GEORGIA INSTITUTE OF TECHNOLOGY
Engineering Experiment Station

PROJECT INITIATION

Date: July 10, 1970

Project Title: To develop Specifications for FOCAL-11
Project No.: A-1264
Project Director: John C. Alderman
Sponsor: Digital Equipment Corporation - PDP-11
Effective: June 1, 1970
Estimated to run until: August 31, 1970

Type Agreement: Industrial Research
Amount: $2,500

Reports: Monthly Progress
Final

Contact Person: Mr. Andrew C. Knowles
Product Line Manager
PDP-11 Product Line
Digital Equipment Corporation
146 Main Street
Maynard, Massachusetts 01754

Assigned to: Nuclear & Biological Sciences Division

COPIES TO:
- Project Director
- Director
- Associate Director
- Assistant Director(s)
- Division Chiefs
- Branch Head
- General Office Services
- Engineering Design Services
- Photographic Laboratory
- Research Security Officer
- Accounting
- Purchasing
- Report Section
- Library
- Rich Electronic Computer Center
- Other: [ ]
PROJECT TERMINATION

Date 3/31/72

PROJECT TITLE: To Develop Specifications for FOCAL-11
PROJECT NO: A-1264
PROJECT DIRECTOR: Mr. J. C. Alderman
SPONSOR: Digital Equipment Corporation
TERMINATION EFFECTIVE: 3/31/72

CHARGES SHOULD CLEAR ACCOUNTING BY: All charges cleared.

Nuclear & Biological Sciences Division

COPIES TO:
Project Director
Director
Associate Director
Assistant Directors
Division Chief
Branch Head
Accounting
Engineering Design Services

General Office Services
Photographic Laboratory
Purchasing
Report Section
Library
Security
Rich Electronic Computer Center
Progress Report on the Development of FOCAL-11

PROJECT NAME: FOCAL-11 SPECIFICATIONS
PROJECT NUMBER: A1264
AUTHORS: GWEN McALLEN
          JOHN ALDERMAN
Progress Report on the Development of FOCAL-11

I. SPECIFICATIONS

The preliminary specifications for FOCAL-11 have been prepared according to the format set forth in Programming Projects Control Standard, the programming department memo #005-003-004-000. These specifications include the following sections:

1.0 OVERALL DESCRIPTION
2.0 HARDWARE ENVIRONMENT
3.0 SOFTWARE ENVIRONMENT
7.0 LANGUAGE

These sections constitute a basic framework for the language.

II. CODING

Roughly, one-third of the coding has been completed and is ready to be tested. The commands that have been coded are: ON, IF, DO, GOTO, FOR, SET, ASK, TYPE.

In order for these commands to be executed, the following subroutines are necessary: READC, PRINTC, PUSHF, POPF, SORTC, SORTJ, PROCESS, TESTC, TESTN, MOVTORØ, GETC, SPNOR, GETNSC, FINDLN, GETARG, SORTV, TSTLF, TSTGRP, GETLN, ECALL, EVAL, GTOPR, MOVSØØ, GETOP. All of these subroutines have been coded.
The subroutines EVAL, MOV500, GETOPR, and GETOP in the present version of FOCAL-11 are similar to the subroutines of the same names in BASIC-11. The other subroutines are similar to corresponding subroutines in FOCAL-8 (1969). In fact, considerable effort was made so that FOCAL-11 would be consistent with FOCAL-8, a feature that is beneficial to both the experienced and inexperienced users of the FOCAL language. The following conventions have been adopted in order to implement consistency.

Macros Being Utilized | Meaning of the Macros in PAL-11 | Comparable FOCAL Macro  
----------------------|--------------------------------|-------------------------
PUSH DUMMY1 = MOV DUMMY1, -(SP) | PUSHA | 
POP DUMMY1 = MOV (SP)+, DUMMY1 | POPA | 
PUSHJ DUMMY1 = JSR R5, DUMMY1 | PUSHJ | 
POPJ = RTS R5 | POPJ | 
GETC = MOV (RA)+, CHAR | GETC | 
SORTC = JSR R5, XSSORTC | SORTC | 
SORTJ = JSR R5, XSSORTB | SORTJ | 
PRINTC = JSR R5, XORINTC | PRINTC | 
GETLN = JSR R5, XGETLN | GETLN | 
FINDLN = JSR R5, XFIND | FINDLN | 
TESTC = JSR R5, XTESTC | TESTC | 
TESTN = JSR R5, XTESTN | TESTN | 
PUSHF = JSR R5, XPUHF | PUSHF | 
POPF = JSR R5, XPOPF | POPF | 
TSTGRP = JSR R5, TSTGRP | TSTGRP | 
TSTLP = JSR R5, LPRTST | TSTLP | 
SPNOR = JSR R5, XNSC | SPNOR | 

SP = the stack pointer (register 6) and is the register that handles the push-down list.
R5 = register 5 and is used in most of the subroutine calls.
The present version of FOCAL-11 has been designed so that the PAL-11A Assembler will read in and store text. Thus the actual input form of a FOCAL-11 program must conform to the conventions of PAL-11A. In addition only the teletype can be used for input and output. Also, indirect statements are the only type of statements that are allowed in the present coding. For example, suppose one wants to run the following FOCAL-11 program.

```
1.10 TYPE 2,9
2.13 SET X = 5
2.15 QUIT
```

In order to use the PAL-11A to store this program, one would type the following statements on the teletype keyboard to the assembler.

```
TXTBGN = #L1.01
L1.01: .WORD L2.13
       .BYTE 1,01
       .ASCII 'TYPE 2,9'
       .EVEN
L2.13: .WORD L2.15
       .BYTE 2,13
       .ASCII 'SET X = 5'
       .EVEN
L2.15: .WORD 0 ; A zero signifies the end of text.
       .BYTE 2,15
       .ASCII 'QUIT'
       .EVEN
       .BYTE 15,12
```

The syntactical form required to start execution of this program remains to be defined.
FOCAL-11

