1. IDENTIFICATION

1.1 Digital-7-95-U

1.2 Execute Subroutine

1.3 January 25, 1966

2. ABSTRACT

2.1 The Execute Subroutine is designed to allow a user to execute any sequence of instructions anywhere in the program, as if it were a subroutine. In essence, it acts as if the XCT instruction were allowed to execute more than just a single instruction anywhere in memory.

The subroutine includes the instructions necessary for initialization, a pseudo command (XXCT) to allow entrance and exit from the requested subroutines, and enough storage space for pushdown stacks and nesting to five levels.

The subroutines can be used with either a PDP-4 or PDP-7.

3. REQUIREMENTS

3.1 Storage

The subroutines occupy 105_8 locations. They are not designed to work with extended memories.

3.2 Subprograms (None)

3.3 Equipment

Normal 4K or 8K PDP-4 or PDP-7

4. USAGE

4.1 Loading

The user is supplied with a symbolic tape which can be assembled with the main program. The tape contains no undefined tags, no origin, and no final START address.

4.2 Calling Sequence

4.2.1 Before the Execute Subroutine is used for the first time, the user must initialize the system by issuing the command XCTSET. XCTSET is defined as a JMS to a set of instructions which will see that the stack pointers are reset and will replace any locations in memory which may have been changed by any previous use of the subroutines. The latter could occur if the program were stopped and restarted while under the control of the Execute Subroutine. The initialization, therefore, attempts to restore memory to its correct original condition before continuing. Note that if the computer should be stopped at certain locations within the subroutine itself, it might be necessary to reload the entire program to restart.

4.2.2 The format for calling the subroutine is as follows:

```
XXCT    /PSEUDO COMMAND, defined as a JMS to subroutine.
ADDR1   /FIRST LOCATION to be executed
ADDR2   /FIRST LOCATION NOT to be executed.
```

ADDR1 should be the location of the first instruction of the loop to be executed. Fifteen bits are used for the address.

ADDR2 is the location of the first instruction of the loop not to be executed, or, more simply, it is the location which when reached causes a return to the main program at XXCT+3.

Fifteen bits are used for the address. The user must be sure that ADDR2 is one which will definitely be reached sometime during the execution of the subroutine. The contents of the accumulator and link are not destroyed when the subroutine is entered, and will contain the correct results of the executed subroutine upon return.

4.2.3 There are no restrictions on actual instructions being executed as a subroutine, except to remember that an exit will be made the first time the instruction at ADDR2 is reached and before its execution. Execute loops may be nested to a depth of 5. If nesting is desired to any other level, the statement XCTLNG=5 should be changed to indicate the desired amount. Three locations are used for each level of nesting.

4.3 Switch Settings (None)

4.4 Start Up

The subroutines can usually be reinitialized by issuing the XCTSET pseudo command. There may be rare instances when the program should be reloaded in its entirety.

4.5 Errors in Usage

Only one halt can occur within the subroutines:

HALT Location	Meaning	Procedure
XCTTM	Execute loops have been nested to a level greater than 5.	If legitimate, reassemble the program changing the XCTLNG=5 statement to the correct length necessary.

5. RESTRICTIONS

5.1 The subroutines will not operate correctly if the subroutines themselves or the area to be executed is in extended memories or if the computer is in extend mode when the subroutines are called.

The XXCT pseudo command should never be issued by a normal PDP-7 execute (XCT) instruction.

6. DESCRIPTION

6.1 Discussion

The Execute Subroutine allows the use of any set of instructions anywhere in the program, as a subroutine. It is most useful where running time and space are not an important consideration and where, for one reason or another, a set of instructions is not coded as a subroutine. It has been found very useful in the debugging stage to simply assemble the Execute Subroutine with the main program, making corrections where necessary and where possible by executing a proper set of instructions anywhere in the program. After debugging, the proper instructions can be coded as subroutines, if desired, and the Execute Subroutine can be removed if necessary to save time and space. It has also been found useful during testing to execute certain portions of a test program without necessarily executing those portions before or after.

The subroutines operate roughly as follows:

1) Save the accumulator.

