DIGITAL RESEARCH

Post Office Box 579, Pacific Grove, California 93950, (408) 373-3403

BATCH PROCESSOR (SUBMIT)

CP/M VERSION _____

SER. # _____

```
00001  1       /* SUBMIT FUNCTION - CREATE $$$.SUB FILE WITH COMMANDS */
00002  1
00003  1   DEAD: DECLARE BDOS LITERALLY '5FEDH', BOOT LITERALLY '0';
00004  1   SUBMIT: PROCEDURE;
00005  2
00006  2
00007  2       /*
00008  2
00009  2
00010  2                       COPYRIGHT (C) GARY A. KILDALL
00011  2                       JUNE, 1975                      */
00012  2
00013  2   DECLARE LIT LITERALLY 'LITERALLY',
00014  2       DCL LIT 'DECLARE',
00015  2       PROC LIT 'PROCEDURE',
00016  2       ADDR LIT 'ADDRESS',
00017  2       CHAR LIT 'BYTE',
00018  2       LCA  LIT '110$0001B',  /* LOWER CASE A */
00019  2       LCZ  LIT '111$1010B',  /* LOWER CASE Z */
00020  2       CTL LIT '5EH',         /* SHIFT - N (CONTROL KEY FLAG) */
00021  2       EOFILE LIT '1AH';      /* CP/M END OF FILE */
00022  2
00023  2   DECLARE
00024  2       (LN1, LN2, LN3) BYTE INITIAL ('001'),
00025  2       FILE(2) BYTE           INITIAL ('$$'),
00026  2       FCBA ADDRESS INITIAL(5CH),
00027  2       BUFA ADDRESS INITIAL(80H),      /* DISK IO BUFFER ADDRESS */
00028  2       DFCB (32) BYTE INITIAL
00029  2           (0,'$$$     ','SUB',0),     /* DESTINATION FILE */
00030  2       REC BYTE,                       /* NEXT RECORD TO WRITE */
00031  2       BUFF BASED BUFA (128) BYTE,     /* DISKIO BUFFER */
00032  2       SFCB BASED FCBA (33) BYTE;/* SOURCE FILE CONTROL */
00033  2
00034  2
00035  2   MON1: PROCEDURE(F,A);
00036  3       DECLARE F BYTE,
00037  3           A ADDRESS;
00038  3       GO TO BDOS;
00039  3       END MON1;
00040  2
00041  2   MON2: PROCEDURE(F,A) BYTE;
00042  3       DECLARE F BYTE,
00043  3           A ADDRESS;
00044  3       GO TO BDOS;
00045  3       END MON2;
00046  2
00047  2   DECLARE
00048  2       TRUE LITERALLY '1',
00049  2       FALSE LITERALLY '0',
00050  2       FOREVER LITERALLY 'WHILE TRUE',
00051  2       CR LITERALLY '13',
00052  2       LF LITERALLY '10',
00053  2       WHAT LITERALLY '63';
00054  2
00055  2   PRINT: PROCEDURE(A);
00056  3       DECLARE A ADDRESS;
00057  3       /* PRINT THE STRING STARTING AT ADDRESS A UNTIL THE
00058  3       NEXT DOLLAR SIGN IS ENCOUNTERED */
00059  3       CALL MON1(9,A);
00060  3       END PRINT;
00061  2
00062  2
00063  2   DECLARE DONT BYTE;
00064  2
00065  2   OPEN: PROCEDURE(FCB);
00066  3       DECLARE FCB ADDRESS;
00067  3       DONT = MON2(15,FCB);
00068  3       END OPEN;
00069  2
00070  2   CLOSE: PROCEDURE(FCB);
00071  3       DECLARE FCB ADDRESS;
00072  3       DONT = MON2(16,FCB);
00073  3       END CLOSE;
00074  2
00075  2   DELETE: PROCEDURE(FCB);
00076  3       DECLARE FCB ADDRESS;
00077  3       CALL MON1(19,FCB);
00078  3       END DELETE;
00079  2
```



```
00080  2   DISKREAD: PROCEDURE(FCB) BYTE;
00081  3       DECLARE FCB ADDRESS;
00082  3       RETURN MON2(20,FCB);
00083  3       END DISKREAD;
00084  2
00085  2   DISKWRITE: PROCEDURE(FCB) BYTE;
00086  3       DECLARE FCB ADDRESS;
00087  3       RETURN MON2(21,FCB);
00088  3       END DISKWRITE;
00089  2
00090  2   MAKE: PROCEDURE(FCB);
00091  3       DECLARE FCB ADDRESS;
00092  3       DONT = MON2(22,FCB);
00093  3       END MAKE;
00094  2
00095  2   MOVE: PROCEDURE(S,D,N);
00096  3       DECLARE (S,D) ADDRESS, N BYTE;
00097  3       DECLARE A BASED S BYTE, B BASED D BYTE;
00098  3           DO WHILE (N := N - 1) <> 255;
00099  3           B = A; S = S + 1; D = D + 1;
00100  4           END;