THE FORMULA CALCULATOR
FOR THE D.E.C. PDP-11

AUTHORS: GUEN MCALENN
          JOHN ALDERMAN

PROJECT NUMBER: A1264
;VARIABLES IN FOCAL-11

R1=Z1
R2=Z2
R3=Z3
R4=Z4
R5=Z5
R6=Z6
R7=Z7
CHAR=R3
SORTCN=R2

;TEXT CHARACTER PLACED IN R3.
;IN SORTCN - SORTCN = X (0 BYTES
;DOWN IN LIST MATCH OCCURRED).
;IN TESTN - IF CHAR IS A NUMBER,
;SORTCN = ITS BINARY EQUIVALENT.
;ADDRESS OF LINE FOUND IN FINDLN.
;ADDRESS OF PREVIOUS LINE TESTED
;IN FINDLN.
;OPERATOR CODE IN IF.P1 COMMANDS.
;OCTAL VALUES ARE: "="=1, "="=2,
;"="=4.
;PROGRAM COUNTER FOR TEXT.
;TEXTPOINTERS - XPC POINTS TO THE
;1ST WORD OF A LINE IN TEXT.
;ADDITIONAL TEXTPOINTER.
;EXTRA REGISTER.
;FLAG FOR DO COMMAND: GROUP=0,
;LINE=100000, ALL=1.
;LINE NUMBER READ IN DO COMMAND.
;VARIABLE POINTER.
;ASK = TYPE SWITCH.
;DEBUG SWITCH - NONZERO FOR
;LITERAL.
;LIST BRANCHER.
;POINTER FOR OUTPUT FORMAT.
;FLOATING-POINT (NORMALIZED)
;1.0 (3 WORDS).
;LOWEST ADDRESS OF 3 CONSECUTIVE WORDS.
;SAME.
;SAME.
;SAME.
;FLYING-PT. (NORMALIZED) 144
;(3 WORDS).
;LOCATION TO STORE VARIABLE NAME.
;LOCATION TO STORE VARIABLE SUBSCRIPT.
;FUNCTION CODE.
;FLAG = 0 (GETC), <>0 (READC).
; Mnemonics in Focal-11

; PUSH DUMMY1
MOV DUMMY1,-(RG)

; POP DUMMY1
MOV (RG),DUMMY1

; PUSHJ DUMMY1
JSR R5,DUMMY1

; POPJ
RTS R5

; GETC
MOVB (R4),CHAR

; PACKC
MOV CHAR,ADDI

; SORTC
JSR R5,XSORTC

; SORTJ
JSR R5,SORTB

; PRINIC
JSR R5,XPRINTC

; GETLN
JSR R5,XGETLN

; FINDLN
JSR R5,XFIND

; GETNSC
JSR R5,XNSC

; TESTC
JSR R5,XTESTC

; TESTN
JSR R5,XTESTN

; BUMP RN
TST (RN)+

; PUSHF
JSR R5,XPUSHF

; POPF
JSR R5,XPPOP

; TSTGRP
JSR R5,TSTGRP

; SKP
BR .+2

; SKP2
BR .+4

; SPNOR
JSR R5,XSPNOR

; TSTLPR
JSR R5,LPRTST

; SORTV
JSR R5,XSORTV

; MOUSTK
JSR PC,MOUSTR

; MOVTOGR
JSR PC,MOVTOGR

; INC ADC
JSR R5,XNCRADC

; GTOPR
JSR PC,GTOPRG

; GETOP
JSR PC,GETOP
RESET=2
INIT=1
WAIT=6
WRITE=12
READ=11
RSLOT=0
PSLOT=1

SIZE OF BUFFER1 (IN BYTES)
FORMAT (ASCII UNFORMATTED)
BYTE-COUNT
RESERVE 1 BYTE OF STORAGE.

SIZE OF BUFFER2 (IN BYTES)
FORMAT (ASCII UNFORMATTED)
BYTE-COUNT
RESERVE 1 BYTE OF STORAGE.

RESET DEVICES TO INITIAL STATE.

ASSOCIATE DEVICES WITH SLOTS (RDEVSLOT).

WDEV 9 PSL07.

SETUP PERMANENT POINTER TO BEGINNING OF TEXT AND A CHANGE-

ADJUST POINTERS TO NEXT LINE.

TEXT IS STORED SUCH THAT THE FIRST WORD OF A LINE IS
A POINTER TO THE NEXT LINE AND THE SECOND WORD IN
THE LINE CONTAINS THE LINE NUMBER. THE REST OF THE LINE
CONTAINS TEXT.

READ - XREADC USES R3.
XREADC READS IN A CHARACTER FROM THE KEYBOARD AND
PLACES IT IN R3 (CHAR).

READC: IOT
WORD XREADC
BYTE WAIT,RSLOT
IOT
WORD BUF1
BYTE READ,RSLOT
MOVB BUF1+6,CHAR
RTS R5

IS TELETYPewriter KEYBOARD READY?

GO - GOTO XREADC.
YES - READ IN CHARACTER FROM KEYBOARD AND PLACE IN BUFFER1.
MOVE CHARACTER TO CHAR.
;PRINTC - PRINTC ON THE TELETYPETR PRINTER
;THE CHARACTER IN CHAR (R3).

;TC:
PUSH RO
PUSH RI
PUSH R2

;IS TELETYPE PRINTER READY?
.PUSH Ri
.PUSH R2
WORD LOOP
BYTE WAIT.PSLOT
SORTJ
WORD TLISTB
WORD SPRINT
PUSHJ XOUTC

;YES - SORT CHARACTER AGAINST
;TERMINATOR LIST (CR, LF).

;NO MATCH - PRINT THE CHARACTER.
POP R2
POP RI
POP RO
POPJ

;i4

;NO: GOTO LOOP.

;MOVE CHAR INTO BUFFER2 AND
;PRINT THE CHARACTER.

;TC:
MOV CHAR,BUF2+6
IOT
WORD BUF2
BYTE WRITE.PSLOT
PUSHJ XOUTC
POPJ

;XOUTL - PRINTS ONLY THE CHARACTER IN CHAR WITHOUT
;SORTING THE CHARACTER AGAINST A LIST FIRST.

;TL:
IOT
WORD XOUTL
BYTE WAIT.PSLOT
PUSHJ XOUTC
POPJ

;CRPRINT - CARRIAGE RETURN, LINE FEED PRINT.

;PRNT:
IOT
WORD BUFCR
WORD WRITE.PSLOT
POPJ

;CKPRINT - CONTROL/K CAUSES THE FOLLOWING CHARACTERS
;TO BE PRINTED: K, CR, LF, AND 6 SPACES.

;PRNT:
IOT
WORD BUFCK
WORD WRITE.PSLOT
POPJ

;FOR:
2
2
BYTE 15,12

;SIZE OF BUFFER IN BYTES.

;FORMAT = ASCII UNFORMATTED.
;BYTE-COUNT.
;CR, LF.

