ABSTRACT

THE FOLLOWING ABSTRACT IS THAT OF PE-T-502,THE INTERNAL STRUCTURE OF THE NEW EMULATORS BY ADRIAN TAGGART, SAN CARSWELL, JOHN HOWELL, DAVE GURTEEN, BARBARA JEREMIAH, GARY FREEMAN, KAROL PRZYBYS.

PT-T-502 IS INTENDED TO BE A MORE DETAILED ADDITION TO THE EMULATOR HANDBOOK. IT SHOULD PROVIDE SUFFICIENT INFORMATION TO UNDERSTAND AND MAKE ALTERATIONS TO THE EMULATOR PROGRAMS.

BY : ADRIAN TAGGART
SAN CARSWELL
JOHN HOWELL
DAVE GURTEEN
BARBARA JEREMIAH
GARY FREEMAN
KAROL PRZYBYS

SUBJECT: THE INTERNAL STRUCTURE OF THE NEW EMULATORS

CONTENTS

INTRODUCTION

1. COMMON USER INTERFACE INTERNALS (DAVE GURTEEN - UK)
    1.1 INITIALISATION
    1.2 COMMAND LINE INPUT
    1.3 COMMAND LINE INTERPRETATION
    1.4 COMMAND PROCESSING
    1.5 SMLC LINES, SITES, ETC.
    1.6 TERMINAL OUTPUT
    1.7 ERROR MESSAGES
    1.8 LOW LEVEL TERMINAL OUTPUT ROUTINES
    1.9 FILE UNIT MANAGEMENT
    1.10 CHARACTER HANDLING
    1.11 CONCAT FILE FORMAT
    1.12 COMMON AREAS AND INSERT FILES

2. EMSCOM COMMON MODULES (ADRIAN TAGGART - UK)

3. EMULATOR SPECIFIC MODULES (ADRIAN TAGGART - UK)

4. 1004 TRANSMIT AND RECEIVE MODULES (JOHN HOWELL - UK)

5. 200UT TRANSMIT AND RECEIVE MODULES (SAN CARSWELL - UK)

6. 2780 TRANSMIT AND RECEIVE MODULES (KAROL PRZYBYS - UK)

7. 7020 TRANSMIT AND RECEIVE MODULES (BARBARA JEREMIAH - UK)

8. GRTS TRANSMIT AND RECEIVE MODULES (JOHN HOWELL - UK)

9. HASP TRANSMIT AND RECEIVE MODULES (GARY FREEMAN - UK)

10. THE SYMBIONTS.

INTRODUCTION

THIS PE-T IS INTENDED TO BE A MORE DETAILED ADDITION TO THE EMULATOR HANDBOOK. IT SHOULD BE READ TO GAIN SUFFICIENT INFORMATION TO UNDERSTAND AND MAKE ALTERATIONS TO THE EMULATOR PROGRAMS.

SECTION 1 DESCRIBES THE INTERNAL STRUCTURE OF FILES AND PROGRAMS CONCERNED WITH THE USER INTERFACE.

SECTION 2 DESCRIBES THE STRUCTURE OF THOSE MODULES CONCERNED WITH THE SYMBIONT/USER INTERFACE COMMUNICATION.

SECTION 3 DESCRIBES THE MODULES WHICH ARE SPECIFIC TO EACH EMULATOR.

SECTIONS 4 TO 9 DESCRIBE THE RECEIVE AND TRANSMIT PROCESSING MODULES FOR EACH EMULATOR, I.E. SECTION 4 - 1004, SECTION 5 - 200UT, SECTION 6 - 2780, SECTION 7 - 7020, SECTION 8 - GRTS, SECTION 9 - HASP.

SECTION 10 IS A NOTE ON THE SYMBIONTS USED FOR THIS RELEASE OF THE EMULATOR SOFTWARE.

SECTION 1. COMMON USER INTERFACE INTERNALS

THE OVERALL STRUCTURE OF THE USER INTERFACE OF THE RJE EMULATORS IS NOW TOTALLY CONSISTENT. THE EMULATORS HAVE A COMMON COMMAND INTERPRETER, AND A LARGE SET OF THE COMMAND PROCESSING ROUTINES, OUTPUT ROUTINES, AND MANY OTHER ROUTINES THAT ARE ALSO COMMON. THE ROUTINES THAT PERFORM SIMILAR FUNCTIONS ARE STRUCTURED SIMILARLY AND 'LOOK' AND 'OPERATE' IN A SIMILAR MANNER.

THE FOLLOWING SECTIONS ARE CONCERNED WITH THE PHILOSOPHIES AND DETAILS OF THIS COMMON USER INTERFACE. IT IS ASSUMED THAT THE READER HAS READ, AND UNDERSTOOD THE EMULATOR HANDBOOK. THE CODE IS REASONABLY WELL STRUCTURED AND COMMENTED (BY PRIME STANDARDS) AND SHOULDN'T PROVIDE THE ANALYST WITH TOO MUCH DIFFICULTY IN UNDERSTANDING IT. NOTE THAT ALL VARIABLES HAVE BEEN DECLARED (IN SOMUCH AS THE FORTRAN - DCLVAR OPTION HASN'T OBJECTED) , AND IN GENERAL EACH IS DESCRIBED BY A COMMENT.

ALL COMMON CODE IS HELD IN THE DIRECTORY RJECOM. THERE ARE FIVE SOURCE FILES, EACH CONTAINING A NUMBER OF SUBROUTINES. THE NAMES OF THE SOURCE FILES ARE PARTIALLY HISTORICAL AND NOT TOO MUCH SIGNIFICANCE SHOULD BE ATTACHED TO THEM OR THEIR CONTENTS

SUFFICE TO SAY THEY MAKE UP ALL THE COMMON CODE. THE NAMES, AND BRIEF DESCRIPTIONS ARE:

CNTRL
THIS FILE CONTAINS THE MAIN CONTROL ROUTINES.

RDLIB
THIS FILE CONTAINS THE LOW LEVEL OUTPUT ROUTINES, AND OTHER ROUTINES THAT ARE ALMOST EMULATOR INDEPENDENT.

EMLIB
THIS FILE CONTAINS THE BULK OF THE EMULATOR COMMON ROUTINES, THAT ARE HIGHLY EMULATOR DEPENDENT.

TTYRDY
THIS FILE CONTAINS THE ONLY PMA ROUTINE, NAMELY TTYRDY ITSELF, THE ROUTINE THAT LOOKS TO SEE IF THERE IS A CHARACTER IN THE USERS AMLC BUFFER.

EMSCOM
THIS FILE CONTAINS MAINLY COMMON SYMBIONT COMMUNICATION ROUTINES AND ITS CONTENTS ARE DESCRIBED ELSEWHERE IN THIS DOCUMENT.

LET'S NOW DESCRIBE THE OVERALL STRUCTURE OF AN RJE EMULATOR FROM THE COMMON USER INTERFACE VIEWPOINT ONLY. THE FIRST IMPORTANT POINT TO NOTE IS THAT THERE ARE APPARENTLY TWO PROGRAMS, THE 'USER SEND UTILITY' TO SEND JOBS, AND THE 'OPERATOR WORKSTATION' TO CONTROL ALL OPERATIONS. IN FACT, THERE IS ONLY ONE PROGRAM. THIS PROGRAM IS THE ONE DESCRIBED HERE AND IT DECIDES IT IS THE SEND UTILITY OR THE WORKSTATION BY EXAMINING THE COMMAND LINE THAT INVOKED IT. IF WE TAKE HASP FOR

EXAMPLE, THE PROGRAMS WSHASP AND RJHASP ARE IDENTICAL, THE PROGRAM LOOKS FOR THE CHARACTERS 'RJ' AT THE START OF THE COMMAND LINE, IF IT FINDS A MATCH IT SETS A FLAG 'WS' FALSE, IF NOT, THEN 'WS' IS SET TRUE. THE PROGRAM MAKES FREQUENT REFERENCE TO THIS FLAG TO DECIDE ITS MODE OF OPERATION. FROM NOW ON THE EMPHASIS WILL BE ON THE WORKSTATION, AS IT IS THE MORE IMPORTANT MODE OF OPERATION, AND IN GENERAL THE FACILITIES OF THE SEND UTILITY ARE A SUB-SET OF THAT OF THE WORKSTATION.

THE SECOND MAIN POINT TO MAKE IS THAT THIS PROGRAM IS FUNDAMENTALLY CONCERNED WITH COMMAND PROCESSING FOR ALL THE EMULATORS. YOU WILL NO DOUBT HAVE DISCOVERED THAT THERE ARE OVER 80 DIFFERENT COMMANDS. MANY OF THESE COMMANDS ARE COMMON TO ALL OR THE MAJORITY OF THE EMULATORS, OTHERS ARE QUITE SPECIFIC TO AN INDIVIDUAL EMULATOR. TO HANDLE THIS SITUATION THERE IS ONE MAIN COMMAND LIST (DESCRIBED IN MORE DETAIL LATER), EACH COMMAND IS 'MARKED' AS A 'GENERAL COMMAND' OR AS A 'SPECIFIC COMMAND'. THERE ARE THUS TWO COMMAND PROCESSING ROUTINES MGENRL AND MSPCFC. THERE IS ONLY ONE MGENRL ROUTINE TO PROCESS THE COMMON COMMANDS BUT AN MSPCFC ROUTINE FOR EACH INDIVIDUAL EMULATOR. A CERTAIN AMOUNT OF EVOLUTION WAS PRESENT IN THE DESIGN AND CODING OF THE EMULATORS, AND CONSEQUENTLY A NUMBER OF COMMON COMMANDS ARE PROCESSED BY MSPCFC, AND EVEN ONE OR TWO SPECIFIC COMMANDS ARE PROCESSED BY MGENRL, BUT THIS FACT IS REALLY QUITE IRREVELANT TO THE WORKING OF THE EMULATORS.

THE THIRD POINT IS THAT THERE IS ONLY ONE MAIN CONTROL ROUTINE FOR ALL THE EMULATORS, IT IS CALLED 'CNTRL' AND IS TO BE FOUND IN RJECOM>CNTRL. WE ARE NOW IN THE POSITION TO UNDERSTAND THE SKELETAL STRUCTURE OF AN EMULATOR.

THE SPECIFIC CODE FOR EACH EMULATOR IS HELD IN A UFD CALLED RJEXXXXXX, WHERE XXXXXX IS THE NAME OF THE EMULATOR, FOR HASP THE UFD IS RJEHASP. IN THIS UFD ARE THE BUILD MACROS FOR THAT INDIVIDUAL EMULATOR, THE MAIN SOURCE IS CALLED WSXXXXXX, AGAIN FOR HASP, WSHASP, THIS SOURCE CONTAINS THE ENTRY POINT TO THE EMULATOR PROGRAM:-

```
        SUBROUTINE MAIN
        CALL CNTRL('XXXXXX')
        END
```

FOR HASP:-

```
        SUBROUTINE MAIN
        CALL CNTRL('HASP ')
        END
```

THAT IS A CALL TO THE COMMON CONTROL ROUTINE WITH THE EMULATOR NAME AS A SINGLE PARAMETER. THIS IS FOLLOWED BY THE SPECIFIC ROUTINE MSPCFC FOR THAT EMULATOR.

LET'S NOW DESCRIBE THE COMMON CONTROL ROUTINE 'CNTRL', BEFORE THIS IS DONE THE FOURTH AND MOST MAJOR POINT CONCERNING THE WORKSTATION PROGRAM MUST BE MADE, AND THAT IS IT PERFORMS TWO FUNCTIONS.

1. TO PROCESS USER/OPERATOR COMMANDS.
2. TO PROCESS RECEIVE FILES, BOTH OPERATOR MESSAGES AND RECEIVE DATA.

TO PERFORM BOTH THESE FUNCTIONS 'IN PARALLEL', THE PROGRAM (UNLIKE MOST PRIME PROGRAMS) NEVER WAITS ON COMMAND INPUT, BUT BUILDS UP A COMMAND LINE A CHARACTER AT A TIME, PERFORMING RECEIVE PROCESSING BETWEEN THE PROCESSING OF COMMAND LINES. A COMMAND LINE IS ONLY PROCESSED WHEN A .CR. CHARACTER IS INPUT. FOR THOSE OF YOU FAMILIAR WITH THE OLD HASP EMULATOR, THIS IS A VERY SIMILAR PROCEDURE.

A SKELETAL 'CNTRL' ROUTINE WRITTEN IN PSEUDO FORTRAN WOULD APPEAR AS FOLLOWS:-

```
LOOP      READ CHARACTERS AND BUILD UP A COMMAND LINE
          IF COMMAND LINE IS INCOMPLETE GOTO RECEIVE
          PROCESS COMMAND LINE
          GOTO LOOP
RECEIVE   IF NOT RECEIVE PROCESSING GOTO SLEEP
          PERFORM RECEIVE PROCESSING
          GOTO LOOP
SLEEP     SLEEP AWHILE
          GOTO LOOP
```

NOW HAVE A LOOK AT THE ROUTINE 'CNTRL' ITSELF, IT IS MORE SOPHISTICATED IN DETAIL THAN THE PSEUDO CODE ABOVE AND PERFORMS ONE OR TWO OTHER FUNCTIONS, BUT THE GENERAL PHILOSOPHY SHOULD BE APPARENT. NOTE THAT IT IS PERFORMING RECEIVE PROCESSING FOR UP TO 'NLINE' REMOTE SITES.

THE REMAINDER OF THIS SECTION IS NOW BROKEN UP INTO SUB-SECTIONS THAT DETAIL INDIVIDUAL ASPECTS OF THE USER INTERFACE IN MORE DEPTH.

1.1 INITIALIZATION

ALL PROGRAM INITIALIZATION IS PERFORMED BY THE ROUTINE INIT, THE FIRST ROUTINE CALLED BY CNTRL, IT PERFORMS A WIDE RANGE OF INITIALIZATION FUNCTIONS, WHICH ARE BEST UNDERSTOOD BY REFERENCE TO THE CODE. NOTE THE CALL TO THE ROUTINE XCMD IF THE PROGRAM IS NOT THE WORKSTATION. IF THE USER HAS INVOKED THE RJ UTILITY IN THE FORM:-

```
RJHASP <FILE> -TO <SITE>
```

THEN XCMD TAKES OVER COMMAND PROCESSING, AND NEVER RETURNS TO INIT, AND CONSEQUENTLY CNTRL. XCMD IS A MINI-CONTROL ROUTINE FOR THIS FORM OF PROGRAM INVOCATION.

1.2 COMMAND LINE INPUT

THE EMULATOR WORKSTATION PROGRAM IS UNLIKE MOST PRIME PROGRAMS IN THAT IT NEVER WAITS ON COMMAND INPUT. WHEN A COMPLETE COMMAND LINE IS NOT AVAILABLE FROM THE USER TERMINAL, THEN THE PROGRAM PERFORMS OTHER NECESSARY PROCESSING.

TO FACILITATE THIS, A LOGICAL FUNCTION CALLED TTYRDY IS CAPABLE OF

EXAMINING THE USERS AMLC INPUT BUFFER, AND SETTING ITSELF TRUE OR FALSE IF THERE ARE ANY CHARACTERS IN THAT BUFFER. THE HIGH LEVEL ROUTINE RDLINE MAKES USE OF THIS FUNCTION TO BUILD UP A COMPLETE COMMAND LINE, A CHARACTER AT A TIME. IF THE AMLC INPUT BUFFER IS EMPTY, AND THE LINE BUILT UP BY RDLINE IS INCOMPLETE, I.E., NO CARRIAGE RETURN CHARACTER HAS BEEN INPUT, THEN RDLINE RETURNS WITH THE 'LINE COMPLETE' FLAG LCOM SET FALSE. WHEN THE NEW LINE CHARACTER IS INPUT BY RDLINE THEN LCOM IS SET TRUE, AND RDLINE AGAIN RETURNS.

THE PROGRAM HAS THREE COMMAND INPUT STREAMS DEFINED BY THE VARIABLE TTY.

TTY=1 THE INPUT STREAM IS THE TERMINAL, CALL T1IN(CHAR)
TTY=2 THE INPUT STREAM IS COMINPUT ON UNIT 6, CALL C1IN(CHAR)
TTY=3 THE INPUT STREAM IS A FILE (THE SITE FILE) CALL X1IN(CHAR)

THE PROGRAM SETS THE STREAM VARIABLE AS APPROPRIATE, AND RDLINE READS FROM THE 'TTY' STREAM.

NOTE THAT ERASE AND KILL CHARACTERS ARE HANDLED, HAVING BEEN OBTAINED BY ERKL$$ AT INITIALIZATION. THE COMMAND INPUT LINE IPLINE IS ALWAYS TRUNCATED TO 80 CHARACTERS.

1.3 COMMAND LINE INTERPRETATION

THE COMMAND LINE HAVING BEEN INPUT BY RDLINE IS INTERPRETED BY THE ROUTINE PARSE. ALL TOKEN HANDLING IS PERFORMED BY THE ROUTINE RDTK, THIS IS AN ADAPTED VERSION OF RDTK$$, THE MAJOR DIFFERENCE BEING THAT RDTK CAN READ FROM ITS OWN COMMAND LINE BUFFER.

THE CALLING SEQUENCE TO RDTK AND ITS RETURN DATA IS IDENTICAL TO THAT OF RDTN$$. THIS INFORMATION IS DOCUMENTED IN THE PRIME REFERENCE GUIDE TO THE FILE MANAGEMENT SYSTEM (PDR3110).

THE WORKSTATION COMMAND ENVIRONMENT IS COMPLICATED BY THE FACT THAT IT SUPPORTS:

1. SUB-MENUS, E.G. CONCAT AND DEFSITE.

2. ALLOWS RAW DATA INPUT IN PARTICULAR COMMAND MODES, E.G. OPMODE.

THE SYSTEM MUST ALSO NOT WAIT ON COMMAND INPUT. SOME OF THE ABOVE MODES ARE ALSO TERMINATED BY A .CR. OR BY THE QUIT COMMAND.

THERE ARE EIGHT MODES IN WHICH COMMANDS MAY BE ENTERED IN THE SYSTEM:

1. COMMAND MODE
2. 'OPMODE' MODE
3. 'OPMESS' MODE
4. 'SIGNON' MODE
5. 'SIGNOFF' MODE
6. 'CONCAT' MODE
7. 'CONCAT' 'JCL' MODE
8. 'DEFSITE' MODE

IN EACH OF THE EIGHT MODES RAW TEXT, .CR., AND THE 'QUIT' COMMAND MAY BE INTERPRETED AS UNIQUE COMMANDS, EACH DIFFERENT IN THEIR OWN RIGHT.

FOR EXAMPLE, A .CR. IN MODE 1, (THE COMMAND MODE) IS IGNORED, BUT A .CR. IN MODE 2, (OPMODE), MEANS TERMINATE OPMODE AND RETURN TO MODE 1, (NORMAL COMMAND LEVEL). THESE ARE REFERRED TO AS PHANTOM COMMANDS.

THE COMMAND DATA STRUCTURE IS HELD IN THE FILE RJECOM>CCMD.

THERE ARE FIVE ARRAYS HOLDING THE MAIN COMMAND DATA.

COM

THIS ARRAY HOLDS THE COMMAND NAME LIST.

GO

THIS ARRAY HOLDS THE COMMAND 'COMPUTED GOTO NUMBER' (COMMAND NUMBER). IF THIS VALUE IS POSITIVE, THE COMMAND IS AN EMULATOR COMMON COMMAND, IF IT IS NEGATIVE, IT IS AN EMULATOR SPECIFIC COMMAND.

NP

THIS ARRAY HOLDS COMMAND PARAMETER DATA IN AN ENCODED FORM, IT IS FULLY DESCRIBED IN THE FILE.

CM

THIS ARRAY HOLDS THE COMMAND MODE THAT THE SYSTEM MUST ENTER ON RECEIPT OF THAT COMMAND.

AM

THIS ARRAY HOLDS THE COMMAND MODE THAT THE SYSTEM MUST BE IN ON RECEIPT OF THAT COMMAND.

CERTAIN COMMANDS IN THE COMMAND LIST HAVE BLANK NAMES, THESE ARE THE PHANTOM COMMANDS REFERRED TO EARLIER.

THE ARRAYS ITEXT, ICR, AND IQUIT ARE DIMENSIONED TO 8, ONE ELEMENT FOR EACH INPUT MODE, THEY CONTAIN THE PHANTOM COMMAND INDEX NUMBERS.

THE FUNCTION OF PARSE CAN NOW BE SEEN TO EITHER LOCATE THE COMMAND IN THE MAIN COMMAND LIST OR OBTAIN THE COMMAND NUMBER BY REFERENCE TO THE ABOVE THREE ARRAYS DEPENDING ON THE INPUT MODE.

IT ALSO PERFORMS THE FOLLOWING (NOT NECESSARILY IN THIS ORDER):

1. CHECKS THAT THE COMMAND IS ALLOWED IN THE CURRENT MODE.

2. RESETS THE CURRENT MODE AS REQUIRED.

3. COUNTS THE COMMAND LINE TOKENS, AND PARTIALLY CHECKS THE VALIDITY OF THIS COUNT.

4. THE SPECIAL FORMS OF THE QUIT COMMAND ARE TRAPPED AT AN EARLY STAGE BY THE FUNCTION 'LQUIT'.

5. THE SPECIAL TECHNIQUE OF PREFIXING AN 'OPMODE' COMMAND LINE WITH THE DESTINATION SITE IS TRAPPED BY THE ROUTINE 'PSITE'.

THE FIRST PARAMETER MAY BE READ INTO THE PARAMETER INPUT BUFFER. THIS WHOLE PROCEDURE IS PERFORMED SEPARATELY BY THE ROUTINE PCOUNT.

1.4 COMMAND PROCESSING

IN THE SYSTEM THERE ARE TWO ROUTINES THAT ACTUALLY PERFORM THE COMMAND PROCESSING. THE ROUTINE MGENRL PROCESSES GENERAL EMULATOR COMMANDS, WHILST MSPCFC PROCESSES SPECIFIC EMULATOR COMMANDS. THERE IS ONLY ONE ROUTINE MGENRL FOR ALL THE EMULATORS, BUT THERE IS AN MSPCFC ROUTINE FOR EACH INDIVIDUAL EMULATOR.

COMMANDS THAT APPLY TO ALL EMULATORS, AND WHICH ARE PROCESSED IN AN IDENTICAL MANNER ARE HANDLED BY MGENRL, COMMANDS THAT ARE SPECIFIC TO AN INDIVIDUAL EMULATOR, OR ARE PROCESSED DIFFERENTLY ARE HANDLED BY MSPCFC, THOUGH THE ACTUAL DIVIDING LINE IS NOT QUITE AS CLEAR CUT AS THIS. GENERAL COMMANDS HAVE A POSITIVE COMMAND NUMBER, SPECIFIC COMMANDS A NEGATIVE ONE. (SEE THE GO ARRAY IN RJECOM>CCMD).

THE STRUCTURE OF MGENRL AND MSPCFC IS IDENTICAL, EACH ROUTINE CHECKS THE VALIDITY OF THE COMMAND NUMBER, AND THEN PERFORMS A 'COMPUTED GOTO' ON THAT NUMBER TO THE APPROPRIATE COMMAND PROCESSING CODES IN THE ROUTINES. NOTE THAT THE ROUTINE FINCOM MUST ALWAYS BE CALLED TO CLEAR THE COMMAND BUFFERS AND FLAGS, TO ENABLE RDLINE TO READ THE NEXT COMMAND LINE.

1.5 SMLC LINES, SITES ETC.

THE COMMON AREA CLINE IN RJECOM>CLINE HOLDS ALL DATA PERTAINING TO THE SMLC LINES AND THEIR SITE ASSOCIATIONS.

EACH EMULATOR IS CAPABLE OF RUNNING UP TO FOUR SMLC LINES TO FOUR SEPARATE SITES. THE PARAMETER 'NLINE' IN RJECOM>PARAM MAY BE SET BETWEEN ONE AND FOUR DEPENDING ON A USERS REQUIREMENTS. BY REDUCING ITS VALUE TO THE MAXIMUM VALUE A USER REQUIRES, AND RECOMPILING AND LOADING THE SYSTEM THE MEMORY REQUIREMENTS OF THE PROGRAM WILL BE GREATLY REDUCED.

THERE ARE SIX ARRAYS, EACH DIMENSIONED TO NLINE. EACH CORRESPONDING ARRAY ELEMENT CONTAINS INFORMATION CONCERNING A SITE.

ALINE CONTAINS THE SMLC LINE NUMBER 0-3. A VALUE OF -2 MEANS THAT THERE IS NO SITE IN THE SLOT, A VALUE OF -1 MEANS THAT ALTHOUGH A SITE EXISTS IN THE SLOT, IT IS NOT FULLY DEFINED.

ASITE CONTAINS THE FOUR CHARACTER SITE NAME.

ELINE CONTAINS A BOOLEAN FLAG, SET TRUE IF THE LINE IS ENABLED,

FALSE IF IT IS DISABLED.

WLINE CONTAINS A KEY, A VALUE OF 1 MEANS THAT RECEIVE PROCESSING IS ENABLED, A VALUE OF 0 MEANS THAT RECEIVE PROCESSING IS ENABLED, BUT HAS BEEN NOTIFIED TO STOP, A VALUE OF -1 MEANS THAT RECEIVE PROCESSING IS DISABLED.

RTODO CONTAINS A BOOLEAN FLAG, SET TRUE IF RECEIVE PROCESSING IS CURRENTLY ACTIVE, SET FALSE IF IT IS NOT.

CREAD CONTAINS A BOOLEAN FLAG, SET TRUE IF THE SITE DEFINITION FILE HAS BEEN READ, SET FALSE IF IT HAS NOT.

THE SOFTWARE 'THINKS' IN TERMS OF THE ARRAY INDEX NUMBER TO THESE SIX ARRAYS, USUALLY CALLED XLINE. THE REAL SMLC LINE NUMBER ALINE(XLINE) IS ONLY USED WHERE EXPLICITLY REQUIRED (FOR EXAMPLE, COMMUNICATION WITH THE OPERATOR).

THE FOLLOWING ROUTINES ARE USED TO ACCESS THIS DATABASE.

GLINE

E.G. XLINE=GLINE(SITE,ICODE)

GIVEN A DEFINED SITE NAME THIS ROUTINE RETURNS ITS ARRAY INDEX NUMBER.

FLINE

E.G. XLINE=FLINE(ICODE)

THIS ROUTINE RETURNS A FREE ARRAY SLOT FOR A SITE.