2) Check to find out whether the nesting limit has been exceeded. If it has, halt; otherwise continue.

3) Save subroutine starting location (ADDR1).

4) Save the address of the last location (ADDR2) on the last location pushdown stack (XCTLOC).

5) Save the contents of the last location on the contents pushdown stack (XCTCON).

6) Place a JMP instruction in the last location which returns to the Execute Subroutine when the subroutine is terminated.

7) Save the address of the return to the main program on the return address pushdown stack (XCTRET).

8) Index all stack pointers, restore the accumulator, and jump to the subroutine at the first location specified.

When the subroutine has been completed, it will return to the Execute Subroutine where the following procedure is followed:

9) Save the accumulator.

10) Decrement the last location and contents of last location pushdown stack pointers (XCTLOC and XCTCON).

11) Restore the original contents of the last location.

12) Decrement the return address pushdown stack pointer.

13) Restore the accumulator, and return to the main program at the location saved on the return address stack (XCTRET).

The initialization procedure (XCTSET) operates basically as follows:

14) Check to see if the contents pushdown stack pointer (XCTCON) is at its base position.

15) If the stack pointer is at its base position, exit and return to the main program.

16) If the stack pointer is higher than its base position, repeat steps 10 through 12.

17) Repeat steps 14 through 16 until all items have been reset.

6.2 Example

The following is an example of how the Execute Subroutine might be used. The example is not to be taken as one which is either particularly applicable or efficient, but simply a use of the subroutine.

In the PDP-7 Bidirectional DECtape Subroutines (Digital-7-22A-I/O) the following 35 millisecond delay loop is used before changing unit selections:

```
MMWAIT,     MMRS                          /READ DECTAPE STATUS
            AND (400)                     /SAVE CONTROL TYPE
            JMP MMSCH9                    /NO DELAY FOR NEW CONTROLS
            LAC MMCHK1+1                  /PICK UP UNIT NUMBER
            SAD MMSEL                     /COMPARE TO PREVIOUS SELECTION
            JMP MMSCH9                    /NO DELAY IF UNIT IS THE SAME
            DAC MMSEL                     /SAVE NEW UNIT NUMBER
            CLA                           /CLEAR THE ACCUMULATOR
            MMSE                          /SELECT UNIT 0, i.e., DESELECT
            LAM DECIMAL-5000+1OCTAL       /COUNTER
            DAC MMBLF                     /TEMPORARY STORAGE AREA
            ISZ 1.-1                      /DELAY LOOP EQUALS 7 μSEC
            JMP .-1                       /DELAY 35 MSEC
MMSCH9,     LAC MMCHK1+1                  /PICK UP SELECT
            ETC.
```

If the user desired to deselect all DECtapes and delay 35 msec somewhere else in his program the following sequence of instructions could be used:

```
            XXCT
            MMSCH9-6                      /FIRST INSTRUCTION TO EXECUTE
            MMSCH9                        /INSTRUCTION TO CAUSE RETURN
```

Note also that the following set of instructions could be used if the user simply required a 50 msec delay:

```
            LAM DECIMAL-7143+1 OCTAL      /50000 DIVIDED BY 7 μSEC
            XXCT
            MMSCH9-3                      /FIRST INSTRUCTION TO EXECUTE
            MMSCH9                        /INSTRUCTION TO CAUSE RETURN
```

The user should be aware that in these particular examples use of the subroutines themselves causes an additional delay (see Section 9); however, for purposes of illustration it is being ignored.

7. METHODS (See Section 6)

8. FORMAT (Not Applicable)

9. EXECUTION TIME

The subroutines require 77 cycles or approximately 135 μsec on a PDP-7 or 616 μsec on a PDP-4 to enter and exit any requested subroutine.