;CK:
12
2
BYTE 136,113,15,12

;SIZE OF BUFFER IN BYTES.

;FORMAT = ASCII UNFORMATTED.
;BYTE-COUNT.

;K, CR, LF.

;BYTE 40,40,40,40,40,40

;6 SPACES.
;PUSHF - XPUHF, pushes a floating-point number (whose
;address is in register R5) onto the stack. XPUHF
;uses registers R0, R5. The 3rd word of the floating-
;pt. number is on the top of the stack.

MOV R0, XRO
MOV (R5)->, R0
PUSH (R0)+
PUSH (R0)+
PUSH (R0)
MOV XRO, R0
RTS R5

;SAVE DATA IN RO.

;PUSH THE 1ST WORD FIRST.
;PUSH THE 2ND WORD NEXT.
;PUSH THE 3RD WORD LAST.
;RESTORE DATA TO R0.

;POPF - XPOPF, pops a floating-point number from the
;stack to the location given in register R5. XPOPF
;uses registers R0, R5. Note that the 3rd word of the
;floating-pt. number is on the top of the stack. Therefore
;the 3rd word comes off the stack first.

MOV R0, XRO
MOV (R5)->, R0
ADD #6, R0
POP -(R0)
POP -(R0)
PUSH -(R0)
MOV XRO, R0
RTS R5

;SAVE DATA IN RO.

;GET ADDRESS OF 3RD WORD + 2.
;POP 3RD WORD TO 3RD WORD ADDRESS.
;POP 2ND WORD NEXT.
;POP 1ST WORD LAST.
;RESTORE DATA TO R0.
SWILL

"t:OMPUTF:a

tr - PLVJAITJ 11; LAD C ' C -

5- 0.07;74'.

;SORTC 	 XSORTC 9

UPON ENTRY A CHARACTER IS IN R3 (CHAR)
AND THE POINTER TO A LIST ADDRESS IS IN R5. XSORTC COM-
PARES THE CHARACTER IN R3 TO THOSE IN A LIST. IF A MATCH
OCCURS, THEN R2 (SORTCN) IS SET = 2 TIMES THE NUMBER OF
BYTES DOWN THE LIST THAT THE MATCH OCCURRED. IF A MATCH
OCCURS IN SORTC, THE 1ST RETURN IS TAKEN. IF NO
MATCH OCCURS, THEN THE 2ND RETURN IS TAKEN.

THE CALLING SEQUENCE FOR XSORTC IS

;SORTC

;WORD LISTADDRESS

;Xxx

;YYy

;SORTC USES REGISTERS R0, R1, R2, R5. SORTCN=R2, CHAR=R3.

;SORTC:

MOV (R5)+, R2

MOVR2,R0

INC R0

;TEST:

MOVR2, R2

BMI SEXC

CMPB CHAR, R1

BNE SRTST

SUB R0, R2

SORTCN=R2

ADD SORTCN, SORTCN

RTS R5

;SORTJ - SORTB. UPON ENTRY: A CHARACTER IS IN R3 (CHAR),
AND THE POINTER TO A LIST IS IN R5, AND A POINTER TO A 2ND
LIST IS IN (R5)+2. SORTB COMPARES THE CHARACTER IN R3
OCCURDS, THOSE IN THE 1ST LIST. IF A
LOCATION, SORTCN + ADDRESS OF LIST2, CONTAINS AN ADDRESS
WHICH IS BRANCHED TO. IF NO MATCH OCCURS, THEN A RETURN
IS EXECUTED TO THE 3RD LINE AFTER THE CALL TO SORTB.

THE CALLING SEQUENCE FOR SORTJ IS

;SORTJ

;WORD LISTADDRESS

;WORD LIST2ADDRESS

;SORTJ CALLS SORTC AND USES REGISTER R5 PLUS THOSE USED
IN SORTC.

;SORTJ:

MOV (R5)+, R2

PUSHJ XSORTC+2

BR SRTAD

ENDP R5

RTS R5

;BAD:

MOVR5,R5

ADDSORTCN, R5

MOVR5, R5

RTS R5

;SORT CHAR AGAINST LIST.

;RETURN HERE IF MATCH FOUND.

;DO MATCH - RETURN TO 3RD

;LINE AFTER CALL.

;ADDRESS OF COMMAND SET UP.

;GO DO COMMAND.
GOTO COMMAND, UPON ENTRY R4 POINTS TO A LINE NUMBER.
GOTO PLACES THIS NUMBER IN THE LOCATION "LINENO" AND PUTS
THE ADDRESS OF THIS LINE IN XPC.
GOTO CALLS GETLN AND FINDLN.

GETLN
FINDLN
ERROR

MOV THISLN, XPC
MOV XPC, R4

$LINE NOT FOUND.
$PUT ADDRESS OF LINE IN PC.

;PROCA - UPON ENTRY XPC AND R4 POINT TO THE FIRST
;WORD OF A LINE IN THE STORED TEXT. UPON ENTRY TO PROCESS
;R4, THE CHARACTER POINTER, NOW POINTS TO THE 3RD WORD
;OF THE LINE. PROCA CONSIDERS ONLY THE 1ST LETTER OF A
;COMMAND AND PROCEEDS TO THE ADDRESS OF THE APPROPRIATE
;COMMAND (IF THE COMMAND IS LEGAL). UPON EXIT R5 CONTAINS
;THE COMMAND TERMINATOR AND R4 POINTS TO THE NEXT CHARACTER.
;PROCA CALLS GETC AND SORTC AND USES REGISTERS R3, R4.
;R5 PLUS THOSE IN SORTC.

;SETUP CHARACTER POINTER.
;GET CHARACTER.
;IS CHAR A C.T.?

ADD #4, R4
GETC
CMP #15, CHAR
BNE +4
POPC $ I
SORTC $ I
\WORD GLIST
BR PROCESS
BIC #40, CHAR
PUSH CHAR
GETC
SORTC $ I
\WORD GLIST
BR +4
BR PRC1 $ I
PUSH CHAR $ I
MOV 2(RG), CHAR
SORTC $ I
\WORD COMLIST
BR +4
ERROR $ I
POPC CHAR $ I
MOV $COME0 (RG)
ADD SORTCH, (RG)
JMP 0(RG)+

;SKIP NEXT LINE IF C.T. FOUND.
;GET 1ST LETTER OF COMMAND.
;GET ADDRESS OF COMMAND.
;ADDRESS OF COMMAND LIST.