GSLOT

E.G. XLINE=GSLOT(SMLC,ICODE)

GIVEN A CONFIGURED SMLC LINE NUMBER THIS ROUTINE RETURNS ITS ARRAY INDEX NUMBER.

1.6 TERMINAL OUTPUT

TERMINAL OUTPUT IS NOT DIRECTLY PERFORMED BY THE PRIME STANDARD OUTPUT ROUTINES, BUT BY A SET OF ROUTINES SPECIALLY PROVIDED FOR THE EMULATORS. THESE PROVIDE MORE FUNCTIONALITY THAN THE STANDARD ROUTINES, AND HANDLE THE PROBLEMS ASSOCIATED WITH CONCURRENT COMMAND LINE PROCESSING AND RECEIVE PROCESSING.

OPERATOR MESSAGES MAY BE OUTPUT TO THE TERMINAL AT ANY TIME, GARBLING THE COMMAND THAT THE USER MAY BE TYPING AT THE TIME, AND DESTROYING THE INVITE TO TYPE PROMPT , EVEN IF HE IS NOT TYPING. TO OVERCOME THIS PROBLEM, EVERY TIME A NEWLINE IS OUTPUT, THEN A NEW PROMPT IS ALSO OUTPUT PLUS ANY COMMAND LINE CHARACTERS THAT THE USER HAS TYPED UP TO THAT MOMENT, THUS THE COMMAND LINE IS ALWAYS SHOWN ON THE TERMINAL IN A READABLE FORM.

CONVERSELY WHEN DATA IS OUTPUT TO THE TERMINAL, USER COMMAND INPUT MAY GARBLE THE OUTPUT, AND MAKE IT UNREADABLE. TO OVERCOME THIS PROBLEM, DURING OUTPUT, THE TERMINAL IS SWITCHED INTO HALF-DUPLEX MODE (ONLY IF IT IS A FULL-DUPLEX TERMINAL) THUS DISABLING CHARACTER REFLECTION OF THE USER INPUT. ANY 'LOST' INPUT IS OUTPUT ALONG WITH THE NEXT PROMPT

AS DESCRIBED ABOVE.

EACH OUTPUT LINE MAY OPTIONALLY BE PREFIXED BY A STANDARD PIECE OF TEXT CONTAINING THE SITE NAME TO WHICH THAT OUTPUT REFERS. IF THE SITE NAME IS 'SIA' THEN THE PREFIX WOULD BE '..SIA - '. OUTPUT THAT IS SITE INDEPENDENT WOULD NORMALLY BE PREFIXED BY '..' .

APART FROM THE OUTPUT ROUTINES THEMSELVES (DESCRIBED BELOW), THE FOLLOWING ROUTINES HELP PROVIDE THE FUNCTIONALITY DESCRIBED ABOVE:-

SPFIX

```
CALL SPFIX(XFIX)
```

THIS ROUTINE SETS THE OUTPUT PREFIX:-
IF XFIX = -1, THEN NO PREFIX IS SET.
IF XFIX = 0 , THEN THE PREFIX IS SET TO '..' .
IF XFIX = 'SITE' , THE PREFIX IS SET TO ..SITE- , WHERE SITE IS AN INTEGER*4 SITE NAME.

PREFIX

```
CALL PREFIX
```

THIS ROUTINE OUTPUTS THE STANDARD PREFIX, SET BY A CALL TO SPFIX.

PROMPT

```
CALL PROMPT
```

THIS ROUTINE OUTPUTS THE STANDARD PROMPT, THE PROMPT CHARACTERS ARE HELD IN THE ARRAY CHPR WHICH IS INDEXED BY THE CURRENT MODE IMODE TO OBTAIN THE CURRENT PROMPT CHARACTER. THE CURRENT 'TOSITE' IS OUTPUT IF ONE HAS BEEN DEFINED AND ANY PARTIAL COMMAND LINE INPUT IS ALSO OUTPUT.

OMODE

```
CALL OMODE(FLAG)
```

THIS ROUTINE WHEN CALLED WITH A FLAG OF VALUE +1 WILL INCREMENT THE LEVEL COUNT OLEVEL, WHEN CALLED WITH A FLAG OF VALUE -1, OLEVEL WILL BE DECREMENTED.

IF THE VALUE OF OLEVEL IS THEN GREATER THAN ONE, CHARACTER REFLECTION WILL BE DISABLED BY THE ROUTINE DUPOUT, IF THE VALUE IS EQUAL TO ZERO THEN CHARACTER REFLECTION IS ENABLED AND A PROMPT IS OUTPUT.

ALL OUTPUT SHOULD BE ENVELOPED BY THE OMODE ROUTINE E.G.

```
CALL OMODE(+1)
CALL FNOU('HELLO',5)
CALL FNOU('_____',5)
CALL OMODE(-1)
```

THIS WILL ENSURE THAT THE OUTPUT IS NOT INTERRUPTED BY COMMAND INPUT (CHARACTER REFLECTION DISABLED), AND THAT A NEW PROMPT IS ONLY OUTPUT AT THE END OF ALL LINES OF THAT OUTPUT.

THE ROUTINES PERR AND PMSG HAVE THEIR OWN INTERNAL CALLS TO OMODE, BUT TO SUPPRESS INTER-LINE PROMPTS MULTIPLE CALLS TO PERR OR PMSG MAY BE

ENVELOPED BY OMODE, THE LEVEL COUNT COPING WITH THE MULTIPLE CALLS TO OMODE.

DUPOUT

```
CALL DUPOUT(+1) /* DISABLE CHARACTER REFLECTION
CALL DUPOUT(-1) /* ENABLE CHARACTER REFLECTION
```

THIS ROUTINE CALLS DUPLX$ TO SWITCH THE TERMINAL BETWEEN FULL AND HALF-DUPLEX MODE. A CALL TO DUPLX$ IN THE ROUTINE INIT AT THE PROGRAM START DETERMINES IF THE TERMINAL IS HALF OR FULL DUPLEX, DUPOUT ONLY FUNCTIONS IF THE TERMINAL IS FULL DUPLEX.

1.7 ERROR MESSAGES

IN ORDER THAT IDENTICAL ERRORS HAVE IDENTICAL ERROR MESSAGES IN THE DIFFERENT EMULATORS, TWO STANDARD MESSAGE OUTPUT ROUTINES ARE AVAILABLE, ALL MESSAGES AND ERRORS SHOULD BE OUTPUT BY THEM.

THE ROUTINES ARE CALLED PMSG AND PERR.

PERR — THIS ROUTINE PRINTS OUT AN EMULATOR STANDARD ERROR MESSAGE. IT IS VERY SIMILAR TO ERRPR$.

THE CALLING SEQUENCE IS:-

```
CALL PERR(ICODE, PREFIX, TEXT, TXTLEN, NAME, NAMLEN)
```

ICODE — ICODE IS A PRIMOS ERROR CODE (E.G., E$EOF), GENERALLY RETURNED FROM A PRIMOS CALL, WITH AN ERROR RETURN CODE, OR IT IS AN EMULATOR ERROR CODE, (E.G., N$BDTK). EMULATOR ERROR CODES DO NOT CLASH WITH THOSE OF PRIMOS. THE STANDARD LIST OF EMULATOR ERROR MESSAGES IS IN RJECOM>ERRORS, AND THE ERROR KEYS IN RJECOM>ERRKEY.

NOTE:
IF ICODE = E$NULL, NO STANDARD MESSAGE IS PRINTED, BUT THE TEXT AND NAME ARRAYS ARE PRINTED.

IF ICODE = 0, NOTHING IS PRINTED.

ICODE IS NOT CLEARED BY THIS ROUTINE.

PREFIX — PREFIX IS AN INTEGER*4 VARIABLE THAT CONTAINS A FOUR CHARACTER TEXT PREFIX FOR THE MESSAGE. ALL ERROR MESSAGES ARE NORMALLY PREFIXED BY '..' TO INDENT THEM, THE TEXT 'PREFIX' IS THEN OUTPUT. IF PREFIX = 0, THEN ONLY THE '..' IS OUTPUT, IF PREFIX = -1, THEN EVEN THIS IS SUPPRESSED. THE PREFIX WILL NORMALLY BE A FOUR CHARACTER SITE NAME.

TEXT — TEXT IS A MESSAGE TO BE PRINTED AFTER THE STANDARD MESSAGE.

TXTLEN — TXTLEN IS THE LENGTH OF THE MESSAGE IN CHARACTERS (IF TEXT AND TXTLEN = 0, THEN NO MESSAGE IS PRINTED).

NAME — NAME IS THE NAME OF THE ROUTINE REPORTING THE MESSAGE. (IT IS AUTOMATICALLY ENCLOSED IN PARENTHESES).

NAMLEN — NAMLEN IS THE LENGTH OF NAME IN CHARACTERS. (IF NAME AND NAMLEN = 0, THEN NO NAME IS PRINTED).

PMSG — THIS ROUTINE PRINTS OUT AN EMULATOR STANDARD INFORMATION MESSAGE. IT IS SIMILAR IN FUNCTION TO PERR.

NOTE THAT AT PRESENT THE ONLY REAL DIFFERENCE BETWEEN THESE TWO ROUTINES IS THAT PERR SETS ERROR FLAGS THAT ABORT A COMINPUT RUN.

THE CALLING SEQUENCE IS:-

```
CALL   PMSG(IMESG, PREFIX, TEXT, TXTLEN, NAME, NUMBER)
```

IMESG — IMESG IS AN EMULATOR MESSAGE CODE (E.G., M$CNFI). THE STANDARD LIST OF MESSAGES IS IN RJECOM>MESGS, AND THE MESSAGE KEYS IN RJECOM>MSGKEY.

NOTE - IF IMESG = 0, THEN NO STANDARD MESSAGE IS PRINTED, BUT THE TEXT AND NAME ARRAYS ARE PRINTED.

PREFIX — PREFIX IS AN INTEGER*4 VARIABLE THAT CONTAINS A FOUR CHARACTER TEXT PREFIX FOR THE MESSAGE. ALL ERROR MESSAGES ARE NORMALLY PREFIXED BY '..' TO INDENT THEM, THE TEXT 'PREFIX' IS THEN OUTPUT. IF PREFIX = 0, THEN ONLY THE '..' IS OUTPUT, IF PREFIX = -1, THEN EVEN THIS IS SUPPRESSED. THE PREFIX WILL NORMALLY BE A FOUR CHARACTER SITE NAME.

TEXT — TEXT IS A MESSAGE TO BE PRINTED AFTER THE STANDARD MESSAGE.

TXTLEN — TXTLEN IS THE LENGTH OF THE MESSAGE IN CHARACTERS. (IF TEXT AND TXTLEN = 0, THEN NO MESSAGE IS PRINTED).

NAME — NAME IS THE NAME OF A SECONDARY PIECE OF TEXT.

NAMLEN — NAMLEN IS THE LENGTH OF NAME IN CHARACTERS. (IF NAME AND NAMLEN = 0, THEN NO NAME IS PRINTED).

1.8 LOW LEVEL TERMINAL OUTPUT ROUTINES

ALL TERMINAL OUTPUT IS DIRECTED THROUGH A NUMBER OF LOW LEVEL OUTPUT ROUTINES. THESE ROUTINES ARE BASED ON THE PRIME GENERIC 'TXOU' ROUTINES, THAT HAVE BEEN 'PACKAGED' FOR THE EMULATOR ENVIRONMENT. THE ROUTINES ARE AS FOLLOWS:-

FNOU — THIS IS A TNOU EQUIVALENT.

FNOUA — THIS IS A TNOUA EQUIVALENT.

FONL	THIS IS A TONL EQUIVALENT.
FODEC	THIS IS A TODEC EQUIVALENT.
FODEC$	THIS ROUTINE IS SIMILAR TO TODEC BUT OUTPUTS A LONG INTEGER. (INTEGER*4)
FODEC3	THIS ROUTINE IS SIMILAR TO TODEC, BUT ONLY OUTPUTS A 3 DIGIT POSITIVE INTEGER WITH LEADING ZEROES.
FOCHAR	THIS ROUTINE IS SIMILAR TO TODEC BUT OUTPUTS A BYTE IN OCTAL FORMAT.
FNOU$, FNOUA$	THESE ROUTINES ARE IDENTICAL IN OPERATION TO FNOU, AND FNOUA RESPECTIVELY. THE CALLING SEQUENCE AVOIDS THE PROGRAMMER HAVING TO COUNT THE CHARACTERS IN THE TEXT STRING. FOR EXAMPLE: CALL FNOU$('$ERROR!$') IS EQUIVALENT TO: CALL FNOU('ERROR',5) THE $ CHARACTER IS THE STRING DELIMITER CHARACTER. ANY CHARACTER MAY BE CHOSEN.
FNOD$, FNO$$, FNOT$	THESE THREE ROUTINES PROVIDE ADDED FUNCTIONALITY OVER THE ROUTINES JUST DESCRIBED, THEY ARE DESCRIBED BY EXAMPLE IN THE CODE.

1.9 FILE UNIT MANAGEMENT

ALL FILE UNITS WITHIN ALL THE EMULATOR CODES ARE ALLOCATED DYNAMICALLY. THIS IS NOT IMPLEMENTED BY THE REV 16 SRCH$$ OPTION BUT BY THREE ROUTINES SPECIFICALLY WRITTEN TO PROVIDE THE SAME FUNCTIONALITY. THE THREE ROUTINES ARE AUNIT, GUNIT, AND DUNIT.

WHENEVER THE SYSTEM REQUIRES A FILE UNIT, THE FUNCTION GUNIT MUST BE CALLED TO OBTAIN A FREE UNIT.

FOR EXAMPLE:

```
FUNIT = GUNIT(ICODE)
```

IF NO UNIT IS AVAILABLE, ICODE WILL BE SET. WHEN THE UNIT HAS BEEN FINISHED WITH, A CALL TO DUNIT WILL CLOSE THE UNIT, AND MAKE IT AVAILABLE FOR FUTURE USE.

FOR EXAMPLE:

CALL DUNIT(FUNIT,ICODE)

THERE ARE ONLY 15 UNITS MADE AVAILABLE, UNITS 1 TO 16, NOT INCLUDING UNIT 6. UNIT 6 IS USED SOLELY FOR COMINPUT. THE ROUTINE AUNIT, CLOSES ALL THESE UNITS AND MAKES THEM AVAILABLE FOR USE.

ANY UNIT (1-16) THAT IS OPEN WHEN THE WORKSTATION OR SEND UTILITY ARE INVOKED IS CLOSED.

1.10 CHARACTER HANDLING

MOST BASIC CHARACTER HANDLING WITHIN THE COMMON USER INTERFACE CODE IS HANDLED BY FIVE LOW LEVEL ROUTINES.

IGET — THIS FUNCTION EXTRACTS A SINGLE CHARACTER FROM AN ARRAY. THE LEFT HAND BYTE OF THE WORD IS ZERO, THE RIGHT HAND BYTE HOLDS THE EXTRACTED CHARACTER.

IPUT — THIS ROUTINE DEPOSITS A SINGLE CHARACTER IN AN ARRAY.

COPY — THIS ROUTINE COPIES CHARACTER STRINGS BETWEEN ARRAYS, THE STRINGS MAY START ON BYTE BOUNDARIES.

ICOMP — THIS FUNCTION COMPARES CHARACTER STRINGS AND RETURNS THE NUMBER OF CHARACTERS MATCHED, THE STRINGS MAY START ON BYTE BOUNDARIES.

EXACT — THIS FUNCTION COMPARES TWO CHARACTER STRINGS FOR EXACT EQUIVALENCE.

THE CHARACTER POINTER ARGUMENTS TO ALL THESE ROUTINES START AT CHARACTER POSITION 1, NOT ZERO. THE ROUTINES, AND THEIR PRECISE CALLING FORMAT ARE BEST UNDERSTOOD BY REFERENCE TO THE CODE.

1.11 CONCAT FILE FORMAT

A CONCAT FILE CONSISTS OF A FILE HEADER, FOLLOWED BY ANY NUMBER OF TEXT OR BINARY SUB-FILES.

THE FILE HEADER IS A SINGLE 16 BIT WORD OF VALUE OCTAL :1002 (TWO BYTES :002:002).

EACH SUB-FILE CONSISTS OF A SUB-FILE HEADER, FOLLOWED BY A DATA BLOCK. THE HEADER IS 3 16 BIT WORDS LONG, THE FIRST WORD IS THE SUB-FILE TYPE

'TEXT' TAKES A VALUE OF 0, AND 'BINARY' A VALUE OF 1. THE NEXT TWO WORDS CONTAIN A LONG INTEGER WHOSE VALUE IS THE LENGTH OF THE SUB-FILE DATA BLOCK IN 16 BIT WORDS.

THE DATA BLOCK IS SIMPLY THE TEXT OR BINARY DATA THAT CONSTITUTES THE

SUB-FILE. THERE IS NO STORED FILE TERMINATOR. NO PADDING OR OTHER PROCESSING IS PERFORMED ON THE SUB-FILES WHEN THEY ARE CREATED.

CONCAT FILE CREATION

THE CREATION OF A CONCAT FILE WITHIN THE WORKSTATION PROGRAM IS A SIMPLE PROCESS GIVEN THE CORRECT FILE FORMAT DESCRIBED ABOVE.

THE FOLLOWING ROUTINES HANDLE THIS PROCESS, THEY ALL COMMUNICATE VIA THE COMMON AREA RJECOM>CONCAT.

SCNCAT

THIS ROUTINE OPENS A TEMPORARY FILE AND WRITES THE FILE HEADER TO IT. IT IS EFFECTIVELY CALLED BY THE 'CONCAT' COMMAND.

TCNCAT

THIS ROUTINE ADDS A TEXT FILE TO THE CURRENTLY OPEN CONCAT FILE, INCLUDING THE HEADER. IT IS EFFECTIVELY CALLED BY THE 'TEXT' COMMAND.

BCNCAT

THIS ROUTINE ADDS A BINARY FILE TO THE CURRENTLY OPEN CONCAT FILE, INCLUDING THE HEADER. IT IS EFFECTIVELY CALLED BY THE 'BINARY' COMMAND.

JCNCAT

THIS ROUTINE ADDS A JCL TEXT (EQUIVALENT TO A TEXT FILE) TO THE CURRENTLY OPEN CONCAT FILE. IT IS EFFECTIVELY CALLED BY THE JCL COMMAND.

XCNCAT

THIS ROUTINE ADDS TEXT TO THE CURRENTLY OPEN CONCAT FILE. THE 'OPMESS' COMMAND MAKES USE OF THE CONCAT ROUTINES TO BUILD A CONCAT FILE FOR TRANSMISSION. THIS ROUTINE ENABLES A SPECIFIC NAMED BUFFER TO BE ADDED TO THE CONCAT FILE.

QCNCAT

THIS ROUTINE 'QUITS' THE CURRENT CONCAT FILE, IT DELETES THE TEMPORARY FILE AND RETURNS TO NORMAL COMMAND LEVEL.

THE ACTUAL CODE TO CLOSE THE CONCAT TEMPORARY FILE AND TRANSMIT IT, IS IN-LINE CODE IN ROUTINE MSPCFC.

1.12 COMMON AREA AND INSERT FILES

THE MAJORITY OF COMMON AREAS ARE HELD AS INSERT FILES, IN RJECOM

A NUMBER OF OTHER COMMON DECLARATIONS ARE ALSO HELD IN INSERT FILES IN THE SAME UFD.

BELOW IS A LIST OF THESE FILES:-

ERRORS

THIS CONTAINS THE LIST OF EMULATOR ERROR MESSAGES.

ERRKEY

THIS CONTAINS THE LIST OF KEYS TO THE EMULATOR ERROR MESSAGES. (ERRORS)

MESGS

THIS CONTAINS THE LIST OF EMULATOR MESSAGES (NOT ERRORS).

MSGKEY

THIS CONTAINS THE LIST OF KEYS TO THE EMULATOR MESSAGES. (MESGS).

CIO

THIS CONTAINS THE EMULATOR INPUT/OUTPUT COMMON BLOCKS, AND TOKEN VARIABLE DECLARATIONS AND EQUIVALENCES.

CLINE

THIS CONTAINS THE CLINE COMMON BLOCK, BEING THE SITE AND SMLC LINE ASSOCIATIONS, AND SEVERAL MISCELLANEOUS VARIABLES.

CONFIG

THIS CONTAINS THE CONFIG COMMON AREA, BEING THE VARIABLES DIMENSIONED AT NLINE THAT DEFINE AN INDIVIDUAL SITE.

CCMD

THIS CONTAINS THE CCMD COMMON AREA, BEING ALL THE DATA PERTAINING TO COMMAND PROCESSING. UNFORTUNATELY, IT CONTAINS A NUMBER OF DATA STATEMENTS, AND SO ITS POSITION AT THE TOP OF A SUBROUTINE IS CRITICAL.

CONCAT

THIS CONTAINS THE CONCAT COMMON AREA, USED BY THE CONCAT ROUTINES TO CREATE A CONCAT FILE.

STATTX

THIS CONTAINS THE STATTX COMMON AREA, THAT DEFINES THE TRANSMIT STATISTICS VARIABLES.

RSTATS

THIS CONTAINS THE STATRC COMMON AREA, THAT DEFINES THE RECEIVE STATISTICS VARIABLES.

PARAM

THIS CONTAINS A NUMBER OF PARAMETRS USED BY THE EMULATORS, IN PARTICULAR THE VALUE OF NLINE (THE MAXIMUM NUMBER OF SMLC LINES) AND SLPTIM (THE SLEEP PERIOD IN THE SUBROUTINE CNTRL) MAY BE MODIFIED BY AN ANALYST TO OPTIMIZE AN EMULATOR FOR A PARTICULAR USER.

SECTION 2

EMSCOM COMMON MODULES

A NUMBER OF COMMON MODULES HAVE BEEN PROVIDED FOR THE EMULATORS FOR OPENING FILES, SENDING SYMBIONT COMMANDS, PERFORMING OPERATIONS ON THE XY FILE ETC. THESE ROUTINES, DESCRIBED BELOW, RESIDE IN UFD RJECOM.

XYUPDT

THIS ROUTINE IS CALLED WHENEVER ANY ACTION (READ/WRITE FROM/TO THE XY FILE) HAS TO BE PERFORMED ON THE XY FILE. THE FORMAT OF THE XY FILE IS AS FOLLOWS (FOR EACH LINE).

```
         SITENAME       2 WORDS (4 ASCII CHARACTERS)
RECEIVE  FILENAME       3 WORDS (6 ASCII CHARACTERS)
TRANSMIT FILENAME       3 WORDS (6 ASCII CHARACTERS)
OPERATOR FILENAME       3 WORDS (6 ASCII CHARACTERS)
ENABLE   WORD           1 WORD  (>L LINE ENABLED)
```

THE XY FILE IS CAPABLE OF HANDLING A MAXIMUM OF FOUR LINES. XYUPDT HAS FOUR PARAMETERS AS FOLLOWS:-

1. KEY - A KEY (RANGE 1-13, DEFINED IN FILE RJECOM>XY$KEY) DEFINING THE ACTION TO BE PERFORMED

2. FILNAM - PARAMETER USED FOR PASSING FILENAMES TO/FROM XYUPDT

3. LINNUM - LINE INDEX NUMBER FOR WHICH THE ACTIONS ARE TO BE PERFORMED

4. ICODE - SET NON-ZERO IF AN ERROR OCCURS (IT IS UP TO THE CALLING ROUTINE TO HANDLE ERROR RECOVERY). ERRORS OCCUR FOR THE FOLLOWING REASONS:-

   I) NO FREE FILE UNITS AVAILABLE

   II) KEY OUTSIDE ALLOWABLE RANGE

   III) FILE SYSTEM ERRORS

   IV) CURRENT SITENAME AND XY FILE SITE NAME NOT THE SAME

   V) LINE ENTRY BEING CHANGED WHILE LINE ENABLED

   VI) XY FILE NOT FREE (10 ATTEMPTS MADE TO ACCESS IT)

SINCE THE SEND UTILITY NEEDS TO ACCESS THE XY FILE, AND SINCE IT MAY BE ATTACHED TO A USERS UFD, AN ATTACH IS MADE TO THE QXXXXX UFD WHENEVER THE SEND UTILITY REQUIRES ACCESS TO THE XY FILE (THE ATTACH IS UNNECESSARY FOR THE WORKSTATION AS IT ALWAYS RUNS IN THE QXXXXX UFD).

TO OVERCOME CLASH CONDITIONS WHEN BOTH THE WORKSTATION AND SEND UTILITY REQUIRE ACCESS TO THE XY FILE, XYUPDT WILL SLEEP FOR 10 PERIODS WAITING FOR THE FILE TO BECOME FREE (AFTER THIS TIME AN ERROR IS GIVEN).

WHEN ACCESS TO THE XY FILE IS GAINED, THE FILE IS READ AND A CHECK MADE TO ENSURE THE CURRENT 'TOSITE' SITE NAME REQUESTING ACTION IS IDENTICAL TO THE CORRESPONDING XY FILE SITENAME FOR THE DESIRED LINE.

SPECIFIC ACTIONS TAKEN FOR THE VARIOUS KEYS IS AS FOLLOWS:-

XY$UXY - IF THE LINE IS NOT ENABLED THE LINE ENTRY FOR THE CURRENT LINE IS CREATED (IF THE ENTRY DOES NOT ALREADY EXIST) AS SPECIFIED ABOVE. IF THE LINE IS ENABLED, AN ERROR REPLY IS GIVEN.

XY$DEL - THE ENTRY FOR THE CURRENT LINE IS DELETED.

XY$ITF - THE SUPPLIED FILENAME IS CHANGED TO THE NEXT AVAILABLE TRANSMIT FILENAME, AND THE TRANSMIT FILENAME INCREMENTED (NOTE XYUPDT IS ONLY CALLED WHEN THE FILE HAS BEEN FULLY CONVERTED TO AVOID BREAKING THE TRANSMIT QUEUE IF THE USER QUITS IN THE MIDDLE OF CONVERTING A FILE).

XY$NRF - RETURNS TO THE CALLER THE NEXT RECEIVE FILENAME TO BE PROCESSED BY THE WORKSTATION.

XY$IRF - INCREMENTS THE RECEIVE FILENAME (THE CURRENT FILE HAVING BEEN COMPLETELY CONVERTED).

XY$NOF - RETURNS TO THE CALLER THE NEXT OPERATOR FILENAME TO BE PROCESSED BY THE WORKSTATION.

XY$IOF - INCREMENTS THE OPERATOR FILENAME (THE CURRENT FILE HAVING BEEN COMPLETELY CONVERTED).

