process

In order to activate the process, this TCB must be passed to the AP supervisor by using the M.PROG trap.[2]

2.4 Using the PROCES Task

The main job of the PROCES task is expanding the system by adding new tasks and processes. In all respects, PROCES

task conforms to the rules set for all tasks. It is designed to be executed as a one task process with no parameters. It is therefore invoked simply by typing "PROCES" when MENU task is in control. Once PROCES task is in control it will identify itself by displaying the words "PROCES TASK" on the screen. To indicate that it is ready to accept a new command, it will display a "#" and wait for the user to type a command on the keyboard. Whenever an error is encountered, an error message will be displayed and PROCES task will wait for the next command. The following commands are available under PROCES task:

HELP

Same as under MENU task, see section 2.3.

HELP NAME

Same as under MENU task, see section 2.3.

DISPLAY NAME.EXT

Same as under MENU task, see section 2.3.

EXIT

This request terminates PROCES task and returns control to MENU task.

TASK NAME [ON DEVICE]

This is a request to add a new task to the system. The task program and the NAME.DES file (see section A.4) must be created prior to this command. The task program must conform with all the rules in section A.9, and must be assembled in a file named NAME.BIN. Both NAME.DES and NAME.BIN must be on

Dectape zero if no device is specified in the request, or else on the device specified (see section A.7).

CREATE NAME

This is a request to create a new process named NAME. It is assumed that NAME.DES (see section A.4) and NAME.SYM files have been created prior to this request and are residing on disk. The NAME.SYM file is checked and the NAME.TCB file created. The process is added to the MENU.ASC and PTLIST.ASC files. If any errors are encountered, the process is not created and an error message is displayed.

The NAME.DES file describes the new process in a free format for the benefit of a future user of the process. The NAME.SYM file, on the other hand, has to be much more precise since it is interpreted by PROCES task and a process is created according to it.

As previously described, a process consists of a number of tasks and task control blocks, one TCB for each task in the process. A process is specified by creating the appropriate TCB's. The function of the NAME.SYM file is to describe the TCB's of a process by specifying how many are involved and what values the TCB variables in various TCB's should contain.

Six of the TCB variables, T.PROG, T.NARG, X.NEXT, T.DAD, T.SON, and T.BRO, describe the structure of the process. These variables must be set permanently when the process is created. The user activating the process at a later time

will have no access to these variables. The other TCB variables that can be set by the user, T.DEV, T.PRTY, T.DIR, and T.PAR1 through T.PAR5, may be assigned values at any time. The description of a task, the NAME.DES file, should be checked to see which variables in the task's TCB ought to be assigned values at all. If a variable is to be assigned a value, it must be classified as a constant, a required parameter or an optional parameter. In the first case, a constant value must be assigned at process creation time. In the second case, the user will be required to supply a value at process activation time. In the last case, a default value is provided at process creation time. At activation time, the value can be changed if so desired. The exact format of this specification is described in section A.5. Every TCB variable that can be assigned a value at activation time must be assigned to a parameter number. These numbers correspond to parameters provided by the user at process activation time. The exact type, range of permissible values and purpose of each parameter must be described, by the person creating the process, in the NAME.DES file.

The six variables describing the structure of a process must be specified for each TCB. The T.PROG entry specifies the task that will use the TCB. The T.NARG variable tells how many parameters the task specified in the T.PROG entry expects. The four remaining variables specify how the tasks

of the process will interact.

A task receives its input from its father (T.DAD pointer in the task's TCB) and passes its output on to its son (T.SON pointer in the task's TCB). Brothers, tasks linked by T.BRO pointers, always receive the same input. If a task has several sons, it will point to one of them by using its T.SON TCB entry. The remaining sons will be chained through their T.BRO pointers, starting with the T.BRO pointer in the TCB pointed to by T.SON. The T.DAD, T.SON, and T.BRO pointers should form a tree. The tree should connect all the TCB's in the process. Care must be exercised to prevent loops in a process, cases where a piece of data could be passed repeatedly between a group of TCB's. Only tasks expecting input data should have fathers, and, similarly, only tasks producing data should have sons.