;ILLEGAL COMMAND.

;GO DO COMMAND.
ON IF COMMAND. UPON ENTRY R4 POINTS TO THE
CHARACTER FOLLOWING THE COMMAND TERMINATOR SORTCH
(R2) IS SUCH THAT THE LOCATION SORTCH + #OPCODE, CONTAINS
THE ADDRESS OF EITHER THE IF OR ON COMMAND.
THE OPCDE REGISTER (RO) WILL
CONTAIN INFORMATION ABOUT THE OPERATOR(S) BETWEEN
THE FIRST AND SECOND EXPRESSION IN THE 2-BRANCH IF OR
ON COMMANDS: "<" = 1, "=" = 2, ">" = 4. R2 IS USED AS
A FLAG. REGISTERS USED - ALL. ON, IF CALL TESTC, GETC
GETMSC, SORTC, SORTJ, ECALL.

MOV #OPCODE, COMADD
ADD SORTCH, COMADD
OPCDE = RO
CLR OPCDE
CLR R2
PUSH R2
PUSH OPCDE
TESTC
JSR PC, EVALC-2
BVS OPTK
BR BOPT
GETMSC
SORTC
- WORD OP LIST
SEH
BCE OPT2
TST 2 (R2)
BEQ IFBR
BR SETBR
MOV #FLAG, RO
MOV OPC
BR AOPT1
MOV = ENT + 7
MOV #FLAG, RI
MOV #FLAG, RO
MOV #FAM
GETMSC
SORTJ
- WORD PROPS
- WORD APROP
ERROR
BR: 
JMP = ENT + 10
MOV #FLAG, RI
MOV #FLAG, RO
CAPR,FAM
BLT R011
BEQ R012
ROR OPCDE
ROR OPCDE
ROR OPCDE
ROR OPCDE
BCS Y2BR
MOV #1, RO
BR IF3

; DISTINGUISH IF AND ON.

; CLEAR OPERATOR CODE.

; SAVE FLAG.

; TEST CHARACTER.

; T- EVALUATES EXPRESSION.

; F- OVERFLOW TEST FOR PAREN MATCH.

; A- GET A NONSPACE CHARACTER.

; SORT CHAR AGAINST OPERATOR LIST.

; MATCH - GOTO OPT2.

; IF MATCH - TEST BRANCHING FLAG.

; IF FLAG = 0, GOTO IF3BR.

; OTHERWISE DO A 2-BRANCH IF, ON.

; MOVE NUMBER EVALUATED BY T E V A L TO FLAG.

; MOVE NUMBER TO FLARG.

; GET NON-SPACE CHARACTER.

; SORT CHAR AGAINST PAREN AND OPERATOR LIST.

; RETURN TO OPT1 IF OPRTR AND TO AOPT IF L-PAREN. (R2<=0)

; RELATIONSHIP BETWEEN EXPRI

; AND EXPR2 COMPARED WITH OPCDE.

; GOTO IF2BR IF OPCDE AGREES.

; NO AGREEMENT - MOVE TO TERMINATOR AND DO NEXT COMMAND.
$3-BRANCH IF.
$TEST VALUE OF EXPRI.
$IF > $0, GO PAST 2 COMAS.
$IF = $0, GO PAST 1 COMA.
$EXPRI-$0 = DO NOT GO PAST
$ANY COMMAS.

$READ THE LINE NUMBER.
$MOVE TO TERMINATOR.

$IFADD=POINTER TO IF COMMAND.
$WAS THE COMMAND AN IF OR AN ON?
$ON COMMAND = DO.
$IF COMMAND = GOTO.

$LOOK FOR COMMA OR C.T.

$NO MATCH.

$FOUND A COMMA.

$OR (ENCOUNTERED, SET FLAG.

$< ENCOUNTERED, SET OPCDE.

$> ENCOUNTERED, SET OPCDE.
Mum: CrtErmarrEn

DO USES REGISTERS

PUSH R4
GETLN, -Ica -
USHF, UCRD TEXTP
PUSN7, UORD CAGSti
TST NAGSU BNI DOONE
FINDLV BR
NOV TNISLN9R2
toorq,
P e 
cab 9
BU1V
NOV R2pRI
TSTORP ERROR
RPI: PUSNJ PRO3A
POPP
1
RD NAGSU
OU
OMPC 9 R2
BEO DOODT
NOV R2 v XPO
EN? R2
75?
unGsu
BGT ;->
NOV R2 o R1 ;
TSTGRP . ;
BR
Dconi ;
NOV (R2) 9 1./NENO
BR DGRPI
ONE .1.
FINDLO
ERROR
PUSNJ PROCA
OP F. UORD PAGSU
OVT: POPF ,UORD TEXTP
PO? R4 PROC
AND RO, ;SAVE TEXTPOINTERS,
CiET THE LIVE NIEZER.
;PUSH UP, OSU,
;IS NAGSU NEGATIVE? ;YES - CO TO DOONE.
;NO - FIND L/CE UHOSE 0
;IS 	 TO OR 1 , LINEN00
;GET ADDRESS OF THE LINE
;IS LINE 0
IV THE GROUP?
4:
EXECUTE LINED
;RESTORE NAGSU
;CNECiZ FOR END OF ?Me
CO TO MONT.
;Si NAGSU FOR DO ALL OR GROUP,
;DO ALL - SHIP NEXT 3
uEs.
;DO GROUP.
;IS NEXT LINE I GROUP?
;NO - COTO DCONT,
;YES,
;FIND THE LINE.
;LINE IS NOT FOUND.
;FIND THE LINE.
;DO THE LINE. DO THE DATA.
;$RESTORE THE DATA.
;CHECK FOR END OF IEXI.
;DO ALL GROUP.
;EXECUTE IEXI.
;YES - EXECUTE LINE.
;YES - FIND LINE WHOSE 0
;NO - GO TO DONT.
;GET ADDRESS OF THE LINE 0.
;IS LINE 0 IN THE GROUP?
}
TESTC - TESTC - TELLS WHETHER A NON-SPACE CHARACTER (EN)
R3) IS A TERMINATOR (1ST RETURN), A NUMBER (2ND RETURN),
A FUNCTION (3RD RETURN), OR AN ALPHABETIC (4TH RETURN).
TESTC CALLS SPNOR, SORTC, TESTD AND USES REGISTERS R3 AND
R5 PLUS THOSE IN THE CALLED SUBROUTINES. UPON ENTRY A
CHARACTER IS IN R3 (CHAR). IF IT IS A SPACE, THEN THE NEXT
NON-SPACE CHARACTER FROM TEXT IS PLACED IN R3 AND
IS TESTED.