00101  3       END MOVE;
00102  2
00103  2   CRLF: PROCEDURE;
00104  3       CALL PRINT(.(CR,LF,'$'));
00105  3       END CRLF;
00106  2
00107  2   DECLARE OLDSP ADDRESS;   /* CALLING PROGRAM'S STACK POINTER */
00108  2
00109  2   ERROR: PROCEDURE(A);
00110  3       DECLARE A ADDRESS;
00111  3       CALL CRLF;
00112  3       CALL PRINT(.'ERROR ON LINE $');
00113  3       CALL PRINT(.LN1);
00114  3       CALL PRINT(A);
00115  3       CALL CRLF;
00116  3       STACKPTR = OLDSP;
00117  3       /* RETURN TO CCP */
00118  3       END ERROR;
00119  2
00120  2   DECLARE SSTRING(128) BYTE,      /* SUBSTITUTE STRINGS */
00121  2       SBP BYTE;                   /* POINTER INTO SSTRING */
00122  2
00123  2
00124  2   SETUP: PROCEDURE;
00125  3       /* X.SUB FILE IS IN FIRST 16 BYTES, SUBSTITUTE STRING IN LAST 16 */
00126  3       DECLARE (I,B,ST) BYTE;
00127  3       DELIMITER: PROCEDURE BYTE;
00128  4           /* RETURNS TRUE IF POSITION I OF BUFF IS A DELIMITER */
00129  4           RETURN (B := BUFF(I)) = ' ';
00130  4           END DELIMITER;
00131  3       /* COPY SUBSTITUTE STRINGS */
00132  3       I,SBP = 0;
00133  3           DO WHILE (I:=I+1) <= BUFF;
00134  3           /* GOBLANK AND COLLECT STRING */
00135  3               DO WHILE DELIMITER; I = I + 1;
00136  5               END;
00137  4           /* START OF STRING */
00138  4           ST = SBP;
00139  4               DO WHILE NOT DELIMITER AND I <= BUFF;
00140  4               IF (SBP := SBP +1) >= LAST(SSTRING) THEN
00141  5                   CALL ERROR(.'PARAMETER LIST TOO LONG$');
00142  5               SSTRING(SBP) = B;
00143  5               I = I + 1;
00144  5               END;
00145  4           /* SAVE LENGTH */
00146  4           SSTRING(ST) = SBP - ST;
00147  4           SBP = SBP + 1;
00148  4           END;
00149  3       /* MARK END OF LIST */
00150  3       SSTRING(SBP) = 0;
00151  3       CALL MOVE(.'SUB',FCBA+9,3); /* SET FILE TYPE TO SUB */
00152  3       CALL OPEN(FCBA);
00153  3       IF DONT = 255 THEN
00154  3           CALL ERROR(.'NO ''SUB'' FILE PRESENT$');
00155  3       /* OTHERWISE FILE IS OPEN - READ SUBSEQUENT DATA */
00156  3       SBP = 128; /* CAUSES READ BELOW */
00157  3
00158  3       END SETUP;
00159  2
```

```
00160  2
00161  2   GETSOURCE: PROCEDURE BYTE;
00162  3       /* READ THE NEXT SOURCE CHARACTER */
00163  3       DECLARE B BYTE;
00164  3       IF SBP > 127 THEN
00165  3           DO; IF DISKREAD(FCBA) <> 0 THEN
00166  4               RETURN ENDFILE;
00167  4               SBP = 0;
00168  4           END;
00169  3       IF (B := BUFF(SBP:=SBP+1)-1)) = CR THEN
00170  3           DO; /* INCREMENT LINE # */
00171  3           IF (LN3 := LN3 + 1) > '9' THEN
00172  4               DO; LN3 = '0';
00173  5               IF (LN2 := LN2 + 1) > '9' THEN
00174  5                   DO; LN2 = '0';
00175  6                   LN1 = LN1 + 1;
00176  6                   END;
00177  5               END;
00178  4           END;
00179  3       RETURN B;
00180  3       END GETSOURCE;
00181  2
00182  2   WRITEBUFF: PROCEDURE;
00183  3       /* WRITE THE CONTENTS OF THE BUFFER TO DISK */
00184  3       IF DISKWRITE(.DFCB) <> 0 THEN /* ERROR */
00185  3           CALL ERROR(.'DISK WRITE ERROR$');
00186  3       END WRITEBUFF;
00187  2
00188  2   DECLARE RBUFF(1024) BYTE, /* JCL BUFFER */
00189  2       RBP ADDRESS,      /* JCL BUFFER POINTER */
00190  2       RLEN BYTE;        /* LENGTH OF CURRENT COMMAND */
00191  2
00192  2   FILLRBUFF: PROCEDURE;
00193  3       PUTRBUFF: PROCEDURE(B);