10. PROGRAM

10.4 Program Listing

```
/EXECUTE SUBROUTINE
/LMH  1-25-66.  WILL NOT WORK WITH EXTENDED MEMORY.
/EXECUTES A SEQUENCE OF INSTRUCTIONS NOT IN SUBROUTINE FORM
/138US PER SUBROUTINE.  105 (OCTAL LOCATIONS)
/FORMAT         XXCT
/               ADDR1       /FIRST INSTRUCTION TO BE EXECUTED
/               ADDR2       /FIRST INSTRUCTION NOT TO BE EXECUTED

XCTLNG=5

XXCT=JMS .
                0
                DAC XCTAC                       /SAVE ACCUMULATOR
                LAW XCTRET              /FOR COMPARISON
                SAD XCTCON              /COMPARE TO CONTENTS STACK POINTER
XCTTM,          HLT                     /TOO MANY RECURSIVE LEVELS
                LAC I XXCT-JMS          /STARTING ADDRESS
                DAC XCT+WA              /TEMPORARY STORAGE AREA
                ISZ XXCT-JMS            /INDEX ARGUMENT POINTER
                LAC I XXCT-JMS          /FIRST NON-EXECUTED LOCATION
                DAC I XCTLOC            /SAVE ON LAST LOCATION STACK
                DAC XCTWA2              /TEMPORARY STORAGE AREA
                LAC I XCT+WA2           /CONTENTS OF LAST LOCATION
                DAC I XCTCON            /SAVE ON CONTENTS STACK
                LAC (JMP XXCTRT)        /JMP INSTRUCTION
                DAC I XCTWA2            /REPLACE CONTENTS OF LAST LOCATION
                ISZ XXCT-JMS            /INDEX ARGUMENT POINTER
                LAC XXCT-JMS            /PICK UP RETURN ADDRESS
                DAC I XCTRET            /SAVE ON RETURN ADDRESS STACK
                ISZ XCTLOC              /INDEX LAST LOCATION STACK POINTER
                ISZ XCTCON              /INDEX CONTENTS STACK POINTER
                ISZ XCTRET              /INDEX RETURN ADDRESS STACK POINTER
XCT1,           LAC XCTAC                       /RESTORE AC, LINK UNCHANGED.
                JMP I XCTWA             /START EFFECTIVE SUBROUTINE

/RETURN FROM SUBROUTINE
XXCTRT,         DAC XCTAC                       /SAVE ACCUMULATOR
                LAM -1                  /FOR SUBTRACTION
                ADD XCTLOC              /DECREMENT LAST ADDRESS STACK POINTER
                DAC XCTLOC              /RESTORE POINTER
                LAM -1                  /FOR SUBTRACTION
                ADD XCTCON              /DECREMENT CONTENTS STACK POINTER
                DAC XCTCON              /RESTORE POINTER
                LAC I XCTCON            /PICK UP ORIGINAL CONTENTS OF LAST LOCATION
                XCT XCTLOC              /RESTORE CONTENTS OF LAST LOCATION
                LAM -1                  /FOR SUBTRACTION
                ADD XCTRET              /DECREMENT RETURN ADDRESS STACK POINTER
                DAC XCTRET              /RESTORE POINTER
XCT2,           LAC XCTAC                       /RESTORE AC, LINK UNCHANGED
                JMP XCTRET              /JMP TO MAIN LOOP.
```

```
/INITIALIZE XCT SUBROUTINES.  WILL RESET LOOPS IF ANY ITEMS REMAIN
/IN TABLE
XCTSET=JMS .
                0                       /WORK AREA, XCTAC
                LAW XCTCON+1            /FOR COMPARISON
                SAD XCTCON              /IS POINTER RESET
                JMP I .-3                       /EXIT
                LAC (JMP XCT3)          /RETURN FROM CLEARING STACK
                DAC XCT2                /REPLACE LAC INSTRUCTION
                JMP XXCTRT+1            /CLEAR ONE ENTRY IN THE STACK
XCT3,           LAC XCT1                /LAC INSTRUCTION
                DAC XCT2                /RESTORE LAC INSTRUCTION
                JMP XCTSET+1-JMS        /CHECK FOR COMPLETION.

XCTAC=XCTSET-JMS
XCTLOC,         DAC I .+1
XCTLOC+XCTLNG+1/
XCTCON,         LAW .+1
XCTCON+XCTLNG+1/
XCTRET,         JMP I .+1
XCTRET+XCTLNG+1/

START
```

11. DIAGRAMS (None)

12. REFERENCES (Not Applicable)