ESTC CALLS - SPNOR 9 SORTC 9 TESTD AND USES REGISTERS R3 AND

R5 PLUS THOSE IN THE CALLED SUBROUTINES. UPON ENTRY A
CHARACTER IS IN R3 (CHAR). IF IT IS A SPACE, THEN THE NEXT
NON-SPACE CHARACTER FROM TEXT IS PLACED IN R3 AND
IS TESTED.

ESTC: SPNOR
SORTC
.WORD TERMS
RTS R5
BUMP R5
CMP 0 #F, CHAR
BEQ X:5
TESTD
RTS R5
BR .+4
RTS R5 ; 1
BUMP R5 ; 2
BUMP R5 ;<
RTS R5

FOR SET COMMAND. UPON ENTRY R4 POINTS TO THE CHARACTER
FOLLOWING THE COMMAND TERMINATOR. FOR SET USES ALL
OF THE REGISTERS AND CALLS GETARG, SPNOR, EVAL, SORTJ,
PUSHF, POPF, PROCESS.

FOR: SET: PUSHJ GETARG
SPNOR
CMP CHAR, #"
BEQ .+4
ERROR
PUSH PT1
JSR PC, EVAL00
POP PT1
MOV PT1, R0
MOV OR0
GETC
SORTJ
.WORD TLIST
.WORD FLIST1
.ERROR

INCR: PUSH PT1
JSR PC, EVAL00
MOV OFLAC, R0
MOV OR0
GETC
SORTJ
.WORD TLIST
.WORD FLIST2
.ERROR
`EVALUATE LIMIT = NO ERROR
DETECTION AFTER LIMIT.

;SAVE TEXT OF OBJECT STMTS.

;DO THE OBJECT STMTS.

;RESTORE THE REMAINING TEXT.

;ADD THE INCREMENT TO THE VARIABLE.

;TEST LIMIT AGAINST VARIABLE.
LIMIT > OR = VAR, GOTO FCONT.
LIMIT < VAR.

;SET INCREMENT = 1.
;FLTONE=NORMALLIZED, FLTING
;POINT 1.
ASK, TYPE COMMAND. UPON ENTRY R4 POINTS TO THE CHARACTER
AFTER THE COMMAND TERMINATOR, ENTER TYPE WITH R9 <
CR = 0, ASK, TYPE USES REGISTERS R0, R3, R5.
ASK, TYPE CALLS SORTF, GETARG, PRINTC, FLINPT, EVAL, GETC, SORTJ,
GETLN, XOUTL.

CLR R0
MOV R0, ATSU
CLR DEBSU
SPNOR
SORTJ
*WORD ATLIST
*WORD ATLIST
TST ATSU
BEQ TYPE2
PUSHJ GETARG
PUSH CHAR
MOV 0, R3
PRINIC
INC LSUB3
MOV 0, R3
JSR R5, FLINPT
POP CHAR
BR ASK
PE2: JSR FC, EVALCO
JSR R5, FOUTPUT
BR TYPE

INC DEBSU
BEQ +4
GETC
SORTJ
*WORD TLIST2
*WORD TLIST3
PRINIC
BR TQUOT+2

GETC
GETLN
MOV LINFO, FISU
BR TASK

MOV CCR, R3
JSR R5, XOUTL
DEC R3
BR *X+6

MOV CCR, R3
PRINTC

GETC
BR TASK

DISTINGUISH ASK AND TYPE
COMMANDS.
RE-ENABLE THE TRACE.
IGNORE SPACES.
SPECIAL CHARACTER?

TEST QUOTE SWITCH.

DO ASK - SETUP PTI.
SAVE IN-LINE CHARACTER.

TYPE COLON.
INDICATE "READC".

READ DATA AND SAVE.
RETEST LAST TERMINATOR.

DO TYPE.
PRINT.

DISABLE TRACE.

TYPE LITERALS.

PASS PERCENT SIGN.
READ FORMAT CONTROL: "27.63".
FISU = OUTPUT INFORMATION.

SPLAT = C.R. ALONE.

NON-PRINTING DELAY FOR CR.

EXCLAMATION POINT = CR, LF.

MOVE TO NEXT CHARACTER.
\textbf{SMALL COMPUTER APPLICATIONS LAB}

\textbf{GETC (GET NON-SPACE CHARACTERS) AND GETLN (GET LINES) BOTH CALL GETC AND BOTH USE REGISTERS R0-R5.}

\begin{verbatim}
: OR: CHP #40,CHAR
BNE +6 ; IS CHAR A SPACE?
GETC ; I YES, GET NEXT CHARACTER.
BR ASKMR ; I TEST IT.
RTS R5 ; I NO.

; TESTN - XTESTN TESTS WHETHER THE CHARACTER IN R3 IS
; A PERIOD (1ST RETURN), A LETTER (2ND RETURN), OR A
; NUMBER (3RD RETURN). TESTN USES REGISTERS R0, R2, R3 (CHAR),
; AND R5.

: EN: CMP 040,CHAR
BEQ +4 ; IS CHAR A PERIOD?
BUMP R5 ; I NO.
MOV CHAR, SORCN ; YES - 1ST RETURN.
SUB #60, SORCN ; IS CHAR A LETTER, A @, ETC.?
BGE +4 ; I NO.
RTS R5 ; I YES.
MOV #11, R0 ; IS CHAR A #?
ADD SORCN, R0 ; YES - 3RD RETURN.
BGT +4 ; I NO - 2ND RETURN.
RTS R5

; FINDLN - XFIND FINDS A PARTICULAR LINE NUMBER IN THE
; TEXT. UPON ENTRY THE DESIRED LINE NUMBER RESIDES IN
; "LINENO". FINDLN USES REGISTERS R0, R1, R2, R4, R5. R1
; USED AS LASTLN, R0 USED AS THISLN. AXOUT = R4. UPON
; EXIT R0, THISLN, XPC, AND R4 CONTAIN THE ADDRESS OF THE
; LINE WHOSE LINE 0 IS > OR = TO THE DESIRED LINE NUMBER.
; R1 AND LASTLN CONTAIN THE ADDRESS OF THE LAST LINE
; TESTED.