The X.NEXT pointers in TCB's link all TCB's of a process into one chain, which is used by the scheduler to locate a task to be executed. These pointers should link the TCB tree from the bottom up for most efficient execution. This means that an X.NEXT pointer may point to a TCB's father or brother, but never a son.

PURGE NAME

This is a command to purge a process of a task from the system. Purging involves deleting all references in PTLIST. ASC and MENU.ASC as well as deleting NAME.BIN, NAME.SYM,

NAME.DES, and NAME.TCB files from the disk. This command should be used with caution since a purged process or task cannot be easily recreated. As a safety precaution, after this command is issued, the user will be asked to confirm his request. A "YES" answer will activate the process, but any other character string will cancel the request.

STORE NAME

This command is used when a currently existing process is being altered. It assumes that one of the files describing process NAME (NAME.DES or NAME.SYM) has already been changed. MENU.ASC and NAME.TCB files are updated.

2.5 An Example of a Process Specification

In order to clarify the use of the MENU and PROCES tasks this section presents an example of a dialogue between a user and the system. The process and the individual tasks described here do not exist in reality, but they do follow all the rules described previously and should provide a comprehensive example. The commands described here are also shown in Figure 2.1. The underlined words are generated by PROCES task and everything else is typed by the user.

Let us assume that the four tasks, CNTRL, INPUT, TRANS, and OUTPUT, have already been created. One can check if this is true by typing "DISPLAY PTLIST.ASC". This request will display a complete list of currently available tasks and processes. To get more information about a particular task or process, say CNTRL, one can type "DISPLAY CNTRL.DES"

```
DISPLAY PTLIST.ASC

HELP CNTRL

DISPLAY INPUT.DES

EDIT
:
:
:
:
:

PROCES

PROCES TASK

#

CREATE NEWPRO

error message

#

EXIT

EDIT
:
:
:
:
:

PROCES

PROCES TASK

#

CREATE NEWPRO

#

EXIT

NEWPRO GIRL
:
:
:
:
```

Figure 2.1 - A dialogue with MENU and PROCES tasks

or "HELP CNTRL". Either one of the above requests will display the contents of the file CNTRL.DES. Let us assume that the four tasks of interest are partially described as follows:

CNTRL.DES

This task requires no inputs and generates a monotonically increasing string of integers in T.OUT. The constant increment for successive integers is taken from T.PAR1. No other parameters are used.

INPUT.DES

This task expects a picture directory block pointer in its T.DIR TCB entry and a line number as input in T.IN. The desired picture line is read and a pointer to it is entered into T.OUT. No parameters are required.

TRANS.DES

This is a task that transforms picture elements using numerical parameters provided in T.PAR1 and T.PAR2. No other parameters are used. T.IN is assumed to contain a pointer to a picture line, and T.OUT becomes a pointer to a transformed line.

OUTPUT.DES

This task outputs pictures. It expects a pointer to a picture directory block in T.DIR and a pointer to the current picture line in T.IN. No outputs are produced and no parameters needed.

Given the above information, let us now create a process named NEWPRO. NEWPRO will read a picture, transform it in two different ways by using two instances of the TRANS task, and write the two new pictures on disk. The schematic view of the process is shown in Figure 2.2. Vertical arrows indicate that the output of one task becomes the input of another one. A horizontal arrow indicates that two tasks receive the same input. As described above, CNTRL task does not receive any input from any other task and both output tasks produce no output for other tasks. Figure 2.3 shows a possible NEWPRO.SYM file. The first three lines of the file indicate that the process consists of six tasks, has one required and three optional parameters and 37_8 constants. The total number of variables and constants given must equal the number of assignment statements that follow. In the case of NEWPRO, the first ten assignments specify the relationships shown in Figure 2.2 and determine the flow of data in the process. The next five assignments chain all tasks in an order reverse to the data flow, as required. The next three lines describe the control task. T.PAR1, the increment, will get its value from parameter four, if specified, or else will have the value two. The input task expects all information about the picture, its name and, optionally, device, from parameter one. The two identical transform tasks are invoked with two different sets of arguments to produce two different transformations