XY$NIF - RETURNS TO THE CALLER THE NEXT TRANSMIT FILENAME TO BE PROCESSED BY THE SEND UTILITY OR WORKSTATION.

XY$ENB - SETS THE ENABLE WORD IN THE XY FILE FOR THE CURRENT LINE.

XY$TFN - REPLACES THE LAST FOUR DIGITS OF THE CURRENT TRANSMIT FILENAME WITH THOSE SUPPLIED WITH TFILE COMMAND (NEW NUMBER HELD IN ARRAY TSEQ).

XY$RFN - REPLACES THE LAST FOUR DIGITS OF THE CURRENT RECEIVE FILENAME WITH THOSE SUPPLIED BY THE RFILE COMMAND (NEW NUMBER HELD IN ARRAY RSEQ).

XY$DIS - UNSETS THE ENABLE WORD IN THE XY FILE FOR THE CURRENT LINE.

XY$SIT - SIMPLY TESTS WHETHER OR NOT THE SITE NAMES MATCH.

ON EXIT, THE XY FILE IS RE-WRITTEN TO DISC IF CHANGES HAVE BEEN MADE, THE FILE IS CLOSED AND THE FILE UNIT DEALLOCATED, AND FOR THE SEND UTILITY ONLY AN ATTACH IS MADE TO THE HOME UFD.

OPNFIL

OPNFIL IS A COLLECTION OF ROUTINES THAT OPEN VARIOUS FILES REQUIRED BY SEVERAL SECTIONS OF THE WORKSTATION AND SEND UTILITY. OPNFIL HAS SIX PARAMETERS, AS FOLLOWS:-

1) KEY - A KEY (IN THE RANGE 1-7, DEFINED IN FILE RJECOM>OP$KEY) THAT DEFINES THE ACTION TO BE TAKEN.

2. FILNAM - USED TO PASS FILENAMES TO/FROM OPNFIL.

3. NAMLEN - LENGTH OF FILNAM IN BYTES.

4. UNIT - FILE UNIT RETURNED TO CALLER IF THE FILE IS SUCCESSFULLY OPENED (SET TO ZERO IF THE FILE CANNOT BE OPENED).

5. LINNUM - INDEX NUMBER FOR THE CURRENT LINE.

6. ICODE - SET NON-ZERO IF AN ERROR OCCURS (IT IS UP TO THE CALLING ROUTINE TO HANDLE ERROR RECOVERY).

ERRORS OCCUR FOR THE FOLLOWING REASONS:-

A) KEY OUTSIDE ALLOWABLE RANGE

B) NO FREE FILE UNITS AVAILABLE

C) FILE SYSTEM ERRORS

D) ERRORS FROM CALL TO XYUPDT

E) ERRORS FROM CALL TO SPOOL$

THE ACTION TAKEN DEPENDS ON THE KEY GIVEN:-

O$FIL - ATTEMPTS TO OPEN THE FILE SPECIFIED BY FILNAM AND NAMLEN. A TSRC$$ CALL IS USED FOR THE SEND UTILITY AND SRCH$$ FOR THE WORKSTATION (THE LATTER IS DONE TO AVOID THE WORKSTATION CRASHING DUE TO BAD PASSWORDS ETC.) THE FILE UNIT IS RETURNED TO THE CALLER.

O$DUM - CREATES A TEMPORARY TRANSMIT FILENAME OF FORMAT TDLXXX, WHERE L IS LINE NUMBER AND XXX IS A THREE DIGIT DECIMAL NUMBER. AN ATTACH IS MADE TO THE QXXXXX UFD IF THE SEND UTILITY IS IN USE. THE TEMPORARY FILENAME IS OPENED, TRUNCATED AND RETURNED TO THE USER TOGETHER WITH THE FILE UNIT. ON EXIT AN ATTACH IS MADE TO THE HOME UFD FOR THE SEND UTILITY.

O$PUN - THE ACTION TO BE TAKEN DEPENDS ON THE VALUE OF THE SITE DEFINITION WORD PUNCH (AS DEFINED)" BY THE PUNCH COMMAND). IF PUNCH HAS A VALUE OF NOTSET (NORMAL) OR ONE (PUNCH FILE WITHOUT TRANSLATION) A DUMMY FILE OF PULXXX IS OPENED, WHERE L = LINE NUMBER, AND XXX IS A THREE DIGIT DECIMAL NUMBER. FOR PUNCH OF VALUE TWO (PUNCH FILE WITH TRANSLATION) A FILE OF PTLXXX IS OPENED. BOTH THESE FILES RESIDE IN QXXXXX UFD. IF PUNCH HAS A VALUE OF THREE (PUNCH FILE, TRANSLATE AND SPOOL) A SPOOL FILE

IS OPENED WITH THE SAME ATTRIBUTES AS FOR THE PRINTER, EXCEPT THE BANNER NAME IS SET TO YYYY:PUNCH:RRRRRR, WHERE YYYY IS THE SITE NAME AND RRRRRR IS THE CURRENT RECEIVE FILENAME. IN EACH CASE THE FILENAME AND FILE UNIT ARE RETURNED TO THE CALLER.

O$REC - ATTEMPTS TO OPEN THE NEXT RECEIVE FILE, AS SPECIFIED IN THE XY FILE. IF THE OPEN IS SUCCESSFUL, THE CALLER IS GIVEN THE FILENAME AND FILE UNIT, OTHERWISE THE FILE UNIT IS RETURNED TO THE FREE POOL AND THE USER IS GIVEN THE FILE SYSTEM ERROR CODE.

O$ZOP - ATTEMPTS TO OPEN THE 'ZOP' FILE FOR SENDING AN OPERATOR MESSAGE TO THE SYMBIONT. IF THE FILE EXISTS THE USER IS GIVEN THE FILE SYSTEM ERROR CODE AND THE FILE UNIT IS DEALLOCATED, ELSE THE FILE UNIT IS PASSED TO THE CALLER.

O$PRN - OPENS A FILE IN THE SPOOL QUEUE WITH THE NAME YYYY:PRINT:RRRRRR WHERE YYYY IS THE SITENAME AND RRRRRR IS THE CURRENT RECEIVE FILENAME. THE FILE HAS THE ATTRIBUTES OF A BANNER PAGE AND NO FORMATTING AND A FORM TYPE, IF ONE HAS BEEN SPECIFIED (BY COMMAND PRFORM).

O$OPS - ATTEMPTS TO OPEN THE NEXT RECEIVE OPERATOR FILES AS DEFINED BY THE FILENAME IN THE XY FILE. RETURNS WITH THE FILENAME AND FILE UNIT IF THE OPEN IS SUCCESSFUL, ELSE RETURNS THE FILE UNIT TO THE FREE POOL AND RETURNS WITH THE FILE SYSTEM ERROR CODE.

ON EXIT AN ATTACH IS MADE TO THE HOME UFD IF THE SEND UTILITY IS RUNNING OR IF AN ERROR HAS OCCURED IN OPENING THE SPOOL FILE.

STEPNM

THIS ROUTINE IS USED TO INCREMENT THE FOUR DECIMAL DIGITS OF THE RECEIVE, TRANSMIT AND OPERATOR FILENAMES (FILENAME FORMAT IS ZLXXXX WHERE Z = R,T OR O, L = LINE NUMBER AND XXXX A FOUR DIGIT DECIMAL NUMBER). OVERFLOW TO 0001 OCCURS WHEN THE DECIMAL NUMBER REACHES 9999.

DECARR

THIS ROUTINE CONVERTS A SINGLE PRECISION BINARY NUMBER TO A FIVE DECIMAL DIGIT ASCII CHARACTER STRING. IT IS USED FOR CHANGING THE TRANSMIT, RECEIVE AND OPERATOR FILENAMES. IT HAS THREE PARAMETERS AS FOLLOWS:-

1) KEY - ZERO SUPPRESSES LEADING BLANKS, NON ZERO NO BLANK SUPPRESSION.

2) NUMBR - NUMBER TO BE CONVERTED.

3) ARRAY - 3 WORD ARRAY TO HOLD THE CONVERTED NUMBER.

THE FIRST BYTE OF THE WORD INDICATES WHETHER THE NUMBER IS POSITIVE (BYTE = BLANK CHARACTER) OR NEGATIVE (BYTE = MINUS CHARACTER). THE NUMBER IS UNPACKED INTO THE REMAINING FIVE BYTES.

ZWRITE

ZWRITE IS CALLED WHENEVER A COMMAND IS TO BE SENT TO THE SYMBIONT. SOME OF THE COMMANDS ARE COMMON TO ALL EMULATORS (E.G. ENABLE) WHILE OTHERS RELATE TO SPECIFIC EMULATORS (E.G. VOICE FOR THE 1004). ZWRITE HAS FOUR PARAMETERS AS FOLLOWS:-

1) NKEY - A KEY (RANGE 1-18 AS SPECIFIED IN FILE RJECOM>ZF$KEY) DEFINING THE ACTION TO BE TAKEN.

2) BUFF - SUPPLIES ADDITIONAL INFORMATION REQUIRED FOR THE COMMAND, AND RETURNS INFORMATION TO THE CALLER.

3) LINNUM - LINE INDEX NUMBER FOR WHICH THE ACTIONS ARE TO BE PERFORMED.

4) ICODE - SET NON ZERO IF AN ERROR OCCURS (IT IS UP TO THE CALLING ROUTINE TO HANDLE ERROR RECOVERY). ERRORS MAY OCCUR FOR THE FOLLOWING REASONS:-

A) KEY OUTSIDE ALLOWABLE RANGE

B) NO FREE FILE UNITS AVAILABLE

C) FILE SYSTEM ERRORS

D) "SYMBIONT COMMAND IGNORED" CAUSED BY THE 'Z' COMMAND FILE NOT BEING DELETED, DUE TO A SYMBIONT CRASH OR SYMBIONT LOOP.

E) ERROR IN ROUTINE XYUPDT

F) ERROR IN ROUTINE RTNAME

ON ENTRY AN ATTEMPT IS MADE TO OPEN THE 'Z' FILE. IF THE FILE EXISTS THE PROGRAM SLEEPS FOR 30 PERIODS OF 0.5 SECONDS WAITING FOR THE 'Z' FILE TO DISAPPEAR. IF THE FILE STILL EXISTS AFTER THIS PERIOD OF TIME THE 'Z' FILE IS DELETED AND AN ERROR RETURN MADE. OTHERWISE A TEMPORARY FILE (CALLED 'ZZZ' TO PREVENT THE SYMBIONT FROM PREMATURELY ACCESSING IT) IS OPENED. THE FIRST THREE WORDS OF A TEMPORARY BUFFER ARE SET TO THE PASSWORD (ALWAYS ZERO), THE PHYSICAL LINE NUMBER AND THE KEY. THE FOLLOWING KEYS SEND ONLY THE ABOVE INFORMATION IN THE 'Z' FILE:-

ZF$KIL - ABORT THE CURRENT TRANSMIT FILE (7020 AND 200UT ONLY).

ZF$OPS - AN OPERATOR MESSAGE IS AVAILABLE FOR TRANSMISSION IN FILE 'ZOP'.

ZF$DIS - DISABLE THE LINE.

ZF$LIS - START MONITORING THE LINE (HASP ONLY).

ZF$DMP - RELEASE ANY RECEIVED DATA FOR THE WORKSTATION TO PROCESS (HASP

ONLY).

ZF$APR - ABORT THE CURRENT PRINT FILE (1004 ONLY).

ZF$APU - ABORT THE CURRENT PUNCH FILE (1004 ONLY).

ZF$OFL - INFORM HOST RJE STATION IS GOING OFFLINE (1004 ONLY)

ZF$HLT - SEND A HALT MESSAGE TO THE HOST (1004 ONLY).

ZF$HGV - SEND A HALT-GO-VOICE MESSAGE TO THE HOST (1004 ONLY).

ZF$RDY - SEND A GO (READY) MESSAGE TO THE HOST (1004 ONLY).

ZF$PAS - INFORM, THE SYMBIONT A PASSWORD CARD IS AVAILABLE IN FILE 'ZOP' (GRTS ONLY).

THE FOLLOWING KEYS REQUIRE ADDITIONAL INFORMATION TO BE SENT TO THE SYMBIONT IN THE 'Z' FILE.

ZF$ENB - ENABLE THE LINE. FOUR WORDS OF ADDITIONAL INFORMATION (DEPENDING ON THE SPECIFIC EMULATOR) IS SENT, SUCH AS WHETHER THE LINE IS LEASED OR SWITCHED, THE OPERATOR FILENAME, ETC (SEE SPECIFIC EMULATOR ENABLE MODULE FOR EXPLICIT DETAILS).

ZF$REC - SENDS THE RECEIVE FILENAME TO BE USED BY THE SYMBIONT. THE FILENAME SENT WILL BE THAT SPECIFIED BY THE RFILE COMMAND, OR THAT DETERMINED BY SUBROUTINE RTNAME (SEE BELOW FOR DETAILS).

ZF$TRN - SENDS THE TRANSMIT FILENAME TO BE USED BY THE SYMBIONT. THE FILENAME SENT WILL BE THAT SPECIFIED BY THE TFILE COMMAND, OR THAT DETERMINED BY SUBROUTINE RTNAME (SEE BELOW FOR RTNAME DETAILS).

ZF$BPS - SENDS THE LINE BAUD RATE AS SPECIFIED BY THE SITE DEFINITION COMMAND BAUD.

ZF$SON - SEND THE SIGNON CARD (HASP ONLY). A BLOCK COUNT (SIZE OF DATA BLOCK) IS ADDED, TOGETHER WITH THE BCB-FCS-FCS-RCB-SRCB HEADER SEQUENCE. THE SIGNON CARD IS COPIED TO THE TEMPORARY BUFFER WHERE IT IS TRANSLATED TO EBODIC (IT IS NOT COMPRESSED USING THE MULTI-LEAVING TECHNIQUES). A TRAILING RCB IS ADDED.

ZF$SOF - SEND THE SIGNOFF CARD (HASP ONLY). THIS IS DONE IN THE SAME MANNER AS FOR THE SIGNON CARD.

IN ALL CASES THE TEMPORARY BUFFER IS WRITTEN TO THE 'ZZZ' FILE. THE FILENAME IS THEN CHANGED TO 'Z' AND THE WAIT ROUTINE RE-ENTERED. IF THE Z FILE DISAPPEARS WITHIN 15 SECONDS, RETURN IS IMMEDIATE TO THE CALLING ROUTINE, OTHERWISE AN ERROR INDICATION IS RETURNED.

DISABLE

DSABLE IS CALLED WHENEVER THE DISABLE OR RECOVER COMMANDS ARE GIVEN. THERE ARE TWO PARAMETERS TO THE CALL AS FOLLOWS:-

1. LINENX - THE LINE INDEX NUMBER
2. ICODE - CURRENTLY NOT USED

IF THE SYMBIONT FILE LOCK IS UNSET (FILE NOT THERE OR DELETABLE), XYUPDT IS CALLED TO DISABLE THE LINE, THE INTERNAL WORKSTATION ENABLE FLAG IS UNSET AND THE OPERATOR IS INFORMED.

IF THE SYMBIONT LOCK IS SET, ZWRITE IS CALLED TO SEND THE SYMBIONT DISABLE COMMAND. IF THE SYMBIONT LOCK IS NOW UNSET THE 'SYMBIONT CRASHED' FLAG IS UNSET, THE INTERNAL WORKSTATION ENABLE FLAG IS UNSET AND THE OPERATOR INFORMED.

THE OPERATOR IS INFORMED OF ANY ERRORS CAUSED BY:-

A) AN ERROR IN XYUPDT

B) AN ERROR IN ZWRITE

C) A FILE SYSTEM ERROR

RTNAME

RTNAME IS CALLED BY ZWRITE TO DECIDE THE CORRECT RECEIVE OR TRANSMIT FILENAME THAT SHOULD BE SENT TO THE SYMBIONT (IF THE RFILE OR TFILE COMMANDS HAVE NOT BEEN GIVEN). THERE ARE TWO PARAMETERS:-

1) NAME - THE CURRENT XY FILE RECEIVE OR TRANSMIT FILENAME.

2) ICODE - SET NON ZERO IF AN ERROR OCCURS (THE CALLING ROUTINE MUST INITIATE ERROR RECOVERY PROCEDURE). ERRORS MAY OCCUR FOR THE FOLLOWING REASONS:-

A) NO FREE FILE UNITS AVAILABLE

B) FILE SYSTEM ERRORS

THE CURRENT QXXXXX UFD IS OPENED AND THE ENTRIES READ ONE BY ONE. ANY FILENAMES OF LENGTH SIX CHARACTERS AND WITH THE SAME FIRST TWO CHARACTERS AS THE SUPPLIED NAME ARE CHECKED TO SEE IF THE LAST FOUR DIGITS ARE NUMERIC. NON NUMERIC FILES ARE IGNORED. THE DIGITS ARE CONVERTED TO BINARY AND THE CURRENT LOW AND HIGH VALUES (REPRESENTING PREVIOUSLY FOUND FILENAMES) ARE CHANGED DEPENDING ON WHETHER THE CURRENT FILENAME IS OUTSIDE THE RANGE. WHEN ALL FILES HAVE BEEN EXAMINED THE FOLLOWING SELECTION PROCESS TAKES PLACE:-

1) IF NO NEW FILES HAVE BEEN FOUND IN THE UFD THE CURRENT XY FILENAME IS USED.

2) FOR A RECEIVE FILENAME, IF THE XY FILENAME IS GREATER THAN THE HIGHEST UFD FILENAME, THE XY FILENAME IS USED, OTHERWISE THE HIGHEST UFD FILENAME IS INCREMENTED BY ONE AND THIS FILENAME IS USED.

3) FOR A TRANSMIT FILENAME, IF THE XY FILENAME IS LESS THAN THE

LOWEST UFD FILENAME, THE XY FILENAME IS USED, OTHERWISE THE LOWEST UFD FILENAME IS USED.

THE FILENAME TO BE USED IN THE SYMBIONT COMMAND IS RETURNED AS THE NAME PARAMETER.

QSCAN

THIS ROUTINE IS CALLED FOR THE LQUEUE AND RECOVER COMMANDS TO EXAMINE AND OUTPUT THE STATE OF THE RECEIVE OR TRANSMIT FILE QUEUE.

THERE ARE SEVEN PARAMETERS TO THE CALL (THREE SET BY THE CALLING ROUTINE AND FOUR RETURNED BY QSCAN) AS FOLLOWS:

1) RORT — THE 6 CHARACTER NAME OF THE CURRENT R OR T FILE HELD BY THE WORKSTATION, I.E. THE WORKSTATION POINTER. ALSO SERVES TO TELL QSCAN WHETHER TO EXAMINE THE RECEIVE OR TRANSMIT QUEUE.

2) FILL — SET TO .TRUE. BY CALLER IF QSCAN IS TO FILL 'SINGLE FILE' GAPS FOUND IN THE QUEUE.

3) SITE — THE NAME OF THE SITE FOR WHICH THE QUEUE IS TO BE SCANNED. (4 CHARACTERS, IN A LONG INTEGER).

.....

4) HIGH — THE NUMBER OF THE HIGHEST NUMBERED FILE IN THE QUEUE (IN BINARY, NOT CHARACTERS).

5) NFILES — THE NUMBER OF FILES FOUND IN THE QUEUE.

6) FILENO — THE NUMBER IN BINARY (I.E. NOT CHARACTERS) OF THE WORKSTATION POINTER, I.E. THE FILE NAME IN RORT.

7) ICODE — ERROR INDICATION (0 IF NO ERRORS FOUND). IF AN ERROR IS FOUND (EITHER A FILE SYSTEM ERROR OR AN ERROR IN OBTAINING A FILE UNIT) NO ERROR MESSAGE IS OUTPUT, BUT THE ERROR CODE IS RETURNED TO THE CALLING ROUTINE.

THE ROUTINE STARTS BY OBTAINING A FILE UNIT AND THEN IF IT IS RUNNING AS THE 'SEND UTILITY' (I.E. NOT A FULL WORKSTATION) IT ATTACHES TO THE QUEUE UFD FOR THE DURATION OF THE ROUTINE. (THE WORKSTATION ALWAYS RUNS IN THE QUEUE UFD, SO NO 'ATTACH' IS REQUIRED).

THE QUEUE TITLE IS OUTPUT, AND THEN THE REAL WORK BEGINS. FIRST, THE ROUTINE EXAMINES EVERY ENTRY IN THE UFD TO FIND THE NUMBER OF FILES IN THE QUEUE, AND (IF THAT IS NOT ZERO) TO RECORD THE LOWEST AND HIGHEST NUMBERED FILES FOUND.

HAVING ESTABLISHED THE UPPER AND LOWER LIMITS OF THE QUEUE, THE ROUTINE THEN PROCEEDS TO CHECK FOR MISSING FILES, TO A MAXIMUM OF 100.

NOTE THAT QSCAN DOES NOT MAKE ALLOWANCE FOR 'WRAPAROUND' WHEN LOOKING FOR MISSING FILES. THUS, FOR EXAMPLE, IF THE QUEUE IS MADE UP OF FOUR

FILES, NUMBERED 9998, 9999, 0001, AND 0002, QSCAN WILL FIND THAT FILES 0003 TO 0102 ARE 'MISSING' AND GIVE UP LOOKING AFTER RECORDING 100 FILES ARE 'GONE'.

HAVING RECORDED THE NUMBERS OF THE MISSING FILES IN ARRAY 'GONE', THE ROUTINE GOES THROUGH THAT ARRAY, OUTPUTTING THE NUMBERS OF THE MISSING FILES. IF A 'SINGLE FILE GAP' IS FOUND AND THE ROUTINE WAS ENTERED WITH FILL= .TRUE., THEN QSCAN WILL CREATE A 'NULL' (EMPTY) FILE TO FILL THAT GAP. A MAXIMUM OF TWO 'SINGLE FILE' GAPS ARE FILLED ON ANY CALL TO QSCAN. (TWO WAS DESIGNATED AS A REASONABLE LIMIT, IF MORE 'SINGLE FILE' GAPS EXIST THERE IS PROBABLY SOMETHING DRASTICALLY WRONG WITH THE QUEUE, REQUIRING OPERATOR ATTENTION).

HAVING OUTPUT THE DETAILS OF THE QUEUE AND ITS GAPS, THE FINAL CHECK IS TO SEE WHETHER THE WORKSTATION POINTER IS CONSISTENT WITH THE FILES IN THE QUEUE. AGAIN, 'WRAPAROUND' IS NOT CATERED FOR. THUS THE CHECK FOR A 'RECEIVE' QUEUE IS THAT THE WORKSTATION POINTER IS LESS THAN OR EQUAL TO THE LOWEST NUMBERED FILE FOUND. THE CHECK FOR A TRANSMIT QUEUE IS THAT THE WORKSTATION POINTER IS GREATER THAN THE HIGHEST FILE NUMBER FOUND. A WARNING MESSAGE IS OUTPUT IF THIS CHECK FAILS - BUT NO ERROR CODE IS SET.

FINALLY, A 'SEND UTILITY' ATTACHES BACK TO ITS HOME DIRECTORY AND THEN THE RETURN IS MADE TO CALLING ROUTINE.

QLIST

THIS ROUTINE IS CALLED FOR THE LQUEUE COMMAND, AND USES THE QSCAN ROUTINE TO SCAN FIRST THE RECEIVE QUEUE AND THEN THE TRANSMIT QUEUE. THERE ARE NO PARAMETERS TO SUBROUTINE QLIST.

THE ROUTINE BEGINS BY GETTING THE NAME OF THE SITE FOR WHICH THE QUEUES ARE TO BE LISTED. (GSITE IS USED TO GET THE SITE NAME, NAMELY THE PARAMETER FROM THE LQUEUE COMMAND IF ONE WAS GIVEN, OR IF NOT, THE NAME OF THE CURRENT 'TO' SITE).

FROM THE SITE NAME, ROUTINE GLINE IS USED TO GET THE LINE INDEX FOR THAT SITE, AND THAT LINE INDEX IS SUPPLIED TO ROUTINE XYUPDT TO FIND THE VALUE OF THE WORKSTATION'S RECEIVE FILE POINTER.

THE VALUE (THE 6 CHARACTER NAME OF THE NEXT RECEIVE FILE TO BE PROCESSED BY THE WORKSTATION) IS THE FIRST PARAMETER SUPPLIED TO THE QSCAN ROUTINE. THE NEXT PARAMETER IS 'FILL' WHICH IS SET TO .FALSE., AS NO GAPS IN THE QUEUE ARE TO BE FILLED FOR THE LQUEUE COMMAND. THE THIRD PARAMETER TO QSCAN IS THE SITE NAME. QSCAN IS THEN CALLED TO PRINT THE STATE OF THE RECEIVE QUEUE. THREE OF THE FOUR PARAMETERS RETURNED FROM QSCAN (I.E. HIGH, NFILES AND FILENO) ARE NOT USED BY QLIST. THE LAST PARAMETER - ICODE - IS CHECKED FOR ERRORS AND IF NON-ZERO, THE ERROR MESSAGE IS PRINTED AND CONTROL RETURNED TO THE CALLING ROUTINE.

ON SUCCESSFUL RETURN FROM QSCAN, XYUPDT IS CALLED AGAIN, THIS TIME TO GET THE WORKSTATION'S TRANSMIT FILE POINTER. THIS POINTER IS THEN USED AS THE FIRST PARAMETER IN THE SECOND CALL TO QSCAN, WHICH PRINTS THE STATE OF THE TRANSMIT QUEUE. IF THE ERROR INDICATOR - ICODE - IS

RETURNED AS NON-ZERO, THE ERROR MESSAGE IS PRINTED BY QLIST.

CONTROL IS THEN RETURNED TO THE CALLING ROUTINE.

RCOVER

THIS SUBROUTINE IS CALLED FROM THE WORKSTATION FOR THE RECOVER COMMAND. THERE IS ONE PARAMETER SUPPLIED TO RCOVER, NAMELY:-

1. LINENX THE INDEX OF THE LINE FOR WHICH 'RECOVERY' IS TO BE ATTEMPTED.

RCOVER BEGINS BY DISABLING THE SPECIFIED LINE, USING SUBROUTINE DSABLE. THIS IS REGARDED AS SUCCESSFUL EVEN IF THE LINE WAS NOT PREVIOUSLY ENABLED,.

THE SITE NAME IS THEN OBTAINED FOR LATER USE BY QSCAN.