: EN: MOV CFRS, R1
MOV R1, R0
MOV LINENO, R2
CMP R2, #2(R0)
BGE FEND3-4 ; IS LINENO > OR = LINE0 IN TEXT?
MOV R0, R1 ; YES - SKIP NEXT 4 LINES.
MOV R0, R0 ; I END OF TEXT?
BNE FINDN ; I NO - GOTO FINDN.
BR FEND3 ; I YES.
BNE FEND3
BUMP R5 ; MOVED PAST LINENO-1ST RETURN.
FINDN ; FOUND LINENO - 2ND RETURN.
BUMP R5 ; SET UP LASTLN.
MOV R0, THISLN ; SET UP THISLN.
MOV R0, AXOUT
MOV R0, AXOUT
AXOUT = R4
CLR XCT
RTS R5
\end{verbatim}
GETARG - UPON ENTRY R3 SHOULD CONTAIN AN ALPHABETIC CHARACTER. IF NOT, AN ERROR DIAGNOSTIC IS GIVEN. IF THE VARIABLE BEING SOUGHT IS NOT FOUND, THEN IT IS CREATED AND ASSIGNED A VALUE OF ZERO. UPON EXIT THE ADDRESS OF THE FOUND VARIABLE (OR CREATED VARIABLE) IS STORED IN PT!. GETARG USES ALL OF THE REGISTERS. GETARG CALLS TESTC, PACKC, GETC, SORTC, TSTLPR, ECALL, SORTV.

ARG TESTC ERROR ERROR ERROR
VAR: CLR ADD1 PACKC GETC SORTC
..WORD TERMS
BR GSERCH
MOV CHAR,ADD1+1
GETC
SORTC
..WORD TERMS
BR GSERCH
MOV CHAR,ADD1+1
GETC
SORTC
..WORD TERMS
BR GSERCH
BR ...

rch: TSTLPR ER GSI MOV ADD1,R1 JSR PC,ECALL BR GSERCH MOV ADD1,R1 JSR PC,ECALL MOV ADDB,C
FIX CMP (RG),R1 ADD OS,R6 JSR RG INTEGER MOV R1,SUBS PUSH R4 MOV ADD1,CHAR SORTV
TH1: .BYTE 0,2 ..WORD SPLIST1 JMP SPECT1 SORTV SORTV
TH2: .BYTE 0,2 ..WORD SPLIST2 JMP SPECT2 SORTV
TH3: .BYTE 0,12
RLST: ..WORD 0 SKP2 JMP CREATE
JMP ADDRESS
PUSH ADDRESS
MOV SUBS,CHAR

..TEST CHARACTER.
..T.
..N.
..F.
..ALPHABETIC CHARACTER-PACK IT IN 1ST BYTE OF ADD1.
..GET A CHARACTER.
..IS CHARACTER A TERMINATOR?
..YES.
..NO - SAVE 2ND LETTER IN 2ND BYTE OF ADD1.
..IGNORE THE REST.

..LOOK FOR SUBSCRIPT VIA SORTC.
..NOT SUBSCRIPTED BY L-PAR.

..TEST PAREN MATCH.
..RESTORE NAME.
..PUT THE NUMBER ON THE STACK.
..CONVERT TO 16-BIT NUMBER AND PLACE THE NUMBER IN R1.
..DELETE NUMBER ON STACK.
..CONVERT TO 16-BIT NUMBER.
..STORE SUBSCRIPT.

..VARIABLE NAME IN CHAR.
..SORT NAME AGAINST SPECIAL LIST1.

..ADDRESS OF SPECIAL LIST1.
..MATCH.
..NO MATCH-SORT AGAINST SPECIAL LIST2.
..ADDRESS OF SPECIAL LIST2.
..MATCH.
..NO MATCH - SORT NAME AGAINST VARIABLE LIST.

..MATCH - SKIP NEXT LINE.
..NO MATCH.
..GET VARIABLE SUBSCRIPT ADDRESS.
..SAVE THIS ADDRESS.
..PLACE SUBSCRIPT IN CHAR.
JSR R5, STV1
JMP CREATE ; I
CNP (R6)+, ADDRSS;-
BEQ VFOUND
SUB $2, ADDRSS
PUSH R2
MOV ADD1, CHAR
JSR R5, STV1
J:1? CREATE ; (RG)-: -
BCE VARSRN
BUMP ADDRSS
1.10t1 DU::? ADDRSS
ADDRSS
PT1
PO? R4
POPJ

SORTV - XSORTV INSTIGATES A SEARCH ON A LIST. UPON
ENTRY THE ENTITY BEING SEARCHED FOR IS IN CHAR. THE
CALLING SEQUENCE FOR SORTV IS
SORTV
; BYTE LENGTH, VARBYTES
; NO. OF BYTES IN LIST
; AND THE NO. OF BYTES/VARIABLE.
; XXX
; RETURN HERE IF MATCH OCCURS.
; YYY
; RETURN HERE IF NO MATCH OCCURS.
; UPON EXIT IF THE ENTITY WAS FOUND, R2 CONTAINS ITS
; ADDRESS.

LSLNG=R1
VARBYT=R0
ADDRSS=R2

LSLNG=BYTE-LENGTH OF LIST.
VARBYT=BYTES/VARIABLE
ADDRSS=ADDRESS IN LIST.

MOVB (R5)+, LSLNG
MOVB (R5)+, VARBYT
MOVB (R5)+, ADDRSS
ADD ADDRSS, LSLNG
CNP CHAR, (ADDRSS)
BEQ STV1
ADD VARBYT, ADDRSS
CNP ADDRSS, LSLNG
BLT STV1
BUMP R5

BEGINNING ADDRESS OF LIST.
LSLNG=LAST LIST ADDRESS+1
DOES CHAR=VARIABLE IN LIST?
YES = 1ST RETURN.
NO=GET ADDRESS OF NEXT VAR.
AT END OF LIST?
NO = GOTO STV1.
YES = 2ND RETURN.

EXIT: POPJ
I. If a left parenthesis was encountered and then a match in the terminator list occurred between SORCNO = 5 and 11, TSTLPRT tests to see if the value of SORCNO is in this range.

EXIT:

CMP #11, SORCNO

GRT LPRXIT

CMP #5, SORCNO

BLE LPRXIT

BUMP R5

EXIT:

TSTGRP tests to see that the line is in the group specified by LINENO. Address of a line is in R1. Compare group J TO THE GROUP NUMBER OF LINENO.

GRP: MOV (R1), R1

SUB R1

CMPB R1, ZLINENO

BNE -4

BUMP R5 ; I

POPJ

IF GROUP NUMBERS AGREE?