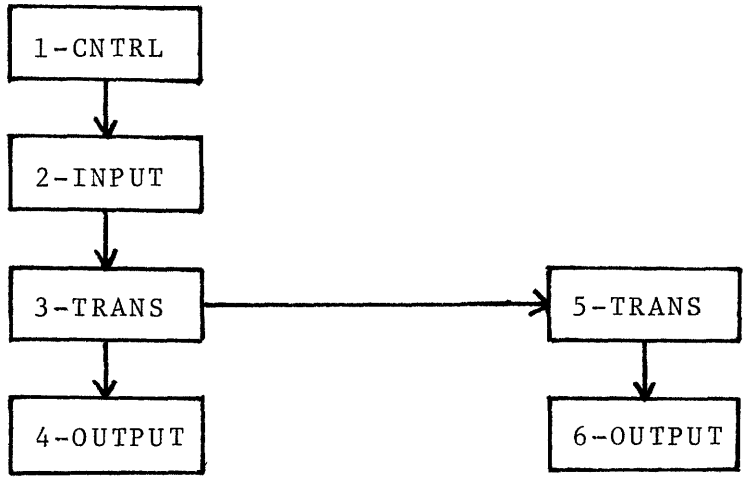

Figure 2.2 - Schematic structure of process NEWPRO

```
TASKS = 6
VARIABLES = 1,3
CONSTANTS = 37
1:T.SON = NU:2
2:T.DAD = NU:1
2:T.SON = NU:3
3:T.DAD = NU:2
3:T.BRO = NU:5
3:T.SON = NU:4
4:T.DAD = NU:3
5:T.DAD = NU:2
5:T.SON = NU:6
6:T.DAD = NU:5
6:X.NEXT = NU:4
4:X.NEXT = NU:5
5:X.NEXT = NU:3
3:X.NEXT = NU:2
2:X.NEXT = NU:1
1:T.PROG = TA:CNTRL
1:T.NARG = NU:1
1:T.PAR1 = NU:PAR 4, 2
2:T.PROG = TA:INPUT
2:T.NARG = NU:0
2:T.DIR = PI:PAR 1
3:T.PROG = TA:TRANS
3:T.NARG = NU:2
3:T.PAR1 = NU:177770
3:T.PAR2 = NU:125252
5:T.PROG = TA:TRANS
5:T.NARG = NU:2
5:T.PAR1 = NU:177700
5:T.PAR2 = NU:52525
4:T.PROG = TA:OUTPUT
4:T.NARG = NU:0
4:T.DIR = PI:PAR 2, NEW PICA
6:T.PROG = TA:OUTPUT
6:T.NARG = NU:0
6:T.DIR = PI:PAR 3, NEW PICB
```

Figure 2.3 - First version of NEWPRO.SYM

of the same input picture. In this case, the TRANS parameters have been specified as constants and therefore cannot be altered at activation time. Finally, the two output tasks will create pictures named PICA and PICB on disk unless something else is specified in parameters two and three at activation time.

Figure 2.4 shows a shorter version of NEWPRO.SYM. This version contains exactly the same information as the version in Figure 2.3. It has been shortened by using multiple assignments on a single line. Note that to do this the numbers of constants and variables must be adjusted. The two versions will produce identical TCB blocks. The first version of NEWPRO.SYM may be easier to read, but the second version will produce a shorter NEWPRO.TCB file.