ROUTINE XYUPDT IS CALLED TO OBTAIN THE WORKSTATION'S RECEIVE FILE POINTER, THEN THIS POINTER IS SUPPLIED AS A PARAMETER TO QSCAN WHICH PRINTS THE STATE OF THE RECEIVE QUEUE. THE FILL PARAMETER IS SUPPLIED TO QSCAN AS .TRUE. SO THAT UP TO TWO GAPS OF ONE FILE IN THE RECEIVER QUEUE WILL BE FILLED BY QSCAN CREATING A NULL FILE. (THE THEORY HERE IS THAT IF THERE ARE MORE SINGLE FILE GAPS THAN TWO, THEN THE QUEUE MUST REALLY BE IN A MESS AND IT SHOULD BE SORTED OUT MANUALLY).

RCOVER THEN CHECKS WHETHER AN R_DUMY OR P_DUMY FILE WAS LEFT BY THE SYMBIONT, AND IF SO, RENAMES IT TO R_NNNN, WHERE NNNN IS ONE MORE THAN THE NUMBER OF THE HIGHEST R FILES FOUND BY QSCAN. HAVING DONE THE RE-NAME, IT CALLS ZWRITE TO UPDATE THE R FILE POINTER FOR THE SYMBIONT WHEN IT IS RE-ENABLED.

NEXT, RCOVER CALLS XYUPDT TO GET THE WORKSTATION'S TRANSMIT FILE POINTER, THEN CALLS QSCAN TO PRINT THE STATE OF THE TRANSMIT QUEUE. AS BEFORE, THE 'FILL' PARAMETER IS SUPPLIED TO QSCAN .TRUE. SO THAT UP TO TWO SINGLE FILE GAPS ARE FILLED BY QSCAN CREATING NULL FILES.

HAVING RETURNED FROM QSCAN, RCOVER THEN CALLS ENABLE TO ATTEMPT TO ENABLE THE SPECIFIED LINE.

THE VARIABLE 'ICODE' IS USED THROUGHOUT AS THE ERROR INDICATION. IF IT IS SET NON-ZERO BY ANY OF THE ROUTINES CALLED BY RCOVER, THEN THE ERROR MESSAGE IS PRINTED AND RCOVER ABANDONS ITS ATTEMPTS AT RECOVERING THE LINE.

SECTION 3

EMULATOR SPECIFIC MODULES

EACH EMULATOR HAS A NUMBER OF MODULES, WHICH ALTHOUGH CONTAIN THE SAME BASIC CODE, ALSO CONTAIN CODE SPECIFIC TO A PARTICULAR EMULATOR. THE FUNCTION OF THE MODULES IS DISCUSSED BELOW.

ENABLE

THE ENABLE MODULE, CALLED WHENEVER THE ENABLE OR RECOVER COMMANDS ARE ISSUED, HAS THE PRIMARY TASK OF SENDING SUFFICIENT INFORMATION TO THE SYMBIONT TO ENABLE A LINE. THE MODULE IS CALLED WITH TWO PARAMETERS.

1) LINENX - THE LINE INDEX NUMBER
2) ICODE - ERROR RETURN CODE (NOT CURRENTLY USED AS ALL ERRORS ARE HANDLED WITHIN THE MODULE).

ON ENTRY TO THE MODULE, PROMPTS ARE INHIBITED TO MAKE THE TRACE OF THE MESSAGES BEING OUTPUT TO THE OPERATOR MORE LEGIBLE. XYUPDT IS CALLED TO CREATE AN ENTRY FOR THE CURRENT SITE (NOTE IF THE CURRENT SITE IS ALREADY SPECIFIED FOR THE CURRENT LINE IN THE XY FILE, THE XY FILE ENTRY IS NOT CHANGED).

THE ENABLE MODULE DETERMINES WHETHER A SYMBIONT ALREADY EXISTS BY TRYING TO DELETE THE SYMBIONT FILE LOCK (SYMLOCK). IF THE FILE DOES NOT EXIST, OR WAS DELETED OKAY, THE SYMBIONT IS NOT RUNNING. IN THIS CASE THE SYMBIONT IS AUTOMATICALLY PHANTOMED, AND A FLAG (RSYMB) IS SET TO INFORM THE MAIN CONTROL LOOP THE SYMBIONT IS RUNNING (WHEN RSYMB IS SET, AND THE MAIN IS ABLE TO DELETE THE FILE LOCK, THE OPERATOR IS INFORMED THE SYMBIONT HAS CRASHED). IN EITHER CASE THE OPERATOR IS INFORMED THE SYMBIONT IS ALREADY RUNNING OR HAS BEEN PHANTOMED.

A NUMBER OF COMMANDS (WHICH COMMANDS ARE SENT DEPENDS ON THE EMULATOR) ARE THEN SENT, SUCH AS:-

A) RECEIVE FILENAME TO BE USED
B) TRANSMIT FILENAME TO BE USED
C) BAUD RATE
D) ENABLE THE LINE
E) LISTEN TO THE LINE
F) SEND SIGNON MESSAGE (IF SIGNON MESSAGE SPECIFIED).

THE INFORMATION SENT WITH THE ENABLE COMMAND DEPENDS ON THE SYMBIONT, SO CHECK INDIVIDUAL MODULES FOR SPECIFIC DETAILS.

THE OPERATOR IS GIVEN A RUNNING COMMENTARY AS EACH COMMAND IS ISSUED, WITH ADDITIONAL INFORMATION SUCH AS THE RECEIVE FILENAME, THE BAUD RATE USED, AND THE DATE AND TIME THE LINE WAS ENABLED. FOLLOWING THE ENABLE COMMAND THE ENABLE WORD IN THE XY FILE IS SET, AND THE WORKSTATION INTERNAL ENABLE FLAG IS SET FOR THE CURRENT 'TOSITE'.

ERRORS MAYS ARISE FOR THE FOLLOWING REASONS:-

1) FILE SYSTEM ERRORS

2) ERRORS FROM ROUTINE ZWRITE WHEN ISSUING THE COMMANDS TO THE SYMBIONT (THE OPERATOR IS INFORMED WHICH COMMAND FAILED).

3) ERRORS FROM XYUPDT WHEN TRYING TO SET PARAMETERS FOR THE CURRENT LINE.

DIAGNS

THE DIAGNS MODULE, CALLED FROM THE MAIN CONTROL LOOP, HAS THE TASK OF DISPLAYING SYMBIONT ERROR MESSAGES ON THE OPERATOR CONSOLE. DIAGNS HAS NO PARAMETERS.

THE SYMBIONT ADDS DIAGNOSTIC MESSAGES TO ONE OR OTHER OF TWO FILES CALLED DIAG1 OR DIAG2 (BOTH FILES ARE REQUIRED, SINCE THE WORKSTATION MAY HAVE THE FORMER OPEN, WHEN THE SYMBIONT WISHES TO ADD A DIAGNOSTIC MESSAGE TO THE FILE).

THE FORMAT OF EACH ENTRY IN THE FILE IS AS FOLLOWS:-

WORD 1 LINE NUMBER AS 'L' WHERE L IS IN ASCII FORMAT.

WORD 2 ERROR CODE AS 2 ASCII CHARACTERS.

WORD 3 ERROR NUMBER (NOT USED).

ON ENTRY AN ATTEMPT IS MADE TO OPEN EITHER OF THE DIAGNOSTIC FILES. IF NEITHER EXIST CONTROL RETURNS TO THE CALLING ROUTINE. OTHERWISE THE FILE IS READ AN ENTRY AT A TIME. IF THE ERROR CODE RECEIVED DOES NOT MATCH ANY CODE IN A LIST HELD IN INSERT FILE DIAGMS, THE ERROR IS IGNORED. OTHERWISE A CHECK IS MADE TO SEE WHETHER THE LINE NUMBER IN THE ENTRY IS CURRENTLY ACTIVE. IF THE LINE IS ACTIVE, THE RELEVANT ERROR MESSAGE HELD IN DIAGMS IS DISPLAYED FOR THE OPERATOR. THE NEXT ENTRY IS THEN READ. WHEN ALL ENTRIES HAVE BEEN DISPLAYED (OR EXAMINED) THE CURRENT DIAGNOSTIC FILE IS CLOSED AND DELETED, AND CONTROL RETURNS TO THE START OF THE ROUTINE.

ERRORS THAT MAY OCCUR ARE:-

I) NO FREE FILE UNITS AVAILABLE

II) FILE SYSTEM ERRORS HANDLING THE DIAGNOSTIC FILES.

THE HASP VERSION OF DIAGNS HAS ANOTHER FEATURE NOT FOUND IN THE OTHER EMULATOR VERSIONS. FIRSTLY, SINCE MANY HASP SYMBIONT DIAGNOSTIC MESSAGES DO NOT INDICATE ERROR CONDITIONS, ONLY THOSE ERROR CODES THAT MATCH A FURTHER MASTER LIST HELD IN DIAGMS WILL CAUSE AN ERROR MESSAGE TO BE DISPLAYED. IF THE MATCH FAILS THE ERROR MESSAGE IS WRITTEN TO THE LOG ONLY. THE OPERATOR MAY CAUSE ALL ERROR MESSAGES TO BE DISPLAYED BY ISSUING THE DEBUG COMMAND WITH PARAMETER ON.

ROPMSG

ROPMSG, CALLED FROM THE MAIN CONTROL LOOP WHEN THE CURRENT LINE IS ENABLED, HAS THE TASK OF DISPLAYING OPERATOR MESSAGES RECEIVED FROM THE HOST SITE. AS THE 1004 AND 2780 EMULATORS DO NOT HAVE OPERATOR DISPLAY FACILITIES, AND HASP OPERATOR MESSAGES ARE INTERLEAVED WITH THE DATA BLOCKS (THE MESSAGES ARE DISPLAYED FROM THE TRANSMIT MODULE), THESE EMULATORS ONLY HAVE A DUMMY ROPMSG MODULE FOR COMPATABILITY WITH THE OTHER EMULATORS (BUT SEE BELOW FOR HASP). ROPMSG DOES NOT HAVE ANY PARAMETERS.

ALTHOUGH THE INPUT BLOCKS DIFFER CONSIDERABLY (DIFFERENT CONTROL CODES, DIFFERENT LINE CODES, ETC.) THE 2OOUT, 7020 AND GRTS ROPMSG MODULES FOLLOW THE SAME METHOD OF GENERATING THE OUTPUT OPERATOR MESSAGE. ON ENTRY RETURN IS IMMEDIATE IF EITHER THE CURRENT SITE IS NOT PROCESSING RECEIVE DATA, OR THE NEXT OPERATOR FILE DOES NOT EXIST. ON OPENING A FILE (DONE BY A CALL TO OPNFIL) PROCESSING CHARACTERS ARE SET UP, AND THE BUFFER SEARCHED FOR LINE TERMINATING CHARACTERS. NON LINE TERMINATING CHARACTERS ARE TRANSLATED (IF NECESSARY) AND ADDED TO THE OUTPUT BUFFER. THE LINE IS DISPLAYED WHEN A LINE, OR BLOCK TERMINATING CHARACTER IS RECEIVED. ONLY THE 2OOUT PROTOCOL ALLOWS MULTIPLE LINE OF TEXT PER BLOCK, SO AFTER DISPLAYING ONE LINE A SEARCH IS MADE FOR THE NEXT LINE. WHEN THE BUFFER HAS BEEN COMPLETELY CONVERTED THE OPERATOR FILE IS CLOSED, DELETED, AND THE OPERATOR FILENAME IN THE XXY FILE INCREMENTED BY A CALL TO XYUPDT.

ERROR CONDITIONS MAY ARISE DUE TO:-

I) ILLEGAL CHARACTERS IN THE RECEIVED MESSAGE

II) FILE SYSTEM ERRORS

III) ERRORS FROM XYUPDT

FURTHER INFORMATION CAN BE GAINED BY LOOKING AT THE CODE FOR THE INDIVIDUAL ROPMSG MODULES.

THE HASP SYMBIONT REQUIRES A 'KICK' EVERY SO OFTEN SO THAT IT MAY RELEASE RECEIVED DATA FOR PROCESSING BY THE WORKSTATION. THE WORKSTATION CODE TO SEND THIS COMMAND IS TO BE FOUND IN ROPMSG.

OPSMSG

OPSMSG IS CALLED WHENEVER THE OPERATOR WISHES TO SEND A MESSAGE TO THE REMOTE SITE (NOTE THAT HASP OPERATOR MESSAGES ARE SENT IN THE DATA STREAM). THE 7020, 2OOUT AND GRTS EMULATORS ONLY, HAVE OPSMSG ROUTINES. ALTHOUGH THE OUTPUT MESSAGE FORMAT AND LINE CODE DIFFERS BETWEEN THE THREE EMULATORS THE IMPLEMENTATION METHOD OF EACH EMULATOR IS SIMILAR.

OPSMSG IS CALLED WITH THE FOLLOWING PARAMETERS:-

I) UNIT - THE FILE UNIT WITH WHICH THE 'ZOP' FILE HAS BEEN OPENED.

II) ITYPE - GRTS ONLY - ZERO FOR AN OPERATOR MESSAGE
- NON ZERO FOR PASSWORD CARD

III) INBUF - GRTS ONLY - INPUT BUFFER (80 CHARACTERS)

EACH INPUT MESSAGE IS EXAMINED CHARACTER BY CHARACTER FOR ILLEGAL CHARACTERS (LESS THAN OCTAL 240) AND IF ANY ARE FOUND THE MESSAGE IS IGNORED. TRANSLATION IS PERFORMED WHERE NECESSARY ON THE INPUT CHARACTERS. HEADER INFORMATION (BLOCK SIZE, PASSWORD ETC.) IS WRITTEN TO THE ZOP FILE, FOLLOWED BY THE TEXT PORTION OF THE MESSAGE, AND FINALLY THE TRAILER SECTION (RECORD SEPARATORS, END OF TEXT CHARACTERS ETC.).

ERRORS MAY OCCUR DUE TO:-

A) ILLEGAL CHARACTERS IN THE INPUT MESSAGE

B) FILE SYSTEM ERRORS

C) ERRORS SENDING THE MESSAGE TO THE SYMBIONT

THE FOLLOWING SECTION DETAILS SPECIFIC EMULATOR ISSUES.

1) 7020 - NO CHARACTER TRANSLATION IS NECESSARY SO THE INPUT BUFFER IS ALSO USED AS THE OUTPUT BUFFER. A DC4 CHARACTER IS ADDED BEFORE THE ETX.

2) 2OOUT - MESSAGES MAY BE TRANSMITTED IN ASCII OR BCD. A PAD CHARACTER MAY BE ADDED BEFORE THE ETX TO ALIGN THE ETX ON AN EVEN BYTE BOUNDARY (CDC HARDWARE RESTRICTION). WHETHER OR NOT PADDING IS REQUIRED IS DEFINED BY THE SITE DEFINITION COMMAND PADCHAR.

3) GRTS - EVERY GRTS OPERATOR MESSAGE MUST BEGIN WITH A $ CHARACTER. A $ CHARACTER IS AUTOMATICALLY ADDED IF THE FIRST CHARACTER IN THE INPUT MESSAGE IS NOT A $ CHARACTER. THE FORMAT AND MEDIA CODES ARE ADDED BEFORE THE $ CHARACTERS. AS WITH 2OOUT THE ETX CHARACTER MUST BE ON AN EVEN BYTE BOUNDARY, AND SINCE THE RECORD SEPARATOR AND THE ETX MUST BE IN CONSECUTIVE POSITIONS, A BLANK CHARACTER IS ADDED BEFORE THE RECORD SEPARATOR IF NECESSARY TO ACHIEVE THE CORRECT ALIGNMENT. THE MESSAGE MAY BE EITHER A PASSWORD CARD, OR AN OPERATOR MESSAGE. THE DISTINCTION IS MADE IN PARAMETER ITYPE AND THE SYMBIONT IS INFORMED WHICH BY THE KEY WORD IN THE CALL TO ZWRITE.

MSPCFC

MSPCFC IS THE MODULE CALLED BY THE COMMAND PROCESS WHEN EMULATOR SPECIFIC ACTION IS REQUIRED FOLLOWING AN OPERATOR COMMAND. (MSPCFC RESIDES IN FILE WSXXXXX (WHERE XXXXX REPRESENTS 2OOUT, HASP ETC.)). A COMPUTED GO TO IS USED TO BRANCH TO THE CODE FOR A PARTICULAR COMMAND. EACH EMULATOR USES A DIFFERENT SUBSET OF THE COMMANDS. WHERE A COMMAND

IS NOT APPLICABLE TO AN EMULATOR THE MESSAGE 'COMMAND MEANINGLESS TO THIS EMULATOR' IS DISPLAYED FOR THE OPERATOR.

THE CODE FOR EACH COMMAND CHECKS THE VARIOUS FLAGS THAT MUST BE SET OR UNSET BEFORE THE ACTION CAN BE CARRIED OUT (E.G., 'TOSITE' DEFINED, LINE ENABLED/DISABLED ETC.). THE OPERATOR IS INFORMED OF THE SUCCESS OR FAILURE OF EACH COMMAND.

WHERE POSSIBLE THE CODE FOR THE SAME COMMAND IS THE SAME FOR EACH EMULATOR. CHECK THE SPECIFIC EMULATOR CODE FOR FURTHER DETALS.

THE ENDSITE COMMAND CODE REQUIRES FURTHER MENTION. WHEN THE ENDSITE COMMAND CODE IS ENTERED, ALL POSSIBLE COMBINATIONS OF ALLOWABLE SITE DEFINITION COMMANDS FOR THIS SITE ARE CHECKED TO ENSURE THEY HAVE BEEN SPECIFIED. IF THEY HAVE NOT BEEN SPEICIFIED THE OPERATOR IS INFORMED. CONFIG COMMANDS NOT RELEVANT TO THE EMULATOR MUST NOT HAVE BEEN SPECIFIED - IF THEY HAVE THE OPERATOR IS INFORMED. IF ANY ERRORS OCCUR THE ERROR FLAG IS SET AND COMMON CODE PREVENTS THE OPERATOR CONTINUING WITH THE CURRENT SITE UNTIL THE SITE DEFINITION FILE HAS BEEN MODIFIED.

MAIN

MAIN IS THE FIRST MODULE ENTERED WHEN THE WORKSTATION/SEND UTILITY IS SET RUNNING (MAIN RESIDES IN FILE WSXXXXX). ITS ONLY FUNCTION IS TO CALL THE MAIN CONTROL LOOP WITH THE EMULATOR NAME.

HEADER

HEADER (RESIDES IN FILE WSXXXXX) IS CALLED BY THE ENDSITE CODE IN MSPCFC, TO DISPLAY A STANDARD ERROR MESSAGE IF CONFIG COMMANDS NOT APPLICABLE TO THE PARTICULAR EMULATOR HAVE BEEN SPECIFIED.

1004 TRANSMIT AND RECEIVE

INTRODUCTION

THE 1004 TRANSMIT AND RECEIVE MODULES FIT INTO THE STANDARD WORKSTATION PROGRAM PRESENTING THE USER WITH A STANDARD INTERFACE. THE FOLLOWING SECTION DESCRIBES THE 1004 DATA FORMAT AND ALSO THE DATA STRUCTURES USED BY THE TRANSMIT AND RECEIVE MODULES.

4.0 1004 DATA FORMAT

A SUMMARY OF THE 1004 DATA BLOCKS IS PRESENTED HERE. FOR FURTHER INFORMATION SEE PE-T-492 (UNIVAC 1004 EMULATOR GUIDE).

A DATA BLOCK IN EITHER DIRECTION CONSISTS OF 330 CHARACTERS WHICH CONTAIN AN INTEGRAL NUMBER OF CARD OR LINE IMAGES. EACH IMAGE IS TERMINATED BY A 'NOT EQUALS' AND THE FINAL IMAGE OF A BLOCK IS TERMINATED BY A FURTHER 'NOT EQUALS' AND PADDED WITH ZEROS TO 330 CHARACTERS.

DATA TYPES CAN BE SPLIT INTO TWO, CARD IMAGES AND PRINTER DATA. A CARD IMAGE MAY BE SENT OR RECEIVED BY A PRIME. A CARD IMAGE CONSISTS OF 80 CHARACTERS, MORE THAN 80 CHARACTERS BEING TRUNCATED AND LESS THAN 80 CHARACTERS BEING PADDED WITH BLANKS. A CERTAIN AMOUNT OF BLANK COMPRESSION IS DONE AS FOLLOWS.

> SPLIT THE CARD IMAGE INTO TEN FIELDS OF EIGHT BITS. IF THERE IS MORE THAN ONE TRAILING BLANK IN A FIELD REPLACE THE BLANKS WITH A 'BLOB', (AN OCTAL 77 IN XS3 CODE).

AN IMAGE WHICH IS COMPRESSED IN THIS WAY IS PRECEDED BY THE CHARACTER C. IF 'BLOBS' OR 'NOT EQUALS' APPEAR IN THE IMAGE AS DATA THEN THE IMAGE IS PRECEDED BY THE CHARACTER B.

PRINTER DATA IS SIMILARLY COMPRESSED AND NEVER CONTAINS A 'BLOB' OR 'NOT EQUALS' AS DATA. A RECORD MAY CONTAIN 132 CHARACTERS AND IS PRECEDED, NOT BY A B OR C, BUT BY PRINTER CONTROL CHARACTERS. THE TWO CHARACTERS IMMEDIATELY PRECEDING THE DATA ARE FORMS CONTROL AND CONTAIN XS3 NUMERICS. THE NUMBER FORMED IS THE NUMBER OF LINES TO THROW. FURTHERMORE, IF THESE NUMERICS ARE PRECEDED BY AN * THEN TOP OF FORM IS REQUIRED BEFORE THE LINE CONTROL. (THIS DEFINITION VARIES SLIGHTLY FROM PE-T-492 BUT HAS BEEN FOUND TO BE CORRECT).

4.1 1004 TRANSMIT PROCESSING

TRANSMIT PROCESSING TAKES PLACE WHEN THE COMMON WORKSTATION OR RJE UTILITY CALLS THE SUBROUTINE TRANSM. THE FILE NAME, PASSED AS AN ARGUMENT, IS PICKED UP AND CONVERTED INTO A TRANSMIT FILE WHICH IN TURN IS READ BY THE SYMBIONT AND TRANSMITTED TO THE HOST. FOLLOWING IS A DESCRIPTION OF THE TECHNIQUES USED TO PROCESS THE SOURCE FILE INTO A TRANSMIT FILE.

4.1.1 DESCRIPTON OF VARIABLES

THE MAJORITY OF THE VARIABLES USED IN TRANSMIT PROCESSING ARE HELD IN COMMON. THERE ARE TWO NAMED COMMON AREAS, TRACOM AND TSVCOM.

BECAUSE TRANSMIT PROCESSING MAY UNDERGO CHANGES IN THE FUTURE TSVCOM HAS BEEN USED TO SAVE INDIVIDUAL VARIABLES INTO ARRAY ELEMENTS. (EACH ELEMENT OF THE ARRAY EQUATES TO THE VARIABLE FOR A GIVEN LINE). IT MAY BE DESIRABLE IN THE FUTURE TO PROCESS A TRANSMIT SOURCE FILE A SECTION AT A TIME. TRACOM IS THE MAIN VARIABLE POOL. EACH INDIVIDUAL VARIABLE HAS A ONE LINE DESCRIPTION IN THE INSERT FILE RJE1004>TRACOM.

ONE FINAL NOTE ON VARIABLES IS THAT ANY VARIABLE ENDING WITH A $ SIGN IS A FORTRAN PARAMETER.

4.1.2 PROGRAM STRUCTURE

TRANSMIT PROCESSING IS SPLIT INTO SEVERAL SUBROUTINES. AS A GENERAL RULE, SUBROUTINES HAVE NO PARAMETERS AND ALL THE INFORMATION IS PASSED VIA THE COMMON AREA.

4.1.2.1 TRANSM

```
CALL TRANSM (TREE,LENGTH,TYPE)
```

DESCRIPTION OF ARGUMENTS:

ALL DUMMY ARGUMENTS ARE INTERGER*2

```
TREE    -    A STANDARD TREENAME UP TO 32 CHARACTERS
             IN LENGTH.
LENGTH  -    NUMBER OF CHARACTERS IN TREE.
TYPE    -    NOT USED.
```

THIS ROUTINE HANDLES ALL THE OPENING AND CLOSING OF FILES NECESSARY FOR THE TRANSMIT PROCESS AND IS CALLED BY THE WORKSTATION OR SEND UTILITY. IT IS RESPONSIBLE FOR CLEANING UP AFTER AN ERROR AND EXITING GRACEFULLY. IT IS ALSO RESPONSIBLE FOR INITIALISING AND PRINTING STATISTICS.

4.1.2.2 TPUTFL

CALL TPUTFL

THERE ARE NO ARGUMENTS FOR THIS ROUTINE.

TPUTFL IS RESPONSIBLE FOR CONVERTING THE SOURCE FILE INTO A TRANSMIT FILE. IT DOES THIS, ROUGHLY SPEAKING, ON A CHARACTER BY CHARACTER BASIS. TO INPUT OR OUTPUT A CHARACTER IT USES TWO INTERNAL SUBROUTINES TGETC AND TPUTC. HAVING OBTAINED A CHARACTER TPUTFL DECIDES WHAT TO DO WITH IT. FOR INSTANCE, A BLANK OR SEVERAL BLANKS MAY HAVE BEEN FOUND IN WHICH CASE COMPRESSION TAKES PLACE. IF A NEW LINE CHARACTER OCCURS, THEN END OF LINE EVENT TAKES PLACE.

THE INTERNAL SUBROUTINE TGETC PICKS UP A CHARACTER FROM THE INPUT BUFFER, REFRESHING THE BUFFER WHEN REQUIRED. THIS CHARACTER IS CHECKED FOR VARIOUS SPECIAL CHARACTERS AND ACTION TAKEN ACCORDINGLY. TGETC IS ALSO MAINLY RESPONSIBLE FOR DEALING WITH CONCAT FILES.

THE OTHER INTERNAL SUBROUTINE, TPUTC SIMPLY ADDS A CHARACTER TO THE OUTPUT BUFFER. BECAUSE OF THE DATA STRUCTURE, A CALL TO TPUTC NEVER CAUSES A BUFFER OVERFLOW. (A 1004 TRANSMIT BLOCK IS ALWAYS 330 CHARACTERS LONG. IT IS THEREFORE A SIMPLE TASK TO ENSURE THE OUTPUT BUFFER IS FLUSHED EVERY FEW COMPLETE BLOCKS).

ALL THE OTHER SUBROUTINES ARE KEPT IN A SINGLE FILE NAMED TSUBS RESIDING IN RJE1004.

4.1.2.3 TOOBIG

NO ARGUMENTS.