00194  4           DECLARE B BYTE;
00195  4           IF (RBP := RBP + 1) > LAST(RBUFF) THEN
00196  4               CALL ERROR(.'JCL BUFFER OVERFLOW$');
00197  4           RBUFF(RBP) = B;
00198  4           IF (RLEN := RLEN + 1) > 126 THEN
00199  4               CALL ERROR(.'COMMAND TOO LONG$');
00200  4           END PUTRBUFF;
00201  3
00202  3       DECLARE (READING,I,TBP,B) BYTE;
00203  3       NONZERO: PROCEDURE BYTE;
00204  4           RETURN (I := SSTRING(TBP) + 1) > 1;
00205  4           END NONZERO;
00206  3       /* FILL THE JCL BUFFER */
00207  3       RBUFF,RBP = 0;
00208  3       READING = TRUE;
00209  3           DO WHILE READING;
00210  3           RLEN = 0; /* RESET COMMAND LENGTH */
00211  4               DO WHILE (B:=GETSOURCE) <> ENDFILE AND B <> CR;
00212  4               IF B = CTL THEN /* CONTROL KEY */
00213  5                   CALL PUTRBUFF(GETSOURCE AND 1$11111B); ELSE
00214  5               IF B <> LF THEN
00215  5                   DO; IF B = '$' THEN /* COPY SUBSTITUTE STRING */
00216  6                       DO; IF (B := GETSOURCE - '0') > 9 THEN
00217  7                           CALL ERROR(.'INVALID PARAMETER SPECIFICATION$');
00218  7                       TBP = 0; /* FIND STRING B */
00219  7                           DO WHILE (B := B - 1) <> 255;
00220  7                           IF NONZERO THEN TBP = TBP + I;
00221  8                           END;
00222  7                       IF NONZERO THEN /* COPY TO RBUFF */
00223  7                           DO WHILE (I := I - 1) <> 0;
00224  7                           CALL PUTRBUFF(SSTRING(TBP := TBP + 1));
00225  8                           END;
00226  7                   END; ELSE /* NOT A '$' */
00227  6                   CALL PUTRBUFF(B);
00228  6                   END;
00229  5               END; /* OF LINE OR INPUT FILE - COMPUTE LENGTH */
00230  4           READING = B = CR;
00231  4           CALL PUTRBUFF(RLEN); /* STORE LENGTH */
00232  4           END;
00233  3       /* ENTIRE FILE HAS BEEN READ AND PROCESSED */
00234  3       END FILLRBUFF;
00235  2
00236  2   MAKEFILE: PROCEDURE;
00237  3       /* WRITE RESULTING COMMAND FILE */
00238  3       DECLARE I BYTE;
00239  3       GETRBUFF: PROCEDURE BYTE;
00240  4           RETURN RBUFF(RBP := RBP - 1);
00241  4           END GETRBUFF;
00242  3
00243  3       CALL DELETE(.DFCB);
00244  3       DREC = 0; /* ZERO THE NEXT RECORD TO WRITE */
00245  3       CALL MAKE(.DFCB);
00246  3       IF DCNT = 255 THEN CALL ERROR(.'DIRECTORY FULL$');
00247  3           DO WHILE (I := GETRBUFF) <> 0;
00248  3           /* COPY I CHARACTERS TO BUFFER */
00249  3           BUFF = I; BUFF(I+1) = '$';
00250  4               DO WHILE I > 0;
00251  4               BUFF(I) = GETRBUFF; I=I-1;
00252  5               END;
00253  4           /* BUFFER FILLED TO $ */
00254  4           CALL WRITEBUFF;
00255  4           END;
00256  3       CALL CLOSE(.DFCB);
00257  3       IF DCNT = 255 THEN CALL ERROR(.'CLOSE ERROR$');
00258  3       END MAKEFILE;
00259  2
00260  2   /* ENTER HERE FROM THE CCP WITH THE FCB SET */
00261  2   DECLARE STACK(10) ADDRESS;  /* WORKING STACK */
00262  2       OLDSP = STACKPTR;
00263  2       STACKPTR = .STACK(LENGTH(STACK));
00264  2
00265  2       CALL SETUP;
00266  2       CALL FILLRBUFF;
00267  2       CALL MAKEFILE;
00268  2       GO TO BOOT; /* REBOOT CAUSES COMMANDS TO BE EXECUTED */
00269  2       END SUBMIT;
00270  1   EOF
```


CP/M VERSION
COPYRIGHT © 1976
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. # _____