In order to completely define NEWPRO, a NEWPRO.DES file must be created. The first line of NEWPRO.DES, which will also become a line of the menu display, might look as follows:

NEWPRO PICTURE; OUTPICTURE1, OUTPICTURE2, LINE-INCREMENT

The remainder of the NEWPRO.DES file would describe the meanings of the parameters named above and the function performed by NEWPRO.

NEWPRO.DES and NEWPRO.SYM must first be created using an editor. Once that is done, PROCES task can be invoked to create the process. In the example shown, an error is detected during the first attempt to create NEWPRO. To correct it,

```
TASKS = 6
VARIABLES = 1,3
CONSTANTS = 17
1:T.SON, 3:T.DAD, 5:T.DAD, 3:X.NEXT, 3:T.NARG, 5:T.NARG = NU:2
2:T.DAD, 2:X.NEXT, 1:T.NARG = NU:1
2:T.SON, 4:T.DAD, 5:X.NEXT = NU:3
3:T.BRO, 6:T.DAD, 4:X.NEXT = NU:5
3:T.SON, 6:X.NEXT = NU:4
5:T.SON = NU:6
1:T.PROG = TA:CNTRL
1:T.PAR1 = NU:PAR 4,2
2:T.PROG = TA:INPUT
2:T.NARG, 4:T.NARG, 6:T.NARG = NU:0
2:T.DIR = PI:PAR 1
3:T.PROG, 5:T.PROG = TA:TRANS
3:T.PAR1 = NU:177770
3:T.PAR2 = NU:125252
5:T.PAR1 = NU:177700
5:T.PAR2 = NU:52525
4:T.PROG, 6:T.PROG = TA:OUTPUT
4:T.DIR = PI:PAR 2, NEW PICA
6:T.DIR = PI:PAR 3, NEW PICB
```

Figure 2.4 - Second version of NEWPRO.SYM

EDIT task is invoked again. After correcting the error and invoking PROCES task the second time, NEWPRO is created successfully and control is returned to MENU task. In order to activate NEWPRO, any one of the following commands may be used:

```
NEWPRO OLD GIRL
NEWPRO OLD GIRL; NOISE1, NOISE2
NEWPRO OLD GIRL;,,5
NEWPRO OLD GIRL; DEFAULT, DEFAULT
```

In the first case, default values will be used for the number of picture lines to be skipped and the names of the two new pictures. In the second case, pictures NOISE1 and NOISE2 will be created by using every other line (the default value) of picture GIRL. In the third case, default picture names will be used, and only every fifth line of the original picture will be included. Finally, in the fourth case, random names will be generated for the two new pictures.

Chapter 3 - Design Considerations

MENU and PROCES tasks have been designed with two opposing objectives in mind, flexibility and simplicity. Maximum flexibility is necessary to enable the user to create any kind of a process which may be needed. Since it is unknown what the specific needs of future processes will be, PROCES task has been made flexible enough to accept just about anything. It is also necessary to make the system simple for the unsophisticated user who is not familiar with the internal details of the system. A compromise has been reached. The MENU task is extremely easy to use and is self-explanatory even to someone with no previous experience with computers. Adding new processes requires a rather thorough understanding of the structure of the system, but in return provides a lot of flexibility.

Originally the MENU task was going to have a question and answer format. The user would be asked to name a process. After naming one from a list presented to him, he would be asked to choose values for the process's parameters in the same manner. This approach requires short, well defined sets of values for each parameter, a very limiting restriction. Another alternative, explaining to the user what is wanted without listing all the possible values, would limit the number of parameter categories, which is also undesirable. In the present system, the person specifying a process can

ask for just about anything as a parameter value, and, by using the NAME.DES file, can explain to the user exactly what is needed.

PROCES task assumes the availability of an editor under the AP system for the creation of NAME.DES and NAME.SYM files. No editor has been included in PROCES task since an editor ought to be a separate task, available independently of PROCES task. The availability of an assembler (under the AP system or elsewhere) is also assumed for assembling new tasks.