WHEN A BLOCK BECOMES GREATER THAN 330 CHARACTERS IT HAS TO BE TERMINATED AT THE LAST COMPLETED RECORD (OR IMAGE). A POINTER IS HELD INDICATING WHERE THE CURRENT IMAGE BEGINS AND BY INFERENCE WHERE THE PREVIOUS ONE ENDS. THE BLOCK IS TERMINATED BY APPENDING AN XS3 'NOT EQUALS' AND PADDING ANY SPARE CHARACTERS WITH BLANKS. AN EOM IS ADDED TO COMPLETE THE RECORD. THE BUFFER IS CHECKED AGAINST A PARAMETER INDICATING THE END OF BUFFER LIMIT AND IF NECESSARY IS SUBSEQUENTLY FLUSHED TO FILE. A NEW HEADER IS INSERTED AND ANY OF THE CURRENT RECORD APPENDED TO THE BUFFER (THIS HAS PREVIOUSLY BEEN COPIED TO A TEMPORARY STORAGE AREA).

4.1.2.4 TGCHR

NO ARGUMENTS.

THIS ROUTINE IS USED INFREQUENTLY TO OBTAIN THE NEXT CHARACTER FROM THE INPUT BUFFER REFRESHING THE BUFFER WHEN REQUIRED. THIS ACTION IS ALSO PERFORMED IN TGETC IN-LINE.

4.1.2.5 TBGBUF

NO ARGUMENTS.

SIMPLY RESETS SOME POINTERS AND LOCATIONS FOLLOWING A BUFFER FLUSH.

4.1.2.6 ENDLIN

NO ARGUMENTS.

THIS ROUTINE IS CALLED AT THE END OF EACH INPUT LINE. IT IS RESPONSIBLE FOR DETERMINING IF A SPECIAL CHARACTER HAS APPEARED IN THE OUTPUT IMAGE MEANING THAT THIS IMAGE HAS TO BE UNCOMPRESSED. (EVERY IMAGE IS COMPRESSED BY DEFAULT). THE ROUTINE POSITIONS A 'B' OR A 'C' AT THE CORRECT LOCATION AND APPENDS A 'NOT EQUALS' TO THE IMAGE. SOME VARIABLES ARE RESET AND PROCESSING CONTINUES.

IT IS IMPORTANT TO NOTE THAT AN IMAGE MUST BE EXPANDED (WHEN NECESSARY) BEFORE A CHECK IS MADE FOR GREATER THAN 330 CHARACTERS.

4.1.2.7 UNCOMP

NO ARGUMENTS.

THIS IS THE ROUTINE THAT UNCOMPRESSES (EXPANDS) A PREVIOUSLY COMPRESSED IMAGE WHEN A SPECIAL CHARACTER IS FOUND AS DATA.

4.1.2.8 TINIT

NO ARGUMENTS.

INITIALISES ALL NECESSARY VARIABLES WHEN A NEW FILE IS ABOUT TO BE PROCESSED.

4.1.2.9 TRESTO AND TSAVE

NO ARGUMENTS.

THESE ROUTINES RESTORE OR SAVE THE DATA STRUCTURE FOR A PARTICULAR LINE. THIS HAS BEEN DONE FOR POSSIBLE FUTURE EXPANSION. THERE IS ALSO A LARGE AMOUNT OF REDUNDANT INFORMATION HELD BY THESE ROUTINES WHICH MAY LATER BE TRIMMED OUT.

4.1.3 DATA STRUCTURE

THE DATA STRUCTURE FOR TRANSMIT PROCESSING REVOLVES AROUND THE INPUT AND, MORE IMPORTANTLY, THE OUTPUT BUFFERS. FIG. 1 GIVES A DIAGRAMTIC REPRESENTATION OF THESE BUFFERS AND THE KEY POINTERS ASSOCIATED WITH THEM.

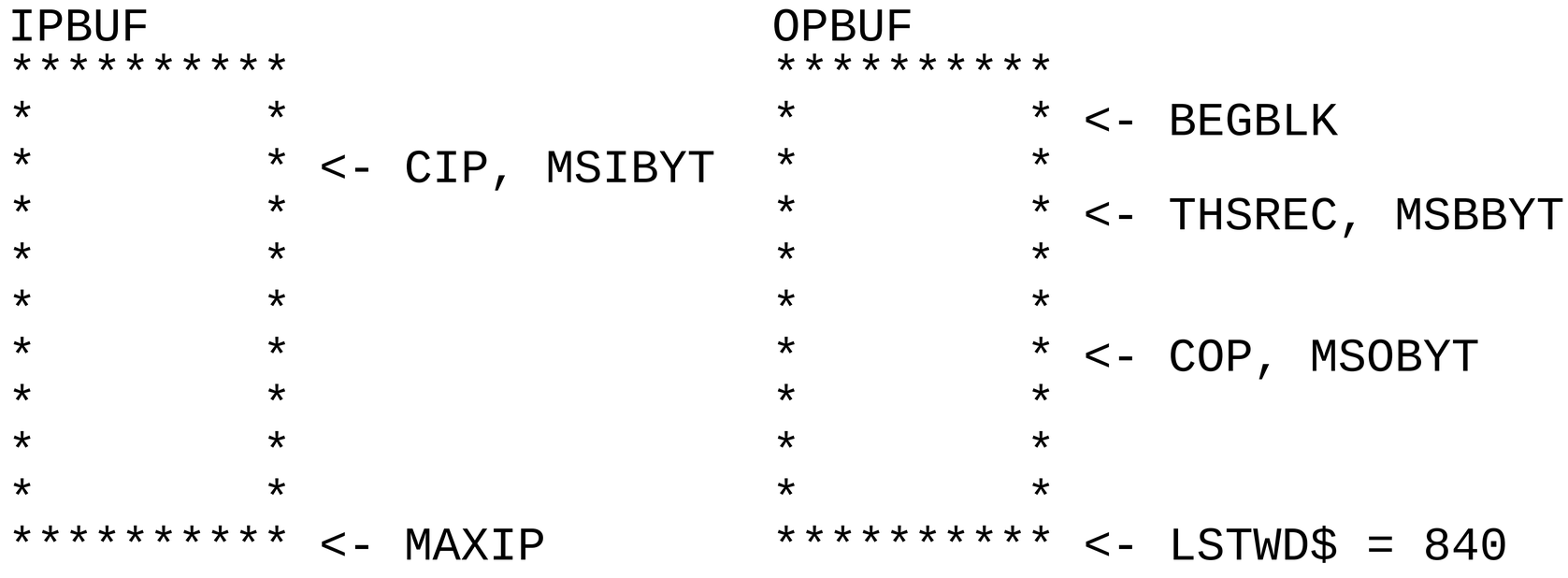


FIG. 1 - INPUT, OUTPUT BUFFERS

IPBUF, THE INPUT BUFFER, IS VERY SIMPLE IN OPERATION. WHEN EMPTY IT IS FILLED TO 1024 WORDS WITH THE EXCEPTION OF THE LAST BUFFER WHICH CAN BE ANYTHING FROM 0 TO 1024 WORDS. THE BUFFER IS FILLED BY A READ CALL TO PRWF$$ AND MAXIP IS SET TO THE NUMBER OF WORDS READ. CIP POINTS TO THE NEXT WORD TO BE PROCESSED AND MSIBYT, WHEN TRUE, INDICATES THAT THE LEFT HAND (MOST SIGNIFICANT) BYTE IS TO BE USED.

OPBUF IS THE OUTPUT BUFFER AND IS SOMEWHAT MORE COMPLEX IN OPERATION AS IS SIGNIFIED BY THE NUMBER OF POINTERS ASSOCIATED WITH IT. THE ENTIRE BUFER CAN HANDLE FIVE FULL TRANSMIT BLOCKS BEFORE IT IS FLUSHED TO FILE. EACH BLOCK CONTAINS A WORD WITH 168, INDICATING THE BLOCK LENGTH, FOLLOWED BY A SOM, FOLLOWED BY 165 DATA WORDS AND FINISHING WITH AN EOM. THUS WE ARRIVE AT THE FIGURE OF 840 FOR A FIVE BLOCK BUFFER.

IMAGES ARE OBTAINED FROM THE INPUT BUFFER, TRANSLATED, COMPRESSED AND OUTPUT TO OPBUF. BEGBLK POINTS TO THE FIRST DATA WORD IN THE CURRENT TRANSMIT BLOCK. THSREC POINTS TO THE FIRST DATA WORD FOR THE CURRENT IMAGE BEING PROCESSED (MSBBYT INDICATES WHICH BYTE THE IMAGE STARTS ON). THIS WILL NORMALLY BE THE 'B' OR 'C' AT THE START OF AN IMAGE. COP AND MSOBYT INDICATE THE NEXT AVAILABLE WORD/BYTE IN THE OUTPUT BUFFER. WHEN A SUFFICIENT NUMBER OF IMAGES HAVE BEEN PROCESSED THE DIFFERENCE BETWEEN COP AND BEGBLK WILL BE GREATER THAN 165 OR 330 CHARACTERS. AT THIS POINT THE BLOCK HAS TO BE TERMINATED AT THE LAST COMPLETED RECORD. THIS IS INDICATED BY THSREC. EXCESS WORDS ARE COPIED TO A TEMPORARY BUFFER AND THE BLOCK IS PADDED OUT TO 330 CHARACTERS AND TERMINATED. A NEW BLOCK IS NOW SET UP AND THE TEMPORARY BUFFER COPIED BACK TO OPBUF. UNFORTUNATELY, IT MAY HAPPEN THAT THERE IS A ONE BYTE OFFSET BETWEEN THE TEMPORARY BUFFER AND THE REAL BUFFER, CAUSING A SLOWER COPYING OPERATION TO OCCUR.

AT THE POINT THE BLOCK IS TERMINATED, A CHECK IS MADE ON HOW LARGE THE BUFFER IS AND, WHEN NECESSARY, THE BUFFER IS FLUSHED AND THE TEMPORARY BUFFER IS COPIED TO THE START OF OPBUF.

4.2 1004 RECEIVE PROCESSING

RECEIVE PROCESSING CONVERTS DATA PLACED IN AN R FILE BY THE SYMBIONT INTO A PRIME STYLE FORMAT. THIS CAN BE IN THE STYLE OF A SPOOL FILE OR IT MAY BE LEFT IN BINARY FORMAT, I.E. NO TRANSLATION, IN THE Q UFD. RECEIVE PROCESSING IS ONLY UNDERTAKEN BY THE WORKSTATION AND MAY RUN ON MULTIPLE LINES. THE WORKSTATION MAY MAKE SEVERAL CALLS TO THE RECEIVE PROCESSING ROUTINES TO COMPLETE TRANSLATION OF A FILE. THIS ALLOWS THE OPERATOR TO PERFORM OTHER TASKS SIMULTANEOUSLY.

4.2.1 DESCRIPTION OF VARIABLES

FOR THE PURPOSE OF EFFICIENCY, THE RECEIVE DATA STRUCTURE USES A SIMILAR TECHNIQUE TO THE TRANSMIT DATA STRUCTURE. IN THE MAIN, ONLY NON-DIMENSIONAL VARIABLES ARE USED FOR THE MAIN BULK OF PROCESSING. HOWEVER, AFTER A FEW RECEIVE BLOCKS HAVE BEEN PROCESSED A RETURN IS MADE TO THE WORKSTATION. IT IS QUITE POSSIBLE THAT THE NEXT INVOCATION OF RECEIVE PROCESSING WILL BE FOR A DIFFERENT LINE. THEREFORE A DIFFERENT DATA SET HAS TO BE USED. AS WILL BE SEEN, THERE ARE TWO SUBROUTINES THAT HANDLE THE SAVING AND RESTORING OF THESE DATA SETS. RECCOM IS THE MAIN VARIABLE DEFINITION COMMON BLOCK. RSVCOM IS WHERE ALL THE NECESSARY NON-DIMENSIONAL ELEMENTS ARE SAVED BETWEEN RECEIVE PROCESSING INVOCATIONS. ALL PARAMETERS END WITH A $ SYMBOL WITH ONE EXCEPTION, MAXLOP, WHICH DETERMINES THE NUMBER OF INPUT BUFFERS TO PROCCESS BEFORE RETURNING TO THE WORKSTATION.

RECCOM, RSVCOM AND PARAMETERS ARE IN RJE1004>RECCOM. EACH VARIABLE HAS A ONE LINE DESCRIPTION OF ITS USE.

4.2.2 PROGRAM STRUCTURE

RECEIVE PROCESSING IS SPLIT INTO SEVERAL SUBROUTINES. THE ONLY ROUTINE WITH AN ARGUMENT IS THE LINK ROUTINE BETWEEN THE WORKSTATION AND RECEIVE PROCESSING. ALL THE ROUTINES USE VARIABLES IN THE MAIN COMMON BLOCKS.

4.2.2.1 RECV

```
CALL RECV (TODO)

TODO - IS A LOGICAL VARIABLE SET TRUE ON RETURN WHEN THERE IS
       MORE PROCESSING KNOWN TO BE DONE.
```

RECV IS THE LINK ROUTINE TO THE WORKSTATION. ITS PURPOSE IN LIFE IS TO ENSURE INPUT AND OUTPUT FILES ARE OPENED AND CLOSED, TO RESTORE AND SAVE THE DATA SETS AND TO LOOK AFTER ANY EARLY INITIALISATION PROBLEMS. IT ALSO HANDLES CLEANING UP AFTER ERRORS HAVE OCCURRED AND THE PRINTING OF STATISTICS.

4.2.2.2 RPUTFL

CALL RPUTFL

THIS ROUTINE OBTAINS A BUFFER FROM THE INPUT FILE BY A CALL TO PRWF$$. EACH RECORD IS PROCESSED IN TWO PARTS. INITIALLY, IT IS INTERROGATED FOR PRINT CONTROL OR 'B' OR 'C'. BLANK COMPRESSION IS ALWAYS ASSUMED UNLESS A 'B' IS FOUND. PRINT CONTROL IS ACTED UPON BY OUTPUTTING RELEVANT CONTROL CHARACTERS. THE REST OF THE IMAGE IS PROCESSED UNTIL 'NOT EQUALS' IS FOUND. A 'NOT EQUALS' INDICATES THE END OF AN IMAGE AND A CALL IS MADE TO DETERMINE WHETHER THE END OF A LOGICAL FILE HAS BEEN REACHED.

EACH IMAGE IS PROCESSED CHARACTER BY CHARACTER. A CHARACTER IS OBTAINED FROM THE INPUT BUFFER BY A CALL TO RGETC. THIS MAY BE RETURNED AS A HORIZONTAL TAB CHARACTER WITH A CHARACTER COUNT, OR SIMPLY AS A CHARACTER. THE CHARACTER IS TRANSLATED, WHEN REQUIRED, AND PLACED IN THE OUTPUT BUFFER BY A CALL TO RPUTC. NOTICE THAT RGETC AND RPUTC ARE ROUTINES INTERNAL TO RPUTFL.

THE MAIN BODY OF SUBROUTINES ARE HELD IN RSUBS. THEY COMPRISE THE FOLLOWING:

4.2.2.3 RGTCHR

THE INFREQUENTLY USED ROUTINE TO OBTAIN A CHARACTER FROM THE INPUT BUFFER. THIS ROUTINE IS CALLED FOR OBTAINING PRINT CONTROL OR THE B OR C CHARACTER.

4.2.2.4 RRESTO RSAVE

THESE ROUTINES RESTORE AND SAVE THE CURRENT DATA SET TO NON-DIMENSIONED VARIABLES.

4.2.2.5 BUFLSH

FLUSHES THE OUTPUT BUFFER BY A CALL TO PRWF$$.

FINALLY, THE ROUTINE THAT HANDLES END OF LOGICAL FILE DETECTION.

4.2.2.6 EOFDET

THE EOFDET SUBROUTINE SUPPLIED WITH THE SYSTEM IS TAILORED FOR A SPECIFIC SITE. THIS ROUTINE IS CALLED ONCE PER INPUT LINE. THE ARRAY EOFBUF (CONTAINING ONE, RIGHT JUSTIFIED, CHARACTER PER WORD) IS INTERROGATED FOR A KNOWN CHARACTER SEQUENCE. EOFBUF CONTAINS EOFPNT CHARACTERS. IT MAY BE THAT THE END OF A LOGICAL JOB CAN BE DEDUCED FROM INTERROGATING ONE LINE ONLY. IT IS MORE LIKELY THAT SEVERAL CONSECUTIVE LINES ARE REQUIRED TO MAKE THIS DEDUCTION. WHEN IT IS DECIDED THAT THE END OF A LOGICAL JOB HAS BEEN REACHED, THE LOGICAL FLAG, ENDLOG IS SET TRUE.

A SITE DEFINITON VARIABLE, EOFKEY(NLINE) IS USED IN END OF FILE DETECTION. FIRSTLY, IF THIS KEY IS ZERO, NO END OF FILE DETECTION

SHOULD BE DONE AND EOFDET SHOULD RETURN IMMEDIATELY. GENERALLY SPEAKING EOFKEY WILL BE SET TO ONE AND NEEDS ONLY TAKE ON AN ALTERNATIVE VALUE IF THE USER IS DEALING WITH MORE THAN ONE HOST OR MORE THAN ONE HOST MACHINE WHOSE END OF LOGICAL JOB SEQUENCE VARIES.

FOR INSTANCE, A USER MAY ALWAYS RUN SMLC LINE 0. HE MAY HAVE TWO DIFFERENT 1004 SITES E.G. ABC AND XYZ. EACH SITE REQUIRES A DIFFERENT END OF FILE DETECTION SEQUENCE. THE ONLY WAY THE 1004 PROGRAM CAN DECIDE WHICH SEQUENCE IS REQUIRED IS BY THE SETTING OF EOFKEY FOR INDEX 1 (LINE 0).

CLEARLY, IT IS THE FIELD ANALYSTS OR USERS RESPONSIBLITY TO TAILOR THE EOFDET ROUTINE TO A SITES NEEDS.

4.2.3 DATA STRUCTURE

RECEIVE PROCESSING, USES TWO OUTPUT BUFFERS PER LINE AND ONE INPUT BUFFER FOR ALL LINES. THE INPUT BUFFER HOLDS ONE RECEIVE BLOCK AT A TIME. THE WORD/BYTE POINTERS ASSOCIATED WITH IPBUF ARE CIP/MSIBYT. MSIBYT, WHEN TRUE, INDICATES THE MOST SIGNIFICANT BYTE OF THE WORD POINTED TO BY CIP IS TO BE USED.

THERE ARE TWO OUTPUT BUFFERS PER LINE TO ALLOW FOR BINARY (PUNCH) RECEIVE FILE PROCESSING. THE INTERACTION OF THESE TWO IF ALTERNATE PRINT/PUNCH FILES ARE PROCESSED IS LARGELY UNTESTED AND WOULD BE A GOOD SOURCE OF ERROR IF ODD SITUATIONS ARISE. IN THE DESIGN STAGE, NOT TOO MUCH ATTENTION WAS GIVEN TO BUFFER SIZE AS THESE ARE VERY OFTEN LARGE, UNUSED AREAS AND THEREFORE DO NOT NOTICABLY DEGRADE PERFORMANCE. HOWEVER, DUE TO THE GROWING SIZE OF THE COMMON USER INTERFACE, IT WOULD APPEAR THAT PERHAPS IT WOULD BE A BETTER IDEA TO HAVE MULTIPLE INPUT BUFFERS (BEING RELATIVELY SMALL) AND ONLY ONE COMMON OUTPUT BUFFER. IF THIS BECOMES A REQUIREMENT THEN SUITABLE MODIFICATION WILL BE MADE AND DOCUMENTED.

THE OUTPUT BUFFERS ALL USE THE SAME POINTERS, COP AND MSOBYT, WHOSE USE IS IDENTICAL TO CIP AND MSIBYT.

THERE IS A FAIR AMOUNT OF CLEANING UP THAT NEEDS TO BE DONE, PATICULARLY IN THE AREA OF STATISTICS GATHERING (NOTABLY RECV). HOWEVER, BECAUSE THE CODE IN THESE AREAS IS NOT OFTEN REFERENCED, NOT MUCH WILL BE GAINED IN PERFORMANCE.

********** 200UT TRANSMIT MODULE DOCUMENTATION **********

SECTION 5.1

INTRODUCTION

THE TRANSMIT MODULE IS CALLED BY THE WORKSTATION OR THE SEND UTILITY WHEN THE USER ISSUES A SEND COMMAND. ITS FUNCTION IS TO READ THE FILE WHICH IS TO BE SENT, CONVERT IT TO THE SPECIFIED CODE, AND PUT THE RESULT INTO A FILE FOR THE SYMBIONT TO READ AND TRANSMIT TO THE CDC HOST.

THE TRANSMIT MODULE IS MADE UP OF THE FOLLOWING ROUTINES :-

NAME	FUNCTION
TRANSM	MAIN TRANSMIT ROUTINE: IT OPENS THE FILES, INITIALISES THE STATISTICS COUNTS AND WORKS THROUGH THE INPUT FILE, CLOSING IT AND THE OUTPUT FILE ON COMPLETION.
TPUTFL	CALLED BY TRANSM TO WRITE A BLOCK TO THE OUTPUT FILE.
TPUTC	CALLED BY TRANSM TO STORE A CHARACTER IN THE OUTPUT BUFFER.
TGETC	CALLED BY TRANSM TO GET THE NEXT CHARACTER FROM THE INPUT FILE.
RDCON	CALLED BY TRANSM TO CHECK WHETHER THE INPUT FILE IS A 'CONCAT' FILE, AND IF SO TO GET DETAILS OF IT.

++

TRANSM

THE WORKSTATION CODE CALLS TRANSM WITH 3 ARGUMENTS :-

I) THE NAME OF THE FILE TO BE SENT,
II) THE LENGTH OF THAT NAME,
III) A TYPE ARGUMENT (NOT USED IN THE CASE OF 200UT).

TRANSM IS ENTERED WITH THE VARIABLE 'TLINE' (IN GENERAL COMMON) INDICATING THE LINE TO BE PROCESSED. THIS IS COPIED INTO VARIABLE LINX IN LOCAL COMMON AND USED AS THE LINE INDEX VALUE THROUGHOUT THE 200UT TRANSMIT ROUTINES. TRANSM CURRENTLY PROCESSES A COMPLETE FILE BEFORE RETURNING CONTROL TO THE WORKSTATION. HOWEVER, THE BUFFERS ARE STRUCTURED IN ANTICIPATION OF THE NEXT PHASE OF THE EMULATORS, WHERE EACH CALL TO TRANSM WILL PRODUCE AN INTEGRAL NUMBER OF OUTPUT BLOCKS, BUT MIGHT ONLY PROCESS PART OF THE INPUT FILE. FOR THIS REASON, MULTIPLE BUFFERS ARE NEEDED - ONE PER LINE - FOR THE INPUT STREAMS, BUT A SINGLE BUFFER IS SUFFICIENT FOR THE OUTPUT.

THE ROUTINE BEGINS BY GETTING THE SITE NAME AND LINE INDEX FOR THE LINE ON WHICH THE FILE IS TO BE TRANSMITTED, THEN INITIALISING THE STATISTICS TIMER LOCATIONS. SUBROUTINE OPNFIL IS USED TO OPEN THE INPUT FILE, AND THEN TO OPEN AN OUTPUT FILE (NAMED T-DUMY , AND RENAMED TO T-NNNN ON SUCCESSFUL COMPLETION OF THE INPUT FILE PROCESSING). IF INPUT AND OUTPUT FILES ARE OPENED OK, STATISTICS LOCATIONS ARE CLEARED, THEN RDCON IS CALLED TO SEE IF THE INPUT FILE IS A 'CONCAT' FILE. THEN THE OUTPUT FILE HEADER IS WRITTEN (SEE THE SECTION 'OUTPUT FILE FORMAT') AND IF THE AUTOREAD OPTION IS IN USE, THE OPERATOR 'READ' MESSAGE IS WRITTEN TO THE FILE BEFORE STARTING ON THE MAIN LOOP OF THE ROUTINE.

THE CENTRAL LOOP OF THE ROUTINE (LABELS 116 THRU 200) IS TO GENERATE OUTPUT BLOCKS OF 12 CARDS, 80 CHARACTERS PER CARD. (LABEL 170 IS THE

END OF A SINGLE CARD). SUBROUTINE TGETC IS USED TO GET THE NEXT CHARACTER FROM THE INPUT FILE, TPUTC TO STORE THE NEXT OUTPUT CHARACTER, AND TPUTFL TO WRITE EACH RECORD, AFTER ADDING THE RECORD TRAILER. THE LOOP HAS TO EXPAND HORIZONTAL AND VERTICAL TABS ON THE INPUT FILE, TO SPACES AND BLANK CARDS ON THE OUTPUT FILE. IT ALSO HAS TO CHECK FOR THE @ CHARACTER IN COLUMN 1 OF A CARD, (AS THIS IS THE CDC END OF RECORD) AND REPLACE IT BY 'ESCAPE' 'END OF RECORD' ON THE OUTPUT FILE.

WHEN THE END OF INPUT FILE IS FOUND, (LABEL 260) AN 'END OF FILE' CARD ('ESCAPE' 'END OF FILE' 79*SPACES) IS STORED, AND THE BLOCK IS THEN OUTPUT BY TPUTFL, USING THE E2 CODE TO INDICATE THAT IT IS THE LAST DATA BLOCK. TPUTFL IS CALLED AGAIN TO OUTPUT THE 'ZERO WORD' TRAILER RECORD.

HAVING COMPLETED THE FILE, XYUPDT IS CALLED TO CLOSE THE OUTPUT FILE, AND CHANGE THE NAME READY FOR THE SYMBIONT TO PICK UP. THE STATISTICS ROUTINE IS CALLED TO OUTPUT STATS. IF REQUESTED, AND TO OUTPUT A MESSAGE INDICATING THAT THE 'SEND' COMMAND IS COMPLETE. THE INPUT FILE UNIT IS THEN FREED, AND CONTROL RETURNED TO THE WORKSTATION.

IF ERRORS ARE FOUND (EG. ON THE INPUT FILE) THEN THE FILE UNITS ARE FREED, AND AN ERROR MESSAGE OUTPUT, BUT NO FURTHER ACTION IS TAKEN. (THE COMMON VARIABLE ICODE IS USED THROUGHOUT THE TRANSMIT ROUTINES AS THE ERROR INDICATOR).

++

TPUTFL

THIS IS THE SUBROUTINE THAT IS CALLED BY TRANSM TO WRITE BLOCKS TO THE OUTPUT FILE - EXCEPT THE FILE HEADER BLOCK, WHICH IS WRITTEN BY TRANSM.