DO GROUP NUMBERS AGREE?

; YES - 2ND RETURN.

; NO - 1ST RETURN.
XGETLN FORMULATES THE LINE NUMBER (E.G., 2 IN THE DO AND GOTO 22:1) AND PLACES THIS NUMBER IN LINENO. THE LOW-
ORDER BYTE OF LINENO IS THE STEP NUMBER AND THE HIGH-
ORDER BYTE IS THE GROUP NUMBER. XGETLN GIVES AN ERROR IF
THE LINE NUMBER IS NOT IN THE RANGE 1.01 TO 99.99.
(EXCEPTION: DO ALL. THERE LINENO=0 AND NO ERROR DIAG-
OSTIC IS GIVEN). XGETLN SETS NASU: 000000 FOR A GROUP 0,
100000 FOR A LINE 0, AND 000001 FOR ALL.
XGETLN USES REGISTERS R0, R1, R2, R3.
XGETLN CALLS SPHOR, EVAL, FLOATING POINT PACKAGE (FIX).
FLY

S

U

B

F

M
Floating-point data stored in G/LAC locations. If the command is a floating-point number, it is treated as an argument.

Upon entry, R4 points to the next character, and R3 contains a terminator character. Improperly formed arguments terminate the program.

Upon exit, R4 points to the next character, and R3 contains the character.
GETC
SORTC
WORD TLIST
BR GEXITA
ERROR ; I
GETC
SORTC
WORD TLIST
BR GEXITA
BR GEXIT

; TEST FOR TERMINATOR.
; INCORRECT TERMINATOR.
; MOVE TO TERMINATOR IN THE
; DO ALL COMMAND.

TSTB LINENO:<=
BEQ +4 ;->
MOV #100000,NAGSU: I
TSTB LINENO:<=
BNE +4 ;->
MOV #1,NAGSU ; I
POPJ

CMP #143,FLACI
BLE +4 ;->
ERROR ; I
TST FLACI ; <-
BGT +4 ;->
ERROR ; I
MOVC FLACI,LINENO:<-
POPJ

; SETUP NAGSU.
; LINE: NAGSU=100000.
; ALL: NAGSU=000001
; GROUP: NAGSU=000000.

; NUMBER IS OUTSIDE RANGE.
; NUMBER IS OUTSIDE RANGE.
1. **Move Subroutine Return Down**

2. **The PDLISI Past Saved Contents**

3. **Of R1, Push Return Address On**

4. **Stack, Back Up TextPointer.**

---

**Evaluate an Arithmetic Expression:**

Upon entry R4 points to current text position ("("). On exit R1, R2, and R5 contain the numeric value of the expression used — all.

---

Clear Paren Count.

Save Contents of R5.

Push NULL ("-1") on stack.

Get a NonSpace Character.

Is it a "-"?

Yes — Ignore it. Goto EVAL03.

Is Character A "("?

No - Goto EVAL01.

Yes - Set OPERAND2 = 0.

---

Get a Character.

Is it an Open Paren?

No - Goto EVAL04.

Push a Zero On the Stack.

Increment Paren Count.

Go Back To EVAL02.

Move Charac Pointer Back One.

Store Paren Count

And Other Data.

---

Get an Operand.

Restore Data.

---

Get an Operator'(In R0).

Is The Stack Null?

No - Goto EVAL06.

Push Operand On The Stack.

Push Operator On Stack.

Go Back To EVAL03.

Save the textPointer.

Get the Table Address.

Find Operator2.

It Must Be Found.

Get Rid Of Byte Pointer.

Store In R5.

Get Table Address.

Find Operator1.

It Must Be Found.

Clear Low-Order Bit.

Save it For Now.

Restore TextPointer.

Is Priority Of Operator1 > Operator2?

No - Go Back To EVAL05.

Yes - Save Operator2.

Get Table Address.
FIND OPERATOR.
$IT MUST BE FOUND.
GET DISPLACEMENT.
GET ROUTINE ADDRESS.
SAVE IT.
DISCARD OLD OPERATOR.
GET DESTINATION ADDRESS.
PUT SOURCE ON STACK.
SAVE TEXTPOINTER.
GET SOURCE ADDRESS AND ROUTINE ADDRESS.
SAVE OPERATORS.
GO COMPUTE VALUE.
RESTORE OPERATOR.
RESTORE TEXTPOINTER.
DISCARD SOURCE.
PLACE RESULT.
IN
OPERAND2.
IS STACK NULL?
NO-TAKE CARE OF REST OF STACK.
IS OPERATOR A CLOSED PARENT?
YES - GOTO EVAL14.
NO. IS IT A NULL?
NO - GOTO EVAL05.
IS THE PARENT COUNT ZERO?
NO - GOTO EVAL13.
YES, POP NULL.
RESTORE CONTENTS OF R5.
CLEAR CONDITION CODES.
RETURN WITH RESULT IN R1, R2, R3.
PAREN COUNT BAD.
IS PAREN COUNT ZERO?
NO - GOTO EVAL15.
POP NULL.
RESTORE CONTENTS OF R5.
YES - ERROR MESSAGE LATER.
JUMP
IF STACK = -1.
POP NULL OFF STACK.
DECREASE PAREN COUNT BY ONE.
DO NOT
CHANGE
THE ORDER OF
THIS TABLE
THIS TABLE GOES WITH THE ONE ABOVE -
SO DO NOT CHANGE ORDER.
GETOPR - GTPROG, GET AN OPERATOR, LOOK FOR "", "", "", 
AND IF FOUND R0 RETURNS OPERATOR AND R1 POINTS 
TO NEXT CHARACTER, IF NOT FOUND, R0 RETURNS ZERO AND 
R1 POINTS TO CHARACTER WHERE FAILURE OCCURRED. 

REGISTERS USED - R0, R3, R4, 

PUSH R3
GETSCR
MOV #EVAL7+7, R0

CLF GTPRO2
CMP R0, #EVAL7+1; HAS SEARCH FAILED?

BST GTPRO1

CLR R0
DEC R4
POP R3
RTS PC

MOV R3, R0
BR GTPRO3

; MOVTH - MOVSC0, MOVE REGISTERS R3, R2, R1 ONTO THE 
; STACK.