PROCES task performs a certain amount of checking of the process specification provided. It checks if data types specified are compatible with the TCB variables they are assigned to; if X.NEXT pointers actually do form a chain; whether the value of T.NARG actually equals the number of arguments specified for each task; if all T.PROG entires are defined; and a few other things. There are, however, two areas left where the user can make mistakes and PROCES task cannot check them. First, PROCES task cannot check if a process specification is compatible with the way the tasks have been specified. A task may require, for example, two parameters and an input picture line from its father. PROCES task cannot check if requirements of this type have been satisfied. Currently, errors of this kind will not show up until execution time. This problem can be remedied by specifying an additional file to be created for each task.

This file would contain information, in a format understandable to PROCES task, showing which variables in the TCB for this task must be assigned data and the type of data required in each case. The second area concerns the accuracy of the description of the process, the NAME.DES file. PROCES task cannot possibly check if the NAME.DES file is correct since it is written in English and does not follow any strict format.

Currently PROCES task places a restriction on processes by not allowing more than sixteen tasks in any one process. It is very likely that more tasks per process will never be needed. In case the need does arise, the restriction can be lifted easily. The three routines checking X.NEXT, T.DAD, T.BRO and T.SON pointers in a newly created NAME.TCB file use sixteen bit words for the check. Removing these routines or rewriting them to use two sixteen bit words instead of one will solve this problem.

Another possibly undesirable restriction is the assumption that all TCB's are the same size and thus the task control block cannot have more than five parameters. In order to allow a variable number of task parameters and therefore variable size TCB's several changes would have to be made in PROCES task. In the MENU task only the calculation of the amount of memory necessary for the TCB block would need to be changed.

MENU task in its current form actually consists of two separate routines, MENU1 and MENU2. MENU1 is a very short

routine permanently residing in core. It does two things: displays the menu and, whenever a user types a command, MENU1 activates MENU2 with the user request as an argument. MENU2 analyzes user requests and carries them out. This arrangement serves a two-fold purpose. First, it allows any active process to activate MENU2 and therefore any process in the system can issue a pseudo user request to activate any other process. The procedure that must be followed to activate MENU2 is described in section 2.3. Second, it uses less memory by only keeping a small routine in core permanently and reading the large MENU2 routine into core memory only when needed.

Appendix - File and Data Set Formats

The following are the exact formats of all files and other structures relevant to MENU and PROCES tasks. Please note that files in memory are always assumed to be preceded by a standard three word PDP-11 DOS line buffer header, not included in the descriptions below. A pointer to a file, therefore, points to the header, and data is assumed to follow immediately after the header.

A.1 The HELP.ASC File

The HELP.ASC file contains a short description of commands available under MENU. It is intended to help a new user in getting acquainted with the system.

A.2 The MENU.ASC File

MENU.ASC is a file containing the menu display. Each line of the display describes a different process. The entries have the following format:

NAME P_1,P_2,...,P_N; Q_1=V_1, Q_2=V_2,..., Q_M=V_M

where NAME is the name of a process, P's are descriptive names of required parameters, Q's are descriptive names of optional parameters, and V's are default values of optional parameters. Every time a process is created or purged, the menu display is altered accordingly.

A.3 The PTLIST.ASC File

PTLIST.ASC contains a list of all processes and tasks currently available. The entries consist of the name of a

process or a task followed by a "P" or a "T", respectively. This file is also updated whenever a process or task is created or purged.

A.4 The NAME.DES File

A NAME.DES file, where NAME is the name of the process or task, must be created by the user for every process or task. It is an ASCII file and must contain all the information a future user of the process or task may need. In the case of a process, the first line of this file becomes the menu display description of the process and therefore should follow exactly the format specified in section A.2. The body of the file should contain a concise description of the function performed, names, types, default values and use of all parameters, input/output devices that can be operated upon, and finally, in the case of a task, a description of inputs expected from, and outputs produced for, the use of other tasks.