THERE IS ONE ARGUMENT TO TPUTFL, WHICH SPECIFIES THE E CODE REQUIRED AT THE END OF THE BLOCK BEING WRITTEN (SEE THE SECTION 'OUTPUT FILE FORMAT').

IF THE ARGUMENT (VARIABLE 'END') IS ZERO, THE FILE TRAILER RECORD IS WRITTEN - A SINGLE WORD RECORD - THEN CONTROL IS RETURNED TO TRANSM.

IF THE ARGUMENT IS NON ZERO, TPUTFL ADDS THE 'ESCAPE' 'ECODE' 'END OF MESSAGE' TRAILER TO THE DATA RECORD, BUILDS THE RECORD HEADER, AND WRITES OUT THE RECORD. (TPUTC IS USED TO STORE THE CHARACTERS IN THE OUTPUT BLOCK). NOTE THAT THE 'END OF MESSAGE' CHARACTER MUST BE IN THE LEFT HAND BYTE, SO TPUTFL ADDS A PAD CHARACTER AFTER THE ECODE IF NECESSARY.

OUTPUT FILE FORMAT

```
FILE HEADER       2 WORDS.

      WORD 0      ZERO
      WORD 1      BIT 13   =1 FOR AUTOREAD,  0 FOR NON AUTOREAD
                  BIT 14   =1 FOR BCD CODE,  0 FOR ASCII CODE.

DATA RECORDS      MINIMUM 6 WORDS, MAXIMUM 491.

      WORD 0      L*64 +1              )
      WORD 1      L-2                  )    RECORD HEADER
      WORD 2      ZERO                 )
      WORD 3      DATA ....
                       ....

      WORD L-1    'ESCAPE'             ECODE     )   RECORD TRAILER
      WORD L      'END OF MESSAGE'     NULL      )

FILE TRAILER      1 WORD.
      WORD 0      ZERO.
```

ON DATA RECORDS, IF THE LAST DATA CHARACTER IS IN THE FIRST BYTE OF THE WORD, THEN THE RECORD TRAILER WILL BE ORGANISED SO THAT THE 'END OF MESSAGE' IS IN THE FIRST BYTE OF A WORD, THUS .....

```
      WORD L-2    ...          'ESCAPE'          )
      WORD L-1    ECODE       PAD CHAR.          )    RECORD TRAILER
      WORD L      'END OF MESSAGE'  NULL         )
```

THE DATA WILL BE EITHER

I) AN OPERATOR MESSAGE; IN WHICH CASE THE ECODE IN THE RECORD TRAILER WILL BE E1 FOR BCD, OR E4 FOR ASCII.

OR

II) CARD DATA, BLOCKS OF 12 CARDS, 80 CHARACTERS EACH; IN WHICH CASE THE ECODE IN THE RECORD TRAILER IS AN E3 NORMALLY, OR AN E2 FOR THE LAST BLOCK IN THE FILE.

++

TPUTC

THIS SUBROUTINE IS CALLED BY TRANSM TO STORE A CHARACTER IN THE OUTPUT BUFFER. THERE IS ONE ARGUMENT, NAMELY THE CHARACTER WHICH IS TO BE STORED (VARIABLE 'CH').

THE LOGICAL VARIABLE OSIDE INDICATES WHERE IN THE WORD THE CHARACTER SHOULD BE STORED (OSIDE = .TRUE. FOR RIGHT HAND BYTE) AND COMMON VARIABLE N INDICATES THE CURRENT WORD IN THE BUFFER.

THE BUFFER BOPBUF IS EQUIVALENCED TO OPBUF(4) TO ALLOW FOR THE 3 WORD RECORD HEADER THAT IS ADDED BY TPUTFL. OPBUF IS DECLARED AS 500 WORDS TO ALLOW FOR THE 3 HEADER WORDS, 2 OR 2-AND-A-HALF WORDS OF TRAILER, AND UP TO 12 CARDS OF 80 CHARACTERS WHICH MAY INCLUDE SOME @ AND 'END OF FILE' CHARACTERS . (THE @ CHARACTER IN COLUMN 1 IS REPRESENTED BY 'ESCAPE' 'END OF RECORD', SO TAKES UP 2 CHARACTER POSITIONS ; SIMILARLY THE 'END OF FILE' IS PRECEDED BY AN 'ESCAPE' CHARACTER, SO AGAIN 2 CHARACTER POSITIONS ARE USED).

NOTE AS A TEMPORARY MEASURE, THIS ROUTINE HAS BEEN INCORPORATED INTO SUBROUTINE TRANSM, AS A DUMMY SUBROUTINE, STARTING AT LABEL 700. THIS IS TO AVOID THE OVERHEAD OF FORTRAN SUBROUTINE CALLS.

THE SUBROUTINE HAS ALSO BEEN LEFT IN ITS NORMAL FORM TO BE CALLED DIRECTLY FROM SUBROUTINE TPUTFL. THIS WAS DONE BECAUSE THE CALLS FROM TPUTFL ARE NOT FREQUENT ENOUGH TO MERIT INCORPORATING TPUTC WITHIN TPUTFL AS WELL.

++

TGETC

THIS SUBROUTINE IS CALLED BY TRANSM TO GET THE NEXT CHARACTER FROM THE INPUT FILE. THERE ARE NO ARGUMENTS, AS THE CHARACTER IS RETURNED IN COMMON VARIABLE 'CHAR' (IN COMMON BLOCK /TRANZ/). THE COMMON VARIABLE 'GOT' INDICATES THE TYPE OF CHARACTER RETURNED.

THE RESULTS ARE RETURNED IN COMMON AS FOLLOWS :-

CHAR	THE NEXT CHARACTER.	
GOT	=1	ORDINARY CHARACTER RETURNED.
	=2	END OF RECORD FOUND (IGNORE CHAR)
	=3	END OF INPUT FILE FOUND (IGNORE CHAR)
	=4	HORIZONTAL TAB (CHAR=NO. OF SPACES REQUIRED)
	=5	VERTICAL TAB (CHAR=NO. OF LINES REQUIRED)
	=6	ERROR IN TGETC (ICODE INDICATES ERROR TYPE)
ICODE	ERROR INDICATOR.	

TGETC PICKS UP THE NEXT INPUT CHARACTER, READING A NEW BLOCK WHEN NECESSARY AND RETURNING WITH GOT=3 WHEN END OF FILE , OR END OF A COMPLETE SECTION OF A CONCAT FILE IS FOUND. (SEE SECTION 1.11 OF THIS PET FOR FORMAT OF A CONCAT FILE).

WHEN A CHARACTER IS PICKED UP, TGETC CHECKS FOR THE FOLLOWING SPECIAL CHARACTERS:-
NULL, BACKSPACE, HORIZONTAL TAB, VERTICAL TAB, FORM FEED, CARRIAGE RETURN, RELATIVE HORIZONTAL TAB, HALF LINE FEED AND RELATIVE VERTICAL TAB.
ALL OF THESE EXCEPT THE 'RELATIVE' TABS ARE IGNORED. FOR THE RELATIVE TABS, THE NEXT CHARACTER INDICATES THE NUMBER OF SPACES OR LINES REQUIRED, SO THAT NUMBER IS RETURNED IN THE VARIABLE 'CHAR', WHILE 'GOT' INDICATES THE TAB TYPE.

FOR ORDINARY (NON SPECIAL) CHARACTERS, TGETC CHECKS WHETHER TRANSLATION IS REQUIRED, AND IF SO PUTS THE BCD VALUE INTO 'CHAR' BEFORE RETURNING. IF NO TRANSLATION IS REQUIRED, THE ASCII VALUE (IE THE VALUE AS ON THE INPUT FILE) IS RETURNED IN 'CHAR'.

NOTE THAT NO TEST SITE WAS AVAILABLE WITH ASCII MODE, SO THE 'NO TRANSLATION' OPTION HAS NOT BEEN TESTED.

NOTE AS A TEMPORARY MEASURE, THIS ROUTINE HAS BEEN INCORPORATED INTO SUBROUTINE TRANSM, AS A DUMMY SUBROUTINE, STARTING AT LABEL 500. THIS IS TO AVOID THE OVERHEAD OF FORTRAN SUBROUTINE CALLS.

++

RDCON

THIS ROUTINE IS CALLED BY TRANSM, FIRSTLY TO CHECK WHETHER THE INPUT FILE IS A 'CONCAT' FILE, AND LATER (IF IT IS A 'CONCAT' FILE) TO FIND THE TYPE AND LENGTH OF THE NEXT SECTION. (SEE SECTION 1.11 OF THIS PET FOR THE FORMAT OF A CONCAT FILE).

THERE IS ONE ARGUMENT TO THE ROUTINE, 'WOT' , WHICH IS EITHER =1 TO INDICATE THAT THIS IS THE FIRST CALL FOR A FILE, OR =2 TO INDICATE THAT THE TYPE AND LENGTH OF THE NEXT CONCAT SECTION ARE REQUIRED.

FOR A CALL WITH WOT=1, RDCON READS THE FIRST WORD OF THE FILE TO SEE IF IT CONTAINS :1002, THE CONCAT FILE HEADER. IF IT IS FOUND TO BE A CONCAT FILE, THE LOGICAL VARIABLE CONKAT(LINX) IS SET =.TRUE.. IF IT IS FOUND NOT TO BE A CONCAT FILE, THE FILE IS REWOUND TO THE BEGINNING.

FOR A CALL WITH WOT=2 (OR AFTER FINDING IT IS A CONCAT FILE WHEN WOT=1) THE SECTION DESCRIPTION IS READ TO ASCERTAIN THE TYPE AND LENGTH OF THE NEXT SECTION OF THE FILE.

RDCON IS ALSO USED TO SET UP THE VALUES FOR ITEMS THAT VARY BETWEEN ASCII AND BCD CODE. THESE ARE :-

- THE ECODE USED TO TERMINATE AN OPERATOR MESSAGE (E1 FOR BCD, E4 FOR ASCII)
- THE 'END OF RECORD' (@ CHARACTER)
- THE 'ESCAPE' CHARACTER AND
- THE 'BLANK' CHARACTER.

SIMILARLY, RDCON SETS UP THE 'READ' MESSAGE IN THE APPROPRIATE CODE, TO BE USED IF THE 'AUTOREAD' OPTION IS SPECIFIED. CONTROL IS THEN RETURNED TO TRANSM.

++

********** 200UT RECEIVE MODULE DOCUMENTATION **********

SECTION 5.2

INTRODUCTION

THE RECEIVE MODULE IS CALLED BY THE WORKSTATION TO PROCESS
I) UP TO 'N' BLOCKS OF A RECEIVE FILE, OR
II) TO THE END OF A JOB - IE A LOGICAL FILE, OR
III) TO THE END OF A RECEIVE FILE,
WHICHEVER OCCURS FIRST. IN THIS WAY, THE GENERAL WORKSTATION PROCESSING IS NOT HELD UP FOR TOO LONG BEFORE CONTROL IS RETURNED. THE RECEIVE MODULE RETURNS A FLAG TO THE CALLING ROUTINE TO INDICATE WHETHER OR NOT THERE IS FURTHER RECEIVE PROCESSING TO BE DONE ON THAT LINE (IE. THE LINE DEFINED BY THE VARIABLE 'RLINE' ON ENTRY TO THE RECEIVE MODULE).

THE RECEIVE MODULE IS MADE UP OF THE FOLLOWING ROUTINES :-

NAME	FUNCTION
RECV	MAIN RECEIVE ROUTINE: ORGANISES OPEN/CLOSE/DELETE/SAVE OF RECEIVE AND PRINT FILES: INITIALISES STATISTICS, ETC.
RPUTFL	CALLED BY RECV TO PROCESS RECEIVE FILE TILL END OF FILE, END OF JOB, OR BLOCKCOUNT REACHED.
RGETC	CALLED BY RPUTFL TO GET THE NEXT CHARACTER FROM THE RECEIVE FILE - READING A NEW BLOCK WHEN NECESSARY.
RPUTC	CALLED BY RPUTFL TO PUT A CHARACTER INTO THE OUTPUT (PRINT) BUFFER, AND / OR TO WRITE OUT THE BUFFER WHEN NECESSARY.
EOFDET	CALLED BY RPUTFL TO SCAN EACH PRINT LINE AS IT IS COMPLETED, AND CHECK FOR THE 'END OF JOB FILE' SEQUENCE.

++

RECV

RECV IS ENTERED WITH THE VARIABLE 'RLINE' (IN GENERAL COMMON) INDICATING THE LINE TO BE PROCESSED.

THERE IS ONE ARGUMENT TO RECV :-
TODO LOGICAL VARIABLE, RETURNED TO CALLER AS .TRUE. IF THERE IS MORE RECEIVE PROCESSING TO DO ON THIS LINE.

AFTER INITIALISATION, RECV CHECKS WHETHER THERE IS ALREADY A RECEIVE

FILE (HEREINAFTER REFERRED TO AS AN R FILE) BEING PROCESSED FOR THIS LINE. IF SO, IT REINSTATES THE BUFFER POINTERS FROM THE PREVIOUS CALL IN WHICH THE FILE WAS PROCESSED. IF NOT, IT CHECKS FOR THE EXISTENCE OF AN R FILE ON THAT LINE, AND IF NONE IS FOUND, RETURNS TO THE CALLING ROUTINE. ON FINDING AN R FILE, IT READS THE FIRST BLOCK TO ASCERTAIN THE CODE TRANSLATION REQUIREMENTS. AT THIS STAGE, RECV HAS AN R FILE OPEN, AND IS READY TO READ THE NEXT DATA BLOCK.

THEN IT CHECKS WHETHER IT ALREADY HAS AN OUTPUT FILE OPEN FOR THAT LINE, IF SO, IT REINSTATES THE BUFFER POINTERS, AND IF NOT, IT GETS ONE.

THEN RPUTFL IS CALLED TO DO THE ACTUAL 'RECEIVE PROCESSING'. CONTROL IS RETURNED TO RECV FOR ONE (OR MORE) OF THE FOLLOWING REASONS:-

A) END OF JOB IS FOUND BY EOFDET,
B) END OF R FILE FOUND,
C) N BLOCKS OF THE R FILE HAVE BEEN PROCESSED,
D) AN ERROR WAS ENCOUNTERED DURING RECEIVE PROCESSING.

NB. IT IS POSSIBLE FOR RPUTFL TO RETURN FOR REASONS A) AND B) TOGETHER.

ON RETURN FROM RPUTFL, RECV SAVES THE BUFFER POINTERS, AND THEN CHECKS THE REASON FOR THE RETURN. FOR 'END OF JOB', THE PRINT FILE IS CLOSED, AND STATISTICS OUTPUT IF REQUESTED. FOR 'END OF R FILE', THE R FILE IS CLOSED, AND EITHER DELETED OR CHANGED TO AN 'S' FILE, DEPENDING ON THE 'SAVE' OPTION SELECTED. IF NO END OF FILE DETECTION IS SPECIFIED (EOFKEY=0) THEN THE PRINT FILE IS CLOSED WHEN THE END OF THE R FILE IS FOUND.

IF A REQUEST HAS BEEN MADE TO STOP RECEIVE PROCESSING ON THIS LINE (IE WLINE=0), THEN IT IS STOPPED (BY SETTING WLINE=-1) WHEN

I) END OF R FILE AND END OF JOB COINCIDE, OR
II) END OF R FILE FOUND AND EOFKEY=0.

IF AN ERROR IS FOUND IN RPUTFL (OR ANY OR THE ROUTINES WHICH IT CALLS) THEN RECEIVE PROCESSING IS STOPPED (BY SETTING WLINE=0) REGARDLESS OF THE PREVIOUS STATE OF WLINE.

NB. IF CARD FILES ARE TO BE USED, RECV MUST BE MODIFIED TO CHECK FOR CARD OR PRINT FILES.

++

RPUTFL

RPUTFL BEGINS BY SETTING THE 'ESCAPE', 'END OF LINE' AND E CODE 1 OR 4 FOR BCD OR ASCII, DEPENDING ON THE TRANSLATION MODE FOR THIS LINE.

IT THEN CHECKS WHETHER PART OF AN INPUT BLOCK WAS SAVED AT THE END OF THE PREVIOUS CALL FOR THIS LINE, AND IF SO, RESTORES THAT BLOCK AND THE RELEVANT POINTERS. FOR THE TEST SITE THIS CODE IS NOT REQUIRED, AS IT NEVER ENDS THE JOB IN THE MIDDLE OF A BLOCK.

SIMILARLY, THE CODE FOR 'CARD' RECEIVE FILES IS INCLUDED BUT NOT TESTED.

THE MAIN WORK IN RPUTFL STARTS AT LABEL 100, WHICH IS THE BEGINNING OF AN INPUT RECORD. THUS, FOR PRINT FILES, IT GETS THE FIRST CHARACTER AS A LINE CONTROL CHARACTER, AND CHECKS FOR PAGE THROW, SINGLE LINE FEED, DOUBLE LINE FEED, OR NO LINE FEED AT ALL. (SEE NOTE ON PRINT OUTPUT).

HAVING DEALT WITH PAPER CONTROL, RPUTFL BUILDS UP A LINE OF PRINT, USING RGETC TO GET EACH CHARACTER FROM THE INPUT FILE, AND RPUTC TO STORE EACH CHARACTER FOR OUTPUT. ON GETTING A CHARACTER, RPUTFL CHECKS WHETHER IT IS AN 'ESCAPE' CHARACTER, AND IF NOT, DOES THE REQUIRED CODE TRANSLATION AND STORES THE RESULT. THE CODE TRANSLATION TABLE IS SET UP SUCH THAT ANY CHARACTER OUT OF RANGE WILL BE 'TRANSLATED' AS A QUESTION MARK. WHEN AN 'ESCAPE' IS FOUND, IT INDICATES THAT THE CHARACTER FOLLOWING IT IS:-

A) END OF LINE, OR
B) CARRIAGE RETURN, OR
C) E2 CODE (END OF BLOCK), OR
D) E3 CODE (END OF BLOCK), OR
E) ZERO OR SPACE COMPRESSION.

HAVING FETCHED THE CHARACTER AFTER THE 'ESCAPE' - Q - IT IS CHECKED FOR THE POSSIBILITIES A) THRU D) ABOVE, AND IF NONE OF THOSE IS ASSUMED TO BE AN E) FOR CHARACTER COMPRESSION. IN THIS CASE, A VALUE OF Q > :40 INDICATES (Q .AND. :37) SPACES ARE REQUIRED, A VALUE OF Q < :40 INDICATES (Q .AND. :17) ZEROS ARE REQUIRED.

FOR CASES C) AND D) - IE. END OF BLOCK - THE BLOCK POINTER IS SET BY RPUTFL SUCH THAT A NEW INPUT BLOCK WILL BE READ ON THE NEXT CALL TO RGETC, THEN THE NEXT PRINT LINE IS STARTED (AT LABEL 100).

FOR A) AND B), ROUTINE 'EOFDET' IS CALLED, TO CHECK FOR THE 'END OF JOB FILE' SEQUENCE, AND IF THAT IS NOT FOUND THEN RPUTFL CONTROL GOES TO LABEL 100 TO START THE NEXT LINE.

WHEN END OF JOB IS DETECTED, RPUTC IS CALLED TO OUTPUT THE CURRENT PRINT BUFFER (READY FOR RECV TO CLOSE THE FILE). RPUTFL THEN CAUSES THE NEXT INPUT BLOCK TO BE READ, ON THE ASSUMPTION THAT IT IS A 4 WORD TRAILER BLOCK AFTER A JOB. IF IT IS NOT, THE BLOCK IS SAVED; IF IT IS, THE BLOCK IS IGNORED. CONTROL IS THEN RETURNED TO RECV TO HANDLE THE END OF JOB.

ON EACH CALL TO RGETC THE COMMON VARIABLE 'WHY' INDICATES THE REASON FOR THE RETURN FROM RGETC, AS FOLLOWS :-

WHY =1 = NORMAL CHARACTER RETURNED,
=2 = BLOCK COUNT COMPLETED,
=3 = END OF INPUT FILE (IE R FILE) FOUND,
=4 = ERROR IN RGETC.

FOR A RETURN OF WHY= 2, 3 OR 4 RPUTFL RETURNS CONTROL DIRECTLY TO RECV. WHY IS ALSO USED BY RPUTFL TO INDICATED TO RECV THE REASON FOR RETURN AS FOLLOWS :-

RPUTFL SETS WHY= 3 WHEN IT FINDS THE END OF FILE AFTER A 4 WORD 'JOB TRAILER' BLOCK.
" " " = 4 ON LOCATING AN ERROR - EG. ILLEGAL CHARACTER ON R FILE.
" " " = 5 WHEN END OF JOB FILE IS FOUND BY EOFDET.

++

RGETC

RGETC IS CALLED BY RPUTFL TO GET THE NEXT CHARACTER FROM THE INPUT FILE AND LEAVE IT IN THE COMMON VARIABLE 'CHAR'. RGETC KEEPS BUFFER AND BYTE POINTERS, AND BUFFER SIZE, SO WILL READ A NEW BLOCK WHEN THE BUFFER POINTER (INPTR) EQUALS THE BUFFER SIZE (INSIZE).

RGETC ALSO MAINTAINS STATISTICS OF THE NUMBER OF CHARACTERS USED FROM THE INPUT FILE (IE. THE R FILE) AND THE NUMBER OF BLOCKS READ . HOWEVER, IT SHOULD BE NOTED THAT RECV CLEARS THE STATISTICS COUNTS AT THE BEGINNING OF EACH LOGICAL JOB FILE, SO THAT THEY ARE PER JOB FILE, RATHER THAN PER TRANSMISSION STATISTICS.

RGETC SETS THE COMMON VARIABLE 'WHY' TO INDICATE THE REASON FOR THE RETURN TO RPUTFL, AS FOLLOWS :-
WHY = 1 = NORMAL CHARACTER RETURNED IN 'CHAR',
= 2 = BLOCK COUNT COMPLETE FOR THIS CALL TO RECV,
= 3 = END OF R FILE FOUND,
= 4 = ERROR IN RGETC.

NOTE AS A TEMPORARY MEASURE, THIS ROUTINE HAS BEEN INCORPORATED INTO SUBROUTINE RPUTFL, AS A DUMMY SUBROUTINE, STARTING AT LABEL 500. THIS IS TO AVOID THE OVERHEAD OF FORTRAN SUBROUTINE CALLS.

++

RPUTC

THE MAIN FUNCTION OF RPUTC IS TO STORE THE CHARACTER FROM COMMON VARIABLE 'CHAR' IN THE OUTPUT BUFFER, AND TO WRITE THE BUFFER OUT WHEN FULL.

WHEN RPUTC IS CALLED WITH LOGICAL VARIABLE 'WROUT' = .TRUE. IT IS A SPECIFIC REQUEST TO WRITE OUT THE CURRENT BUFFER, WITHOUT STORING 'CHAR'.

THERE ARE THREE OTHER FUNCTIONS PERFORMED BY RPUTC :-

I) TO MAINTAIN COUNTS OF NUMBER OF PRINT LINES OUTPUT, AND THE MAXIMUM LINE LENGTH FOUND,

II) TO CHECK WHEN A 'NEWLINE' CHARACTER IS BEING STORED AND ENSURE THAT IT IS EITHER IN THE RIGHT HAND BYTE OR IS FOLLOWED BY A 'NULL' (SEE NOTE ON PRINT OUTPUT).

III) TO STORE THE CHARACTER IN 'CHAR' IN THE BUFFER KLINE, ONE CHARACTER PER WORD (IN THE RIGHT HAND BYTE) FOR USE BY EOFDET WHEN THE LINE IS COMPLETE.

NOTE AS A TEMPORARY MEASURE, THIS ROUTINE HAS BEEN INCORPORATED INTO SUBROUTINE RPUTFL, AS A DUMMY SUBROUTINE, STARTING AT LABEL 700. THIS IS TO AVOID THE OVERHEAD OF FORTRAN SUBROUTINE CALLS.

++

EOFDET

EOFDET IS THE END OF JOB DETECTION ROUTINE. (THE RELEASED VERSION OF 200UT ONLY INCLUDES END OF JOB DETECTION FOR OUR TEST SITE).

THE EOFKEY SITE DEFINITION PARAMETER SPECIFIES THE END OF JOB (OTHERWISE KNOWN AS END OF FILE) DETECTION, AS FOLLOWS :-

```
  EOFKEY = 0  = NO END OF FILE DETECTION,
         = 1  = SEARCH FOR TEST SITE STYLE END OF JOB FILE,
                (RELEASED VERSION)
         > 1  = OTHERS AS ADDED ( IF ANY).
```

THE EOFDET ROUTINE IS CALLED BY RPUTFL AT THE END OF EACH PRINT LINE, TO CHECK THAT LINE FOR THE END OF JOB FILE SEQUENCE.

THE LINE OF CHARACTERS IS STORED IN THE COMMON ARRAY KLINE, WITH COMMON VARIABLE KPTR = THE NUMBER OF CHARACTERS IN THE LINE, EXCLUDING THE NEWLINE OR CARRIAGE RETURN CHARACTER.

TO ADD FURTHER STYLES OF 'END OF JOB FILE' DETECTION, IT IS RECOMMENDED THAT THE ANALYST STUDIES THE TEST SITE EXAMPLE AND WORKS FROM THAT. THE CHECK ON EOFKEY AT THE BEGINNING OF THE ROUTINE WILL NEED TO BE MODIFIED TO ACCOUNT FOR ADDED MODES OF EOFDET.

++

PRINT OUTPUT

THERE ARE TWO POINTS THAT SHOULD BE NOTED ABOUT GENERATING PRINT FILES :-

I) IF A 'NEWLINE' CHARACTER IS IN THE LEFT HAND BYTE OF A WORD, THE RIGHT HAND BYTE WILL BE IGNORED. FOR THIS REASON, RPUTC CHECKS WHETHER A 'NEWLINE' CHARACTER IS BEING STORED IN THE LEFT BYTE,

AND IF SO, ADDS A NULL CHARACTER IN THE RIGHT BYTE.

II) 'FORM FEED' (OTHERWISE KNOWN AS 'PAGE THROW') CHARACTERS MUST BE PRECEDED BY A 'NEWLINE' CHARACTER IN THE PREVIOUS WORD, AND FOLLOWED BY A 'NEWLINE' IN THE SAME WORD.

IE			OR		
	NL	NUL		...	NL
	FF	NL		FF	NL

++

FORMAT OF A RECEIVE FILE FOR 200UT

THE FIRST BLOCK IN EACH FILE IS A 4 WORD BLOCK AS FOLLOWS :-

WORD 1 = 0
WORD 2 = 0 FOR BCD FILE, NON ZERO FOR ASCII.
WORD 3 = UNDEFINED
WORD 4 = N = NUMBER OF WORDS IN NEXT BLOCK.