PUSH R3
PUSH R2
PUSH R1
RTS PC

; GETOP - GTPOO, GET AN OPERAND. 
; UPON ENTRY R4 POINTS TO THE START OF AN OPERAND. 
; UPON EXIT R1, R2, R3 CONTAIN THE VALUE OF THE OPERAND 
; IF LEGAL, IF NOT LEGAL, A FATAL ERROR CALL IS MADE. 
; ON LEGAL OR ILLEGAL EXITS, RA WILL ALWAYS POINT 
; ONE CHARACTER AFTER THE SCAN WAS ENDED. NOTE:
; BE CAREFUL - IT MAY, BY WAY 
; OF CALLS TO "EVAL", RE-ENTER ITSELF BEFORE 
; COMPLETION. REGISTERS USED - ALL.

PUSH R4
GETSCR
TESTC
BR GTPI5
BR GTPI8
JMP EFUN
BR GTPO2
POP R4
SUB #6, SP
MOV SP, R0
ATOP
MOV R3
RTS PC

; SAVE TEXTPOINTER.
; GET NONSPACE CHARACTER.
; THE CHARACTER IS A 
; TERMINATOR-GOTO GTPI5.
; NUMERIC.
; FUNCTION.
; ALPHABETIC.
; RESTORE TEXTPOINTER.
; RESERVE SPACE FOR $.
; MOVE ADDRESS FOR $ TO R0.
; ASCII TO NUMERIC CONVERSION.
; PLACE THE $ 
; IN R1, R2, R3.
GET THE NUMBER
AND
RETURN
$DOES \# BEGIN WITH DECIMAL PT?
$YES - GOTO GTPI8.
$NO.
$PLACE VARIABLE ADDRESS IN PT1.
$PLACE VARIABLE ADDRESS IN R0.

NOTE THAT IF VARIABLE DOES NOT EXIST, THEN GETARG
$WILL CREATE THE VARIABLE AND SET IT = 0.

CLRC R1
GETC
SORTC
WORD TERMS
BR EFUN2
ASL R1
ADD CHAR,R1
BR EFUN

COMPLETE TRANSLATION TO
FUNCTION CODENAME.

SAVE FUNCTION CODENAME.
$WAS TERMINATOR A L-PAREN?
$NO.
$YES, EVALUATE FON ARGUMENT.
$RESTORE FUNCTION CODENAME.
$MOVE FON ARGUMENT TO STACK.

IS FUNCTION CODENAME
$IN LIST?
$YES- SKIP NEXT LINE.
$NO.
$GET ADDRESS OF POINTER.
$SOURCE ADDRESS IN R1.
$SAVE CHARACTER POINTER.
$RESERVE SPACE ON STACK.
$DESTINATION ADDRESS.
$GO EXECUTE FUNCTION.
$MOVE ANSWER
$TO REGISTERS R1,R2, AND
R3.
$RESTORE TEXTPOINTER.
$DISCARD USELESS DATA ON STACK.

INCORRECT ARGUMENT.
COMMANDS - SET, FOR, IF, GOTO, ON, DO, COMMENT, ASK, TYPE, LIBRARY.

.BYTE 'E', 'V', 'R', 'O', 'S', 'F', 'A', 'T', 200
; ERASE, WRITE, MODIFY, QUIT, RETURN, ADDITIONAL SPACE FOR
; USER-DEFINED COMMANDS. NEG. NO. ENDS LIST.

.EVEN

GO: SET
FOR
DD: IF
GOTO
ON
DO
COMMENT
ASK
TYPE
LIBRARY
ERASE
WRITE
MODIFY
QUIT
RETURN
*

CRPRNT
CKPRNT

;* - PLACE ADDED COMMAND HERE.

;CR - PRINT CR, LF.
;↑K - PRINT ↑, CR, LF, 6 SPACES.

PLS: .BYTE '(', 'T
LIST: .BYTE '(', ', ', '>', 200  NEG. NO. ENDS LIST.

TOP: AA
AA
AB
AC
AD
IST2 1F1
PROCESS
PCI
PROCESS

MS:
.BYTE 49, '*', '*', '*/9
.BYTE 99, ';', 15, 13, '200

ST1: FINCR
PROCESS
PCI
PROCESS
ST2: FLIMIT
FINFIN
ERROR
FINFIN

IST3: .BYTE 'z', 's', '1', 's', 'S', '200

LIST: TINIR
TQOUY
TCRLF
TCRLF2
TDUMP
TASK4
PROCESS
PCI
PROCESS

IST2: .BYTE 'z', 15, 200
.EVEN

IST3: TASK4
PCI

;CODES FOR FUNCTIONS (LISTED WITHOUT AN INITIAL F).

NTABL: .BYTE 33, 2, 150, 136
 .BYTE 115, 126, 124, 154
 .BYTE 75, 72, 125, 44
 .BYTE 160, 20, 67, 113
 .BYTE 17, 175, 157, 53
 .BYTE 130, 117, 101
.EVEN
<table>
<thead>
<tr>
<th>ADDRESS OF FUNCTION</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>FABS</td>
<td></td>
</tr>
<tr>
<td>FEQ1</td>
<td></td>
</tr>
<tr>
<td>FSGN</td>
<td></td>
</tr>
<tr>
<td>FITR</td>
<td></td>
</tr>
<tr>
<td>FRAC</td>
<td></td>
</tr>
<tr>
<td>FHOD</td>
<td></td>
</tr>
<tr>
<td>FEXP</td>
<td></td>
</tr>
<tr>
<td>FSIN</td>
<td></td>
</tr>
<tr>
<td>FCOS</td>
<td></td>
</tr>
<tr>
<td>FAIV</td>
<td></td>
</tr>
<tr>
<td>FLOG</td>
<td></td>
</tr>
<tr>
<td>FARD</td>
<td></td>
</tr>
<tr>
<td>FOR</td>
<td></td>
</tr>
<tr>
<td>FXOR</td>
<td></td>
</tr>
<tr>
<td>FCOR</td>
<td></td>
</tr>
<tr>
<td>FEOU</td>
<td></td>
</tr>
<tr>
<td>FADC</td>
<td></td>
</tr>
<tr>
<td>FSU</td>
<td></td>
</tr>
<tr>
<td>FTIN</td>
<td></td>
</tr>
<tr>
<td>FDAY</td>
<td></td>
</tr>
<tr>
<td>FRAN</td>
<td></td>
</tr>
<tr>
<td>FLIN</td>
<td></td>
</tr>
<tr>
<td>FILE</td>
<td></td>
</tr>
<tr>
<td>EVEN</td>
<td></td>
</tr>
</tbody>
</table>