A.5 The NAME.SYM File

The NAME.SYM file contains a precise definition of a process. Note that all numbers in this file are assumed to be octal. The first line of the file must contain "TASKS = N" where N is the number of tasks and therefore also the number of TCB's involved in the process. Next line contains "PARAMETERS = P,Q" where P is the number of required parameters and Q is the number of optional parameters. Next line must contain "CONSTANTS = R" where R is the number

of statements below assigning constant values. This is followed by assignments of values and parameter numbers to TCB variables. The assignments may come in any order, one per line. The number of assignments must equal P+Q+R. The following are the three allowable assignment formats for required parameters, optional parameters, and constant values, respectively.

```
N: VARIABLE = TYPE: PAR X
N: VARIABLE = TYPE: PAR Y, VALUE
N: VARIABLE = TYPE: VALUE
```

N is a task number, VARIABLE is the name of a TCB variable (see section A.6), TYPE is a data type (see section A.7), X is the number of a required parameter to the process, Y is the number of an optional parameter to the process, and VALUE is the actual value of the appropriate type (see section A.7). Two or more assignments of the same value can be combined as follows:

```
2: T.DAD = NU: 3
3: X.NEXT = NU: 3
```

is equivalent to

```
2: T.DAD, 3: X.NEXT = NU: 3
```

Any number of variables can appear on the left side of an assignment, and the "PARAMETERS = P,Q" and "CONSTANTS = R" lines must be specified accordingly. Parameter numbers must be consecutive, with the required parameters being numbered 1 to P and optional parameters numbered P+1 to Q.

A parameter may only appear in one assignment statement.

A.6 Task Control Block Variables Visible to the User

Each TCB variable occupies one word of storage and, unless otherwise specified, has the value of zero.

T.DEV

This variable contains a system pointer to a device control block for a task which will do input/output operations. A device type value may be assigned to T.DEV at process creation time or at activation time.

X.NEXT, T.DAD, T.SON, T.BRO

These variables contain offsets to other TCB's in the process. Numerical values not greater than the number of tasks in the process are assigned to these variables at process creation time. These values are then converted to offsets at process activation time. The values should not be changed at activation or execution time.

T.PROG

This variable contains a program identification code. Task name values must be assigned to T.PROG at creation time. The value should not be altered at process activation or execution time.

T.STAT

This variable contains status flags used for communication between the task and the system.

T.PRTY

T.PRTY contains a numerical priority of the task.

Priority can be set and altered at any time.

T.IN

T.IN contains a pointer to a buffer containing an input picture line. This variable contains a value at execution time only and is set by the supervisor.

T.OUT

T.OUT contains a pointer to a buffer containing an output picture line. This variable has a value at execution time only, and must be set by the task.

T.TASK

This variable may be used in any manner by the task during execution.

T.NARG

T.NARG contains a number specifying the number of arguments to the task. This value must be set at process creation time.

T.PAR1 through T.PAR5

These variables contain the values of the parameters of the task. Values can be assigned at process creation or activation time and should be of the type expected by the task.

A.7 Data Types

The system recognizes five different data types, numerical data, picture names, device names, task names and file names.

Numerical data (type 0) is indicated by "NU" and accept-

able values are any octal numbers, positive or negative, that can be stored in a sixteen bit word. Numbers preceded by a minus are stored in two's complement notation.

Picture name data (type 1) is indicated by "PI". Acceptable values consist of either the word "OLD" or the word "NEW" followed by a picture name and, optionally, by a device specification consisting of the word "ON" and a three character device name. The word "OLD" indicates that the picture already exists while "NEW" indicates that the picture must be created and should not exist on the specified device. Picture names must consist of one to six alphanumeric characters. The only exception is the picture name "DEFAULT" which indicates that a random picture name should be generated If a device is not specified, disk 0 is assumed.