THE LAST WORD IN EACH BLOCK IS A COUNT OF THE NUMBER OF WORDS IN THE NEXT BLOCK, A 'ZERO' COUNT INDICATING 'END OF FILE'.

FOR FILES RECEIVED FROM THE TEST SITE, EACH JOB IS FOLLOWED BY A 4 WORD TRAILER BLOCK, WHICH CAN BE IGNORED (IE. THERE IS NO DATA IN IT).

WITHIN EACH DATA BLOCK THERE ARE AN INTEGRAL NUMBER OF PRINT LINES, IN THAT NO LINE EVER EXTENDS ACROSS TWO DATA BLOCKS. FOR THE TEST SITE, NO BLOCK EVER CONTAINS LINES FROM MORE THAN ONE LOGICAL JOB FILE, IN OTHER WORDS WHEN 'END OF LOGICAL JOB FILE' IS FOUND, THE REMAINDER OF THE BLOCK CAN BE IGNORED.

WITHIN A BLOCK, THE DATA IS ALWAYS TERMINATED BY THE SEQUENCE 'ESCAPE' 'E2' 'ETX'. (THIS IS FOLLOWED BY THE WORD GIVING THE SIZE OF THE NEXT BLOCK).

++

FORMAT OF A PRINT LINE

EACH PRINT LINE APPEARS IN THE RECEIVE FILE AS A MINIMUM OF THREE CHARACTERS, NAMELY

1) PAPER CONTROL CHARACTER
   0 - 136 PRINTABLE CHARACTERS
2) ESCAPE
3) LINE TERMINATOR

THE 'PRINTABLE CHARACTERS' MAY CONTAIN ZERO OR SPACE COMPRESSION, AS DESCRIBED IN SECTION 5.2.3.

THE PAPER CONTROL CHARACTERS ARE THE STANDARD FORTRAN ONES, AS FOLLOWS

CHARACTER	BCD CODE	ASCII CODE	MEANING
1	:301	:261	PAGE THROW
SPACE	:320	:240	SINGLE LINE FEED
0	:112	:260	DOUBLE LINE FEED
+	:260	:253	NO LINE FEED

ANY OTHER CHARACTERS FOR PAPER CONTROL ARE TREATED AS FOR 'SPACE', I.E. A SINGLE LINE FEED IS PROVIDED.

THE LINE TERMINATOR CHARACTER IS PRECEEDED BY AN 'ESCAPE', AND MAY BE AN 'END OF LINE' OR 'CARRIAGE RETURN'.

THE CODES ARE AS FOLLOWS:-

	BCD	ASCII
ESCAPE	:076	:233
END OF LINE	:320	:301
CARRIAGE RETURN	:301	:215

NOTE THAT THE PARITY BIT WILL BE MASKED OFF BY SUBROUTINE RGETC, SO THAT FOR EXAMPLE, RPUTFL WILL RECEIVE A BCD 'END OF LINE' CHARACTER AS :120 ALTHOUGH IT APPEARS IN THE RECEIVE FILE AS :320.

SECTION 6

6.1 2780 TRANSMIT MODULE

INTRODUCTION

THE TRANSMIT ROUTINES ARE CALLED BY THE WORKSTATION TO TRANSLATE A FILE INTO A TRANSMIT FILE THAT IS DIGESTABLE BY THE SYMBIONT.

TO MAKE THE PROGRAM AS MAINTAINABLE AS POSSIBLE, SEVERAL MEASURES HAVE BEEN TAKEN.

A) THE CODE HAS BEEN WRITTEN AS MODULARLY AS POSSIBLE.

B) THE FRAMING CHARACTERS APPLICABLE FOR EACH BLOCK OF THE TRANSMIT FILE ARE LOOKED UP AT THE BEGINNING OF EACH RUN FROM TABLES.

THE TRANSMIT MODULE IS MADE OF THE FOLLOWING ROUTINES:-

TRANSM - MAIN ROUTINE FOR TRANSMIT PROCESSING. BASICALLY OPENS AN INPUT FILE AND A DUMMY TRANSMIT FILE, CALLS THE APPROPRIATE PROCESSING ROUTINE.

TPUTFL - OUTPUTS A BLOCK INTO THE OUTPUT FILE PUTTING IN THE CORRECT FRAMING CHARACTERS AT THE END OF THE BLOCK.

TBLK - WRITES OUT A BLOCK OF DATA INTO AN OUTPUT ARRAY.

TBLKF - WRITES OUT A BLOCK OF A FORTRAN FILE INTO AN OUTPUT FILE.

TPUTC - OUTPUTS A CHARACTER INTO THE OUTPUT ARRAY.

TPUTFC - OUTPUTS A FORTRAN FORMS CONTROL CHARACTER INTO THE OUTPUT ARRAY.

TPUTL - OUTPUTS THE CONTENTS OF AN ARRAY INTO THE OUTPUT BUFFER.

TGETBF - GETS THE NEXT BUFFER OF AN INPUT FILE FOR A NORMAL SOURCE FILE.

TGETBB - GETS THE NEXT BUFFER OF A CONCAT FILE.

TGETL - GETS THE NEXT LINE OF AN INPUT FILE. (LOOKS FOR THE NEXT NEW LINE CHARACTER).

TGETLB - GETS THE THE NEXT LINE FOR A BINARY FILE. - IE GETS THE NEXT 80 CHARACTERS OF DATA.

TGETC - GETS THE NEXT CHARACTER OF THE INPUT FILE.

RJOPT - LOOKS AT THE CONFIGURATION FOR THE SITE DEFINED AND WORKS OUT THE INDEX INTO THE TABLES OF FRAMING CHARACTERS.

TINIT - INITIALISES VARIABLES FOR THE NEXT INPUT FILE.

HEADER

THIS ROUTINE OUTPUTS THE HEADER FOR TRANMIT FILES. IT IS ENCODED AS FOLLOWS:

A) WORD 1 IS THE PASSWORD, CURRENTLY SET TO ZERO.
B) WORD 2 ENCODES THE TRANSMIT MODE: BIT 14 SET - DOUBLE BLOCKING
BIT 15 SET - LINE CODE IS EBCDIC
BIT 16 SET - NONTRANSPARENT DATA

TBLK

THIS CODE IS IN TWO SECTIONS:-

1) INITIALISATION CODE
2) NORMAL CODE THAT OUTPUTS A BLOCK.

THE INITIALISATION CODE IS THE CODE USED TO OUTPUT DEVICE SELECTION. WE NORMALLY DO ONE OF THE VARIOUS THINGS DEPENDING ON WHAT MODE WE ARE IN.

A) OUTPUT DEVICE SELECTION AS A RECORD WITH NOTHING ELSE IN IT.

B) OUTPUT DEVICE SELECTION AS VFC WITH DATA IN THE RECORD.

C) OUTPUT DEVICE SELECTION, AND END THE BLOCK.

IN EACH CASE A SEPARATE CHARACTER SEQUENCE IS OUTPUT, THE DATA BEING CONTAINED IN OPT5, OPT6, ETC.
FOR THESE OPTIONS, AFTER OUTPUTTING THE RECORD START CHARACTERS, WE SHOULD JUMP TO THE FOLLOWING LABELS:

A) 90
B) 300
C) 500

THE SECOND SECTION OF CODE DOES THE REAL WORK. THE IDEA IS TO FILL UP THE BLOCK WITH AS MUCH DATA AS POSSIBLE. SEVERAL RESTRICTIONS APPLY:-

A) SPACE FOR DATA IS LIMITED TO A NORMAL 400 BYTES (INCLUDES FRAMING, CRC CHARACTERS), UNLESS THE 600 BYTE OPTION IS USED.

B) THERE IS A MAXIMUM NUMBER OF RECORDS ALLOWED IN A BLOCK.

C) WE MAY HAVE REACHED THE END OF OUR INPUT FILE.

WE ENTER THE PROCESSING CODE WITH THE NEXT CHUNK OF DATA TO BE OUTPUT IN OUR TEMPORARY BUFFER. THE SEQUENCE OF ACTIONS IS AS FOLLOWS:-

A) OUTPUT ANY START OF RECORD CHARACTERS THAT NEED TO BE OUTPUT E.G., 'DLE STX'. NOTE THAT IF THIS IS THE CASE, THAT THE SYMBIONT OUTPUTS AN 'STX' AT THE START OF EACH BLOCK, SO WE HAVE TO SKIP THIS 'STX' IF WE TOLD TO OUTPUT ONE AT THE BEGINNING OF EACH RECORD.

B) STANDARD VFC IS THEN OUTPUT, EG, NEW LINE, FOR HOST MODE FILES.

C) OUTPUT OUR LINE.

D) DECIDE WHETHER WE SHOULD FINISH OUR BLOCK. IF SO, RETURN, OTHERWISE OUTPUT THE RECORD END CHARACTERS ('ITB') AND START THE LOOP AGAIN.

NOTE THAT AS WE OUTPUT MORE RECORDS INTO A BLOCK MORE CRC CHARACTERS ARE GOING TO BE NEEDED. THE OUTPUT BLOCK SIZE IS DECREMENTED AFTER EACH RECORD IS OUTPUT (2 CHARACTERS PER RECORD).

TBLKF

THIS ROUTINE IS SIMILAR TO TBLK EXCEPT THAT IT PROCESSES FORTRAN TYPE FILES. CONSIDER THE DIFFERENCE BETWEEN 2780, PRIME AND FORTRAN TYPE FORMATS:

2780 HAS VFC AT THE BEGINNING OF A LINE, BUT IS ACTIONED AT THE END OF THE LINE.

PRIME HAS VFC AT THE END OF A LINE AND IS ACTIONED THEN.

FORTRAN HAS VFC AT THE BEGINNING OF A LINE, AND IS ACTIONED BEFORE THE LINE.

WITH 2780 AND PRIME FORMATS, VFC IS ACTIONED AT THE END OF A LINE, BUT IS STORED IN DIFFERENT PLACES, BUT STILL WITHIN THE SAME RECORD. WITH CONVERSION FROM FORTRAN TO 2780 THOUGH, WE NEED TO GET VFC FOR A PARTICULAR RECORD FROM THE FOLLOWING LINE.

NOTE THAT 2780 DOES NOT IMPLEMENT OVERWRITING OF LINES - SO IF YOU REALLY WANT TO SEND SUCH A FILE, SEND IT AS BINARY.

TGETBB

GETS THE NEXT INPUT BUFFER FOR CONCAT FILES. CONCAT FILES DIFFER FROM NORMAL SOURCE FILES IN THAT THEY ARE ENCODED IN THE FOLLOWING WAY:-

A) THE FILE STARTS WITH A WORD :002001:
B) THE FILE IS MADE UP OF 'SECTIONS'. EACH SECTION IS A PART OF CONCATTED DATA, EACH WITH A HEADER OF WHICH:

WORD 1 IS THE TYPE - BINARY OR TEXT
WORD 2-3 IS THE LENGTH (DOUBLE LENGTH INTEGER)

TO UNDERSTAND HOW THE ROUTINE WORKS, SOME OF THE CONTROL VARIABLES WILL BE DESCRIBED.

IRNW - NUMBER OF WORDS IN THE CURRENT INPUT BUFFER.

THESE MAY OR MAY NOT BE IN THE SAME 'SECTION' AS THE PRESENT ONE.

CNMWDS - NUMBER OF WORDS IN THIS BUFFER IN THE SAME SECTION.

TNMWDS - NUMBER OF WORDS TO PROCESS AFTER PROCESSING THE 'CNMWDS' IN THIS BUFFER.

WE GET CALLED EVERY TIME WE WANT ANOTHER BUFFER WE CAN RETURN WITH:

A) RETCOD = 0
CNMWDS SET
TNMWDS SET

THIS MEANS THAT WE HAVE GOT THE NEXT BUFFER IN THE NORMAL WAY.

B) RETCOD = XM$EOS
TNMWDS SET

THIS MEANS THAT WE HAVE REACHED THE END OF THE CURRENT SECTION. WE GIVE A CHANCE FOR THE CALLING ROUTINES TO SORT THEMSELVES OUT SINCE OUR MODE MAY HAVE BEEN CHANGED - EITHER BINARY OR NON BINARY ALLOWED. ALL WE HAVE DONE, IS TO READ IN THE NEXT SECTION HEADER, AND NOT RETURNED THE NEXT LINE.

C) RETCOD = XM$EOF

WE HAVE REACHED THE END OF OUR INPUT FILE.

TGETC

ROUTINE GETS THE NEXT CHARACTER, CHECKS TO SEE IF IT IS IN THE INPUT BUFFER, AND IF NOT, READS IT OFF DISC.

TGETL

THIS ROUTINE GETS THE NEXT LINE OF DATA. IT TRANSLATES IT TO EBEDIC, IF NECESSARY, EXPANDS COMPRESSION, AND PADS OUT THE LINE TO THE DEFAULT LENGTH WITH THE PAD CHARACTER, IF NEEDED. ITS FUNCTIONS ARE GOING TO BE DESCRIBED IN MORE DETAIL.

A) THE START OF A LINE ALWAYS STARTS ON A WORD BOUNDARY. IF WE ARE ON THE RHB, THEN WE MUST LOOK AT THE FOLLOWING BYTE. WE LOOK FOR THE NEW LINE CHARACTER 'EOLCHR' TO FIND THE END OF LINE.

B) COMPRESSED SPACES NEED TO BE EXPANDED. THE FORMAT OF COMPRESSION IS 'RHT' FOLLOWED BY NUMBER OF SPACES COMPRESSED.

C) TRANSLATION AND CHARACTER VALIDITY IS CHECKED BY LOOKING UP 'TABLE'. NEGATIVE ENTRIES DENOTE ILLEGAL CHARACTERS - A FATAL ERROR.

D) IN CERTAIN MODES, WE NEED TO PAD A LINE OUT TO A FIXED LENGTH. WE DO THIS WITH THE PAD CHARACTER - USUALLY A SPACE OF THE APPROPRIATE CODE.

TGETLB

ROUTINE GETS THE NEXT 80 CHARACTERS OF AN INPUT FILE, UNPACKING THEM INTO ARRAY 'TEMP'.

THIS ROUTINE DOES NOT BOTHER TO SEE WHAT CHARACTERS IT GETS - AND SO CAN ONLY BE CALLED FOR FILES THAT WILL BE TRANSMITTED AS TRANSPARENT EBCDIC. FOR THE LAST RECORD OF THE FILE, WE PAD OUT TO 80 CHARACTERS WITH OUR PAD CHARACTER (A SPACE).

TPUTC

OUTPUTS A CHARTACTER AT A TIME INTO THE NEXT FREE LOCATION OF THE OUTPUT BUFFER.

TPUTFC

USED FOR OUTPUTTING VFC CHARACTERS WHEN PROCESSING FORTRAN FILE. IT GETS THE FIRST CHARACTER OF ARRAY TEMP, TRANSLATES IT TO THE 2780 EQUIVALENT AND INSERTS IT IN THE BYTE ADDRESS GIVEN BY PTR.

TPUTFL

THIS ROUTINE APPENDS BLOCK END, AND FILE END FRAMING CHARACTERS. IT THEN WRITES THE BLOCK OFF TO DISC. THE OUTPUT BUFFER POINTERS ARE THEN RE-INITIALISED FOR THE NEXT BLOCK, SO THAT WE CAN START WRITING THE NEXT BLOCK AT THE START OF THE OUTPUT BUFFER.

TPUTL

OUTPUTS A LINE OF DATA INTO THE OUTPUT BUFFER. THE DATA IS PASSED IN AS A PARAMETER, AND IS UNPACKED. THE ADVANTAGE OF USING THIS ROUTINE IS THAT IT SPEEDS UP PROCESSING, BY REDUCING THE NUMBER OF SUBROUTINE CALLS, BY DEALING IN LINES AND NOT CHARACTERS.

TRANSM

AT THE MOMENT, THIS ROUTINE HAS CERTAIN HOOKS IN IT SO THAT IT WILL HANDLE A QUOTE OF BLOCKS TO BE TRANSLATED INTO THE OUTPUT FILE - IN A SIMILAR METHOD TO RECEIVE PROCESSING.

I.E. WE LOOK TO SEE IF WE HAD AN INPUT/OUTPUT FILE OPEN FROM THE LAST TIME ROUND, IN CASE WE HAVE NOT FINISHED PROCESSING THE LAST FILE.

THIS ROUTINE DOES MOST OF THE FILE MANAGEMENT REQUIRED, AS WELL AS SEVERAL OTHER THINGS. IT:-

A) OPENS THE INPUT FILE
B) OPENS A DUMMY OUTPUT FILE
C) INITIALISES THE STATS VARIABLES AND POINTERS
D) READS IN THE FIRST BUFFER OF DATA, SO THAT WE CAN TELL WHETHER THIS IS A CONCAT FILE
E) PASSES CONTROL TO TPUTFL TO PROCESS THE FILE
F) CLOSES THE FILES OPENED.

6.2 2780 RECEIVE MODULE

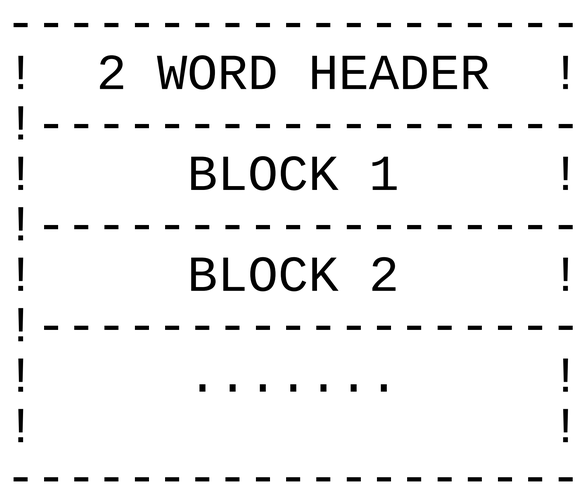
INTRODUCTION

THESE ROUTINES ARE CALLED BY THE WORKSTATION TO TRANSLATE RECEIVE FILES PRODUCED BY THE SYMBIONT, INTO PRINT, PUNCH OR BINARY FILES. EACH TIME WE ENTER THESE ROUTINES, WE PROCESS A MAXIMUM OF A FIXED NUMBER OF BLOCKS, AND THEN RETURN CONTROL BACK TO THE WORKSTATION. THIS PREVENTS US FROM LOCKING OUT THE WORKSTATION WITH LONG RECEIVE FILES.

A BRIEF DESCRIPTION OF THE ROUTINES IS AS FOLLOWS.

RECV	MAIN DRIVING ROUTINE FOR RECEIVE PROCESSING. IT LOOKS TO SEE IF ANYTHING IS TO BE DONE, AND IF SO WILL DO THE NECESSARY FILE MANAGEMENT, AND INITIATE THE RECEIVE FILE TRANSLATION.
RPUTFB	PROCESSES BINARY FILES. NO TRANSLATION IS NEEDED - ONLY THE EXTRACTION OF DATA FROM THE RECEIVE FILE DATA STRUCTURE.
RPUTFL	HANDLES PRINT, PUNCH AND CARD FILES. THE ROUTINE GETS A RECORD AT A TIME, AND OUTPUTS IT, PUTTING IN THE REQUIRED VFC CHARACTERS.
RGETL	ACCESSES A RECEIVED BLOCK, AND GETS THE NEXT RECORD, UNPACKING IT INTO A TEMPORARY ARRAY.
RGETC	GETS THE NEXT CHARACTER FROM THE RECEIVE FILE.
RGETBF	READS THE NEXT BLOCK OF THE RECEIVE FILE OFF DISC AND PUTS IT IN THE INPUT BUFFER.
RPUTC	OUTPUTS A CHARACTER IN THE NEXT FREE LOCATION OF THE OUTPUT BUFFER.
HTREC	PROCESSES A HORIZONTAL TAB RECORD. IT TRANSLATES IT INTO AN INTERNAL FORMAT.
RVFC	READS A VFC FILE INTO ARRAY VFC.

RECEIVE FILE FORMAT



A COMPLETE RECEIVE FILE COMPRISES A TWO WORD HEADER, FOLLOWED BY ONE OR MORE RECEIVE BLOCKS. THE HEADER FORMAT IS AS FOLLOWS.

WORD 1 BIT 15 SET - CODE ASCII
BIT 16 SET - NON TRANSPARENT
BIT 14 SET - DOUBLE BLOCKING

WORD 2 PASSWORD - USUALLY BLANK

EACH RECEIVE FILE MAY CONTAIN ONE OR MORE BLOCKS.

RECEIVE BLOCK FORMAT

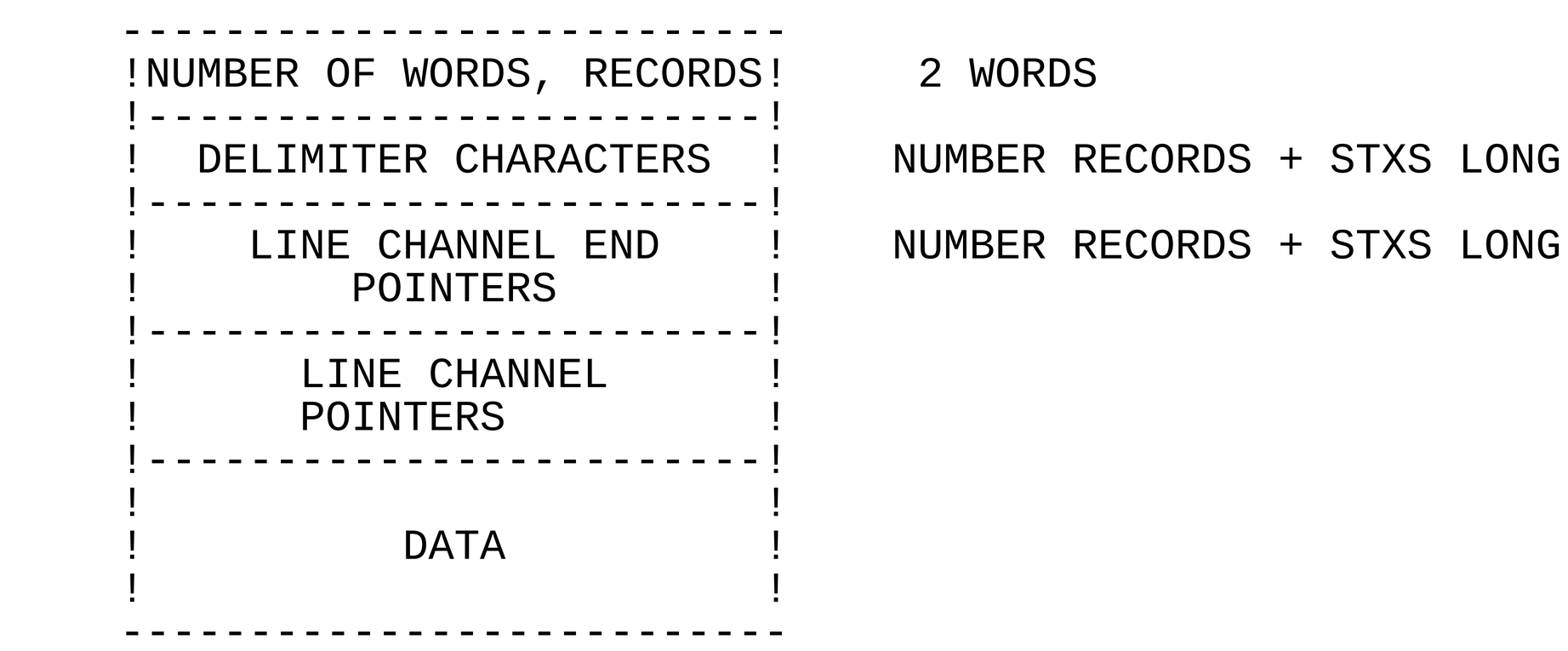
THE RECEIVE BLOCK COMPRISES:

A) A COUNT OF THE NUMBER OF WORDS, RECORDS IN THE BLOCK
B) LIST OF THE LAST CHARACTER OF DATA IN EACH CHANNEL
C) CHANNEL END POINTERS
D) DMC CHANNEL POINTERS
E) A COPY OF THE DMC CHANNELS USED TO RECEIVE THE BLOCK.

OUR ACTUAL RECORDS ARE CONTAINED IN THE DMC CHANNELS. TO ACCESS THEM WE MUST DO THE FOLLOWING:

A) GET THE LAST CHARACTER OF A RECORD FROM THE LIST OF DELIMITER CHARACTERS.
B) LOOK UP THE DMC CHANNEL END POINTER. WE KNOW WHICH IS THE START CHANNEL- THE NEXT DMC CHANNEL THAT WE HAVENT USED YET.
C) PICK UP THE POINTERS TO THE ACTUAL DATA USING THE DMC CHANNEL POINTERS.
D) OUR RECORD IS NOW BOUNDED BY THE START OF THE CHANNELS, AND THE OCCURRENCE OF THE DELIMITER CHARACTER. THE DATA LEFT IN REMAINDER OF THE CHANNEL IS RUBBISH LEFT IN THE BUFFER, AND IS NOT USED.

RECEIVE BLOCK FORMAT



A) NUMBER OF WORDS/RECORDS

FIRST WORD -> NUMBER WORDS IN BLOCK
SECOND WORD -> NUMBER RECORDS IN BLOCK.

B) DELIMITER CHARACTERS

THERE IS ONE WORD ENTRY FOR:

I) EACH INPUT RECORD
II) EACH STX FRAMING CHARACTER

NOTE THAT NON-TRANSPARENT FILES HAVE ONLY 1 STX AT THE BEGINNING OF EACH BLOCK. FOR TRANSPARENT BLOCKS, WE HAVE A 'DLE STX' AT THE BEGINNING OF EACH RECORD.

THE DELIMITING CHARACTER IS CONTAINED IN THE LHB OF EACH WORD, UNLESS WE ARE IN TRANSPARENT MODE, IN WHICH CASE THE LHB IS 'DLE' AND THE RHB IS THE DELIMITING CHARACTER.

C) DMC END POINTERS

THESE POINT TO THE END DMC CHANNEL POINTER PAIR FOR EACH RECORD/STX CHANNEL. WE ALWAYS ASSUME THAT WE USE DMC CHANNELS SEQUENTIALLY - I.E. THE START OF THE NEXT RECORD IS THE CHANNEL AFTER THE END OF THE LAST RECORD. SO, WE ASSUME THE FIRST RECORD STARTS IN CHANNEL ZERO.

D) DMC CHANNEL POINTERS

THERE ARE TWO POINTERS FOR EACH DMC - THEY ARE THE START AND END WORD RECORD POINTERS. THESE POINTERS ARE TAKEN FROM THE START OF THE RECEIVE BLOCK.

HTREC

TO ENCODE THE HT ARRAY WE START FROM THE END OF THE RECORD WORKING THROUGH TO THE START. FOR EACH POSITION THAT NO HT HAS BEEN COME ACROSS, SET ELEMENT OF HT ARRAY TO 0. WHEN WE REACH THE FIRST 'HT' CHAR SET FLAG TO SHOW WE NEED TO INCREMENT THE SPACE COUNTER. FOR ANY POSITION FROM NOW ON, WE WILL NEED TO OUTPUT SPACE TO GET TO THIS POSITION. WHENEVER WE REACH ANOTHER HT CHARACTER WE NEED TO RESET THIS COUNTER.

INITIALIZE THE PART OF THE HT ARRAY THAT IS NOT GOING TO BE SET BY TRANSLATING THE HORIZONTAL TAB RECORD - THE HT RECORD IS SHORTER THAN THE HT ARRAY

START FROM END OUTPUT IN EACH POSITION NUMBER OF SPACES TO GET TO NEXT HT CHARACTER (COUNT).

WHEN SET = .TRUE. WE HAVE HAD A HT CHARACTER.

RECV

THIS ROUTINE IS THE MAIN DRIVER FOR RECEIVE PROCESSING. IT MANAGES INPUT AND OUTPUT FILES, AND LOOKS TO SEE IF THERE ARE ANY MORE FILES TO PROCESS.

WE CAN SUBDIVIDE THIS ROUTINE INTO SEVERAL SECTIONS:

A) GENERAL HOUSEKEEPING - WORK OUT WHETHER WE WERE DOING ANYTHING OR WERE NOTIFIED TO CLOSE.
B) OPENING AN INPUT FILE AND INITIALISING ASSOCIATED VARIABLES.
C) DECIDING WHAT SORT OF FILE OUR INPUT FILE IS CARD, PUNCH OR PRINT.
D) OPENING AN OUTPUT FILE AND INITIALISING ASSOCIATED VARIABLES.
E) HOUSEKEEPING AT THE END OF PROCESSING - FLUSHING OUTPUT BUFFERS, UPDATING POINTERS.

A) WHENEVER WE PROCESS A FILE WE PROCESS A MAXIMUM OF 'CNMBLK' BLOCKS AT A TIME. IF WE HAVE MORE TO DO, WE SET FLAG 'TODO' AND RETURN TO ALLOW THE WORKSTATION TO SEE IF IT HAS ANYTHING ELSE TO DO. WE WILL RE-ENTER THE ROUTINE WITH OPEN INPUT AND OUTPUT FILES, SO WE MUST RESUME THIS PROCESSING FOR ANOTHER QUOTA OF BLOCKS.

WE CAN GET CALLED WITH FLAG 'WLINE=0' WHICH TELLS US THAT THE NEXT TIME WE HAVE NOTHING TO DO, TO STOP PROCESSING. SO WHENEVER WE ENTER THIS ROUTINE WE DO THE FOLLOWING:-

1) SEE IF WE WERE DOING ANYTHING
2) SEE IF WE WERE NOTIFIED TO CLOSE
3) LOOK FOR THE NEXT RECEIVE FILE

B) WHENEVER WE OPEN AN INPUT FILE, WE MUST FIRST CALL THE STATS INITIALISATION ROUTINE. WE THEN OPEN THE INPUT FILE AND READ IN:

1) HEADER
2) NUMBER OF WORDS, RECORDS FOR THE FIRST BLOCK

THE HEADER TELLS US THE CODE OF THE FILE - ASCII OR EBCDIC, AND ITS TRANSPARENCY.

WE THEN INITIALISE SEVERAL POINTERS/VARIABLES SPECIFIC TO THIS LINE, E.G. HORIZONTAL TAB ARRAY (HT), LINE COUNT (LC). WE THEN INITIALISE VARIALES THAT ARE INITIALISED EVERY TIME WE ENTER THIS ROUTINE WITH THIS INPUT FILE. THESE ARE VARIABLES WHICH ARE COMMON TO ALL LINES. THE MOST IMPORTANT OF THESE IS 'CNMBLK' - THE NUMBER OF BLOCKS TO BE PROCESSED ON THIS CALL TO RECEIVE. WE THEN INITIALISE ALL THE STATS VARIABLES FOR THIS FILE.

C) THE CODE ROUND LABEL 130 WORKS OUT WHAT DEVICE THIS FILE IS MEANT TO BE ROUTED TO. TO GET THE DEVICE SELECTION CODE, WE GET THE FIRST RECORD AND MAKE THE FOLLOWING DECISIONS:

1) IF WE ARE HOST, THE FILE MUST BE A CARD FILE.
2) IF FIRST CHARACTER IS 'ESC', WE HAVE DEVICE SELECTION.

3) IF WE HAVE DEVICE SELECTION, AND SECOND BYTE IS A '4' (:364) WE HAVE A PUNCH FILE.
4) IF SECOND BYTE IS NOT A '4' THEN WE HAVE A PRINT FILE.
5) IF NO DEVICE SELECTION CODE IS NOT PRESENT, THEN ASSUME A PRINT FILE IF ASCII OR A PUNCH FILE IF EBCDIC, UNLESS WE HAPPEN TO BE HOST, IN WHICH CASE IT IS PUNCH.

BECAUSE OF THE MANY IMPLEMENTATIONS OF THE 2780 POINT (5) OF THE ABOVE WILL NEED CHANGING FOR VARIOUS SITES.

D) HAVING DECIDED WHAT TYPE OF FILE WE NEED TO OPEN, WE OPEN OUR OUTPUT FILE AND INITIALISE ITS ASSOCIATED VARIABLES. THE RECEIVE FILE WILL NOW BE TRANSLATED BY CALLING RPUTFL FOR A NON BINARY FILE, AND RPUTFB FOR A BINARY FILE.

RGETBF

ROUTINE READS IN A BLOCK OF DATA INTO THE INPUT BUFFER AT ONE TIME, PLUS THE NUMBER OF WORDS IN THE NEXT BLOCK - I.E. THE FIRST WORD OF THE NEXT BLOCK. THIS ALLOWS US TO KNOW THE AMOUNT OF DATA TO READ IN THE NEXT TIME ROUND, I.E. IN THE NEXT BLOCK.

THERE ARE SEVERAL VARIABLES THAT REALLY CONTROL THE FLOW IN THIS SUBROUTINE. THEY ARE:

CNMWDS, CNMRECS - THE NUMBER OF WORDS, RECORDS IN THE CURRENT BLOCK.

NNMWDS, NNMREC - THE NUMBER OF WORDS, RECORDS IN THE NEXT BLOCK.

WHENEVER WE GET A NEW BLOCK WE NEED TO COPY NNMWDS, NNMREC INTO CNMWDS CNMRECS. THWN NNMWDS IS ZERO, WE KNOW WE HAVE REACHED THE END OF OUR FILE.

RGETL

THIS ROUTINE ACCESSES A RECEIVE BLOCK, AND FROM IT EXTRACTS A RECORD OF DATA AND UNPACKS IT INTO AN ARRAY CALLED TEMP.

DESCRIPTION

PRINCIPLE CONTROL VARIABLES IN THIS ROUTINE ARE:

CREC - THE CURRENT RECORD NUMBER
CBLK - THE NUMBER OF BLOCK BEING PROCESSED ON THIS CALL TO RECEIVE PROCESSING.

THE FIRST PART OF CODE CHECKS WHETHER A RECORD IS IN THE CURRENT BLOCK, AND IF NOT WHETHER WE CAN GET ANOTHER BLOCK, I.E.

A) OUR BLOCK QUOTA IS UP
B) WE HAVE REACHED THE END OF OUR INPUT FILE.

WHENEVER WE GET ANOTHER BLOCK, WE IGNORE THE FIRST RECORD SINCE THIS

ALWAYS CONTAINS 'STX'.

THE SECOND SECTION OF CODE GOES THROUGH THE DMC CHANNEL POINTERS (SEE SECTION ON RECEIVE FILE FORMAT) AND GIVES US:

A) START AND END ADDRESSES OF RECORD IN OUR BUFFER
B) THE LAST CHARACTER IN THE DATA

IF WE GET A DELIMITER CHARACTER OF 2, WE HAVE GOT A 'DLE STX', AND SO HAVE TO JUMP TO THE TOP OF THE ROUTINE TO ACCESS THE NEXT CHANNEL.

THE NEXT SECTION OF CODE UNPACKS THE RECORD INTO ARRAY 'TEMP', ALL THE TIME LOOKING FOR OUR DELIMITER CHARACTER TO FIND THE END OF THE DATA. WHEN WE HAVE GOT OUR LINE, WE CHECK TO SEE IF THE LAST CHARACTER IS 'EM' AND IF SO WE IGNORE IT. THIS CHARACTER IS APPENDED TO THE END OF EACH CARD IN A CARD FILE IN CERTAIN MODES, TO DENOTE THE END OF THE CARD, BUT IS NOT PART OF THE DATA.

RPUTC

THIS SUBROUTINE OUTPUTS A CHARACTER INTO THE OUTPUT BUFFER. NOTE THAT IT IS WRITTEN SO THAT IF A CHARACTER IS OUTPUT IN THE LHB OF A WORD, THE RHB IS SET TO ZERO. THIS MEANS THAT WE CAN PAD OUT THE WORD BY INCREMENTING THE OUTPUT ARRAY POINTER 'OTPTR'.

RPUTFB

THIS SUBROUTINE PROCESSES BINARY FILES. IN ORDER TO SPEED IT UP, THE ROUTINE RPUTL HAS BEEN EXPANDED INLINE, AND INSTEAD OF TAKING THE DATA OUT OF THE TEMPORARY ARRAY 'TEMP', IT IS TAKEN OUT OF THE RECEIVE FILE.

BINARY FILES MUST BE TRANSMITTED IN TRANSPARENT EBEDIC AS A CARD OR PUNCH FILE. THE PROBLEM COMES WHEN WE TRY TO FIND THE END OF THE DATA. WE CANNOT DO WHAT WE DO IN ROUTINE RGETL - I.E. SEARCH THROUGH OUR DMC CHANNEL FOR OUR DELIMITER CHARACTER, BECAUSE A DELIMITER CHARACTER MAY BE PART OF THE DATA. INSTEAD WE LOOI FOR THE DELIMITER CHARACTER IN POSITION 83, IF WE HAVE AN 'ESC' IN COLUMN 1. OTHERWISE WE LOOK FOR THE DELIMITER CHARACTER IN COLUMN 81. THIS IS ONE OF THE AREAS WHERE MODIFICATIONS WILL BE NEEDED FOR CERTAIN SITES.

RPUTFL

THIS ROUTINE PROCESSES NON-BINARY RECEIVE FILES. IT IS WRITTEN IN TWO PARTS:
A) PUNCH FILE PROCESSING
B) PRINT FILE PROCESSING

PUNCH FILE PROCESSING DIFFERS FROM PRINT FILE PROCESSING IN THAT WE DO NOT OUTPUT ANY OTHER VFC CHARACTERS EXCEPT FOR ONE NEW LINE CHARACTER AFTER EACH LINE. (EXCEPT FOR TRANSPARENT EBCDIC - WE ALWAYS ASSUME THAT IT IS BINARY). WE PROCESS THE PUNCH FILE IN THE FOLLOWING WAY:

A) GET THE NEXT RECORD
B) DECIDE IF WE HAVE VFC - IF SO, WE IGNORE IT AND ASSUME A NEW LINE.
C) OUTPUT THE LINE (EXCLUDING VFC), TRANSLATING INTO ASCII FOR EBCDIC, NONTRANSPARENT FILES. ILLEGAL CHARACTERS ARE FLAGGED, THEN ASSUMED TO BE SPACES, SO THAT PROCESSING CAN CONTINUE.
D) APPEND A NEW LINE ("NL") CHARACTER, AND START AGAIN

PRINT FILE PROCESSING IS THE SAME AS PUNCH FILE PROCESSING, EXCEPT THAT WE CAN ACTION ANY VFC RECEIVED. THE TYPE OF VERTICAL FORMS CONTROL IS LOOKED UP IN THE SAME TABLE FOR BOTH ASCII AND EBCDIC CHARACTERS. THE TYPES ARE AS FOLLOWS:

TYPES 1-3	SKIP THAT MANY LINES
4	FORM FEED (DEFAULT)
5-11	DEFINED IN VFC FILE
12	HT RECORD

WE ACTION OUR VFC AS FOLLOWS:
A) IF NO VFC IS PRESENT, ASSUME VFC WAS THE SAME AS LAST TIME. (DEFAULT IS A NEW LINE).
B) A REQUEST TO SKIP TO A CHANNEL WHOSE CORRESPONDING LINE NUMBER IS NOT ON THE CURRENT PAGE, WILL CAUSE A PAGE THROW, FOLLOWED BY THAT NUMBER OF "NL" CHARACTERS.
C) A REQUEST TO SKIP TO A CHANNEL WHOSE CORRESPONDING LINE NUMBER IS ON THE CURRENT PAGE, WILL CAUSE THE OUTPUT OF THAT NUMBER OF "NL" CHARACTERS.
D) A REQUEST FOR OUTPUT OF NEW LINE CHARACTERS WILL CAUSE A CHECK TO BE MADE ON THE MAXIMUM PAGE SIZE, TO SEE IF A PAGE THROW IS REQUIRED.

WE CAN DEFINE A VFC CONTROL FILE USING COMMAND VFCDEF. IT CAN ASSIGN A LINE NUMBER CORRESPONDING TO A CHANNEL NUMBER, AND THE PAGE SIZE. NOTE SOME OF THE FUNCTIONS OF THE CODE.

LABEL 450	- OUTPUTS N NEW LINES
410	- OUTPUTS FORM FEED AND N NEW LINES

RVFC

ROUTINE READS IN A NEW VFC FILE. EVERY TIME A NEW "VFCFIL" COMMAND IS ENTERED AT THE WORKSTATION, A FLAG IS SET TO TELL THE RECEIVE ROUTINES TO READ IN THE VFC FILE, BEFORE THE NEXT RECEIVE FILE IS PROCESSED.

6.3 PUNCH AND PRINT FOR 2780

TO AID THE ANALYST TO DETERMINE WHICH MODE HE SHOULD CONFIGURE A PARTICULAR SITE SEVERAL GUIDES ARE GIVEN AS TO WHAT IS IMPLEMENTED AT THE MOMENT. BY THE VERY NATURE OF 2780 AND THE MANY INTERPRETATIONS OF IT, THERE ARE BOUND TO BE SITES WHERE WHAT WE DO IS WRONG.

RUNNING TO IBM SITE

THERE IS NO PROBLEM ACTUALLY TRANMITTING TO AN IBM SITE. THE PROBLEM IS WHAT WE GET BACK. DUE TO THE MANY USES A 2780 CAN BE USED TO A DECISION HAS BEEN MADE NOT TO TRANSLATE FILES AT ALL THAT WE THINK ARE TRANPARENT PUNCH FILES. THE ALGORITHM FOR DECIDING THIS IS GIVEN LATER ON. SO, AS A GUIDE, ANY MODE THAT WILL CAUSE YOUR HOST SITE TO GIVE CORRECT INTERPRETATION OF THIS FILE WILL BE OK. GENERALLY SPEAKING, WE DON'T REALLY WANT TO USE A MODE WHICH WILL CAUSE PADDING, WHERE WE CAN USE AN "EM" CHARACTER, ON THE GROUNDS OF EFFICIENCY.

RUNNING TO A NON-IBM SITE

THIS IS WHERE WE CAN HAVE HEADACHES. WE SHALL SPLIT THIS TOPIC INTO 2 SECTIONS - IE TRANSMIT AND RECEIVE.

SOME MACHINES CAN ACCEPT PRINT AND PUNCH FILES IN HOST MODE. (SO CAN WE). AS HOST, WE SEND CARDS AS A PUNCH FILE, WITH COMPONENT SELECTION, AND VFC, AND PRINT FILES WITH VFC AND COMPONENT SELECTION. THESE FUNCTIONS ARE EASILLY CHANGEABLE BY UPDATING TBLK FOR COMPONENT SELECTION, AND THE CORRESPONDING ENTRY OF ESCHAR AND HSTT IN TDATA. FOR THIS MODE WE WANT TO BE SET UP AS "RJE HOST". A WORD OF CAUTION, THOUGH. THIS HAS NOT BEEN TRIED TO A LIVE SITE, ONLY TO OURSELVES, RUNNING IN LOOPBACK.
IF WE ARE TALKING TO A SITE THAT IS MORE LIKE AN IBM, WE WANT TO SET OURSELVES UP AS A SLAVE. AS FAR AS WE ARE CONCERNED THE ONLY DIFFERENCE BETWEEN THE 2 MODES IS WHAT WE TRANMSIT - DEVICE SELECTION AND VFC WHEN IN HOST MODE.

RECEIVE PROCESSING IS WHERE THE FUN STARTS. WE ARE GEARED UP TO ACCEPT FILES IN HOST MODE WITH COMPONENT SELECTION AND VFC. IF THE SITE DOES NOT SEND EITHER OF THESE, WE DO OUR BEST. TO CHOSE THE DEVICE WE ARE CURRENTLY RUNNING TO SEE SECTION ON ROUTINE RECV IN RECEIVE PROCESSING. TO MAKE CHANGES TO THIS ASPECT OF ROUTINE PROCESSING, THE ROUTINE IS RECV AROUND LABEL 130. WE EVEN EXPECT VFC ON BINARY FILES(TRANSPARENT EBCDIC PUNCH FILES AGAIN). IF FOR SOME REASON THE PRESENT CODE IS NOT PERMISSIBLE CHANGES WILL BE NEEDED IN RPUTFB.

PRIME TO PRIME COMMUNICATION

IT IS RECOMMENDED THAT ALL FILES SHOULD BE SENT AS BINARY DATA, UNLESS A MIXTURE OF PUNCH, PRINT FILES ARE GOING TO BE SENT, WHERE THE CORRECT COMPONENT SELECTION, AND AUTOMATIC SPOOLING AT THE OTHER END IS REQUIRED.

SENDING DATA AS BINARY ENSURES THAT AN EXACT COPY OF THE ORIGINAL IS RECEIVED BY THE OTHER END (EXCEPT THAT NULLS MAY BE APPENDED TO THE END TO MAKE UP AN 80 BYTE RECORD). THE ADVANTAGES OF THIS MODE ARE:

A) SPEED - NO TRANSLATION NECCESSARY ON CONVERSION.
B) HIGHEST DATA PACKING DENSITY - NO ARTIFICIAL PADDING TO MAKE UP THE CORRECT LENGTH RECORDS.
C) THIS IS THE ONLY MODE THAT WILL SEND OVERPRINTED LINE FORTRAN FILES.

DO NOT USE TRANSPARENT EBCDIC PUNCH MODE. IT IS NOT COMPATIBLE WITH PRIME-TO-PRIME COMMUNICATION.

PRINT DATA MAY BE SENT IN ANY MODE, BUT NOTE THAT IN TRANSPARENT MODE, EACH LINE IS PADDED TO 136 CHARACTERS, WHICH IS INEFFICIENT AND SLOW. ALSO NOTE THAT 2780 PROTOCOL DOES NOT SUPPORT LINE OVERPRINTING (FORTRAN CARRIAGE CONTROL "+"), SO IF THIS FEATURE IS IMPORTANT, SEND IT AS A BINARY FILE.

PUNCH DATA MAY BE SENT IN NON-TRANSPARENT MODE. IT IS ALWAYS MORE EFFICIENT TO USE THE OPTION "PAD NO", TO MINIMIZE THE NUMBER OF BLOCKS SENT.

THE ADVANTAGES OF SENDING DATA AS PUNCH OR PRINT FILES IS THAT THE CORRECT RECEIVE FUNCTION WILL ALWAYS BE SELECTED AT THE REMOTE END, SO THAT WE CAN INTERSPERSE PRINT AND PUNCH FILES WITH NO WORRIES.

WHEN RUNNING PRIME-TO-PRIME, BOTH MACHINES CAN BE SET UP AS HOST, TO ALLOW SENDING PRINT FILES IN BOTH DIRECTIONS, BUT ONLY ONE MACHINE MAY BE SET UP AS THE PRIMARY MACHINE.

SECTION 7.1

ICL 7020 TRANSMIT PROCESSING MODULE

7.1.1 INTRODUCTION

THE TRANSMIT MODULE IS CALLED BY THE WORKSTATION OR THE RJ UTILITY WHEN THE SEND COMMAND IS ISSUED. IT TAKES AS INPUT THE USER FILE WHICH IS TO BE SENT AND CREATES A TRANSMIT (T-) FILE OF A FORMAT SUITABLE FOR THE SYMBIONT TO TRANSMIT TO THE ICL HOST.

THE TRANSMIT MODULE WAS ORIGINALLY WRITTEN IN THE FORM OF SUBROUTINES; THESE WERE CHANGED LATER TO DIRECT GOTO'S BUT THE SUBOUTINE STRUCTURE HAS BEEN RETAINED AND THE FOLLOWING DESCRIPTION REFERS TO THESE SUBROUTINES.

THE TRANSMIT MODULE CURRENTLY PROCESSES A COMPLETE FILE IN ONE GO, BUT THERE IS ONE INPUT BUFFER PER LINE SO THAT IT COULD BE CHANGED TO PRODUCE A CERTAIN NUMBER OF WHOLE OUTPUT BLOCKS AND RETURN.

THE TRANSMIT MODULE CONSISTS OF THE FOLLOWING ROUTINES:-

TRANSM	MAIN ROUTINE, OPENS AND CLOSES INPUT AND OUTPUT FILES AND PRINTS FINAL STATISTICS.
TPUTFL	CALLED BY TRANSM TO PROCESS THE FILE, ADDING HEADER AND TRAILER WORDS FOR EACH RECORD.
TGETC	CALLED FROM TPUTFL TO GET THE NEXT CHARACTER FROM THE INPUT FILE.
TPUTC	CALLED FROM TPUTFL TO PUT A CHARACTER IN THE OUTPUT BUFFER.
TOUTP	CALLED FROM TPUTFL TO WRITE AN OUTPUT BUFFER TO DISC.
TXINIT	CALLED FROM TRANSM TO INITIALISE VARIABLES AND POINTERS.

7.1.2 TRANSM

THE TRANSM ROUTINE IS PASSED THREE ARGUMENTS:-

1. THE NAME OF THE FILE TO BE SENT
2. THE LENGTH OF THAT NAME
3. NOT USED FOR 7020

COMMON BLOCK VARIABLE CLINE CONTAINS THE NUMBER OF THE LINE ON WHICH THE FILE IS TO BE TRANSMITTED.

TRANSM INITIALISES POINTERS IF IT IS THE FIRST TIME IT HAS BEEN CALLED FOR THIS LINE.

THE INPUT FILE IS THEN OPENED AND THE FIRST WORD IS READ TO CHECK IF THE FILE IS A CONCAT FILE; IF IT IS, THEN THE NEXT THREE WORDS ARE

READ TO GET THE TYPE AND LENGTH OF THE FIRST CONCAT SECTION (THE START OF A SUBSEQUENT CONCAT SECTIONS IS READ IN ROUTINE TPUTFL).

A TEMPORARY OUTPUT FILE (CALLED TDXXXX) IS THEN OPENED (IT WILL LATER BE RENAMED TO A TXXXXX FILE IF PROCESSING HAS BEEN SUCCESSFUL).

TPUTFL IS THEN CALLED TO PROCESS THE INPUT FILE.

ON RETURN FROM TPUTFL, THE INPUT AND OUTPUT FILES ARE CLOSED, (OUTPUT FILE RENAMED TO TXXXXX) AND STATISTICS INFORMATION IS OUTPUT.

CONTROL THEN RETURNS TO THE WORKSTATION OR RJ UTILITY.

7.1.3 TPUTFL

TPUTFL IS CALLED FROM TRANSM TO PROCESS THE INPUT FILE. TGETC IS CALLED TO GET THE NEXT CHARACTER FROM THE INPUT FILE (RETURNED IN COMMON VARIABLE CHAR, RIGHT ADJUSTED, ZERO FILLED).

FOR TEXT DATA, IF THE CHARACTER RETURNED IS AN RVT OR RHT CHARACTER, TGETC IS AGAIN CALLED TO GET THE NUMBER OF NL OR SPACE CHARACTERS WHICH ARE TO BE OUTPUT. IF THE CHARACTER RETURNED IS A NL CHARACTER, THEN THIS IS THE END OF A LOGICAL RECORD, THE NL IS CHANGED TO RS, AN ETB CHARACTER IS ADDED AND THE RECORD HEADER IS FILLED IN. LINES OF DATA LONGER THAN 80 CHARACTERS ARE TRUNCATED AND NULL CHARACTERS ARE IGNORED.

FOR BINARY DATA (CAN OCCUR ONLY IN A CONCAT SECTION), A RECORD IS TERMINATED AFTER EVERY 80 DATA CHARACTERS (THE HEADER AND TRAILING ETB BEING ADDED) AND THE LAST RECORD IS PADDED OUT TO 80 CHARACTERS WITH SPACES. NULL CHARACTERS ARE NOT IGNORED.

TPUTC IS CALLED FROM TPUTFL EACH TIME TO ADD A CHARACTER TO THE OUTPUT BLOCK.

EACH TIME AN OUTPUT RECORD IS COMPLETED, THE OUTPUT BUFFER POINTERS ARE CHECKED, AND IF 1024 WORDS HAVE BEEN REACHED OR EXCEEDED, TOUTP IS CALLED TO WRITE THE OUTPUT BUFFER TO DISC. A WHOLE NUMBER OF RECORDS MUST BE WRITTEN AS THE RECORD HEADER CANNOT BE FILLED IN UNTIL THE END OF THE RECORD HAS BEEN REACHED AND THE LENGTH OF THE RECORD IS KNOWN.

WHEN THE END OF THE INPUT FILE HAS BEEN DETECTED, THE LAST ETB IS CHANGED TO ETX (AND THE LAST RS TO HLR FOR TEXT DATA). IF THE LAST RECORD HAS ALREADY BEEN WRITTEN TO DISC, THE LAST ONE OR TWO WORDS MUST BE READ BACK AND REWRITTEN. THE LAST BUFFER IS WRITTEN TO DISC AND CONTROL RETURNS TO TRANSM.

NOTE THAT