Device type data (type 2) is indicated by "DE". Acceptable device names are any of the following, with an appended digit if several devices of a kind are attached to the system: DK, DT, FXR, FXT, KB, LP, MT, SC, SI, TV, and SY.

Task names (type 3) are indicated by "TA" and their values must be the names of existing tasks.

File name parameters (type 4) are indicated by "FI" and their values must consist of a file name (up to six alphanumeric characters) followed by a period followed by a file name extension (up to three alphanumeric characters). This may optionally be followed by a device specification consisting of the word "ON" followed by a three character device

name. The default device is disk 0.

A.8 The NAME.TCB File

The user of the system will not need to concern himself with the structure of this file since it is produced by PROCES task for the use of MENU task only. The description is included here for completeness. A schematic structure of the file is shown in Figure A.1. The "Number of TCB's" entry specifies the number of TCB's and therefore the number of tasks in the process. "TCB block offset" is a byte offset of the TCB block from the beginning of the NAME.TCB file header. "First logical TCB" is the number of the first TCB in the X.NEXT chain and need not be the first physical TCB within the TCB block. "Offset of variable descriptions" is a byte offset of variable descriptions from the beginning of the file. Following these four words are six word descriptions of constants and required and optional parameters. Each six word description starts with a data description word. The left byte of this word is a one in the case of optional parameters, zero otherwise. The right byte of this word indicates the data type. The next four words contain the value, a constant value in the case of constants, a default value in the case of optional parameters and no value in the case of required parameters. These values are updated using the user's request line, before being inserted into the TCB block. Finally, the last word of each description contains the destination of the data, a byte offset from the

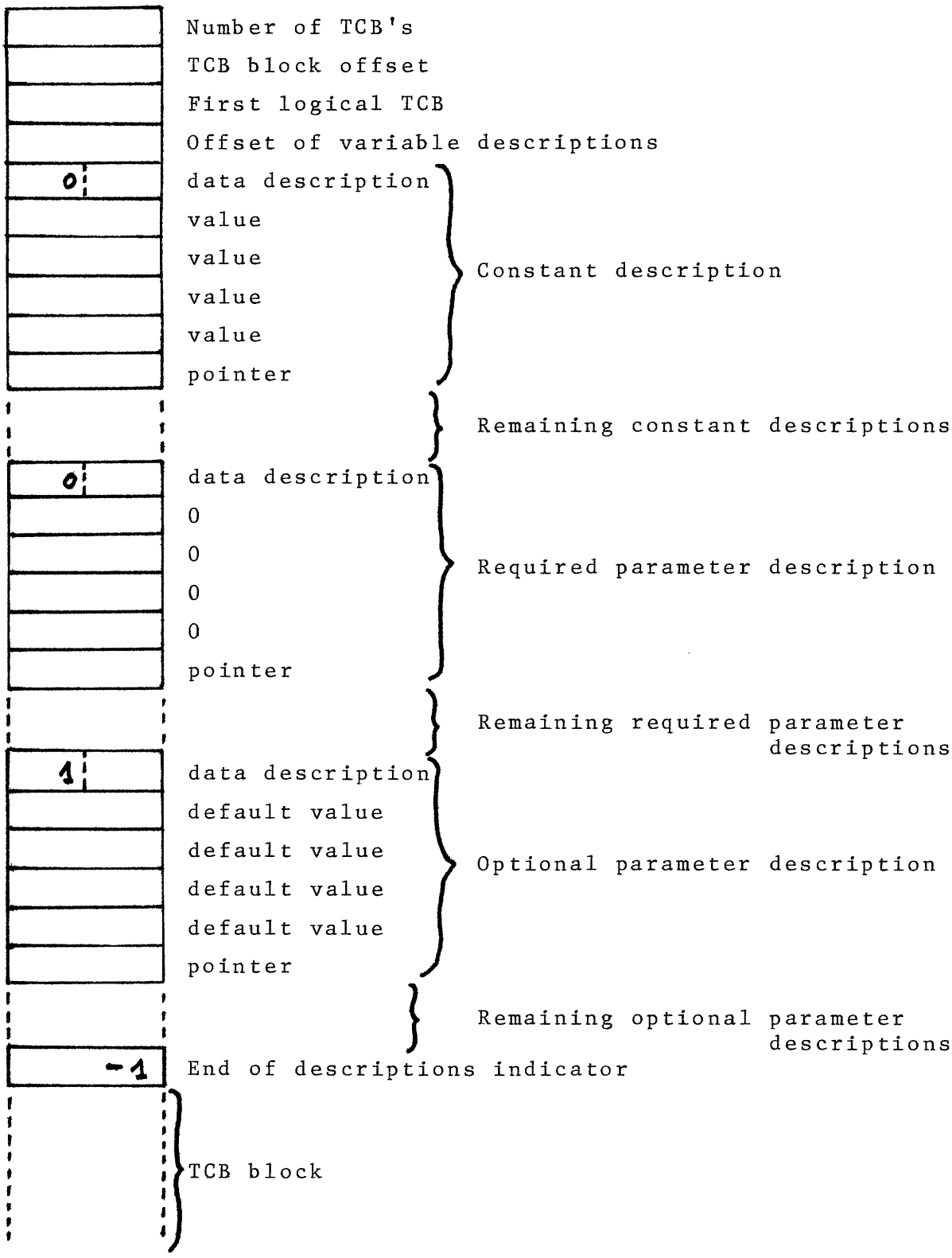

Figure A.1 - NAME.TCB file format

beginning of the TCB block. If the value is to be inserted into several locations, these locations will be chained. Numerical constants are already inserted into the TCB block and their address word in the data descriptions is set to octal minus one. Following the descriptions is an end indicator and the TCB block.

A.9 Task Program Format

Tasks are generally programs performing operations on pictures. The function of a task should be limited to one specific job. If several picture transformations must be performed, it may be advantageous to write several independent tasks. This alternative will offer more flexibility in defining other processes.

A task program should be written in pure code, i.e. no data may be stored in the program. This is desirable in order to allow several different processes to timeshare the same physical copy of the program in core. Each instance of a task will have its own task control block where it can store variables and pointers to files and allocated storage (see section A.6). A pointer to the task's TCB will be provided in register five. Values may also be stored in the general registers, since these will be saved by the supervisor each time the execution of the task is interrupted. A task should also be position independent, or written in such a way that wherever it is loaded, it can be executed with no need to update any address references in the program.[3]

A task should consist of three phases, the initialization phase, the computation phase and the termination phase. The three phases should begin at external entry points named XY.INI, XY.CMP and XY.TRM, where XY is a mnemonic unique to the task.

The task can expect to receive its input data through the T.IN entry in its TCB. Usually, it will be a pointer to a buffer containing the next line of the picture being processed. This buffer may be read, but should never be altered. It is the responsibility of the task to notify the supervisor when it no longer needs this buffer. The task must also request the space necessary for an output buffer from the supervisor. Output is passed to other tasks by putting a pointer to the output buffer into the T.OUT entry in the TCB. Whenever the task cannot continue processing because it needs data or its output buffer is full, it should set the appropriate flags and execute a monitor call. The flags are located in the T.STAT entry in the TCB[2].

References

1. W. F. Schreiber and D. E. Troxel, Digital Wirephoto System, Research Laboratory of Electronics, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1973, unpublished.

2. Charles Lynn, The PDP-11 Picture Processing System, Technical Report APWP #13, Research Laboratory of Electronics, Massachusetts Institute of Technology, Cambridge, Massachusetts, September 13, 1973.

3. Digital Equipment Corporation, BATCH-11/DOS-11 Assembler (MACRO-11), DEC-11-OMACA-B-D, Maynard, Massachusetts, March, 1973, p. 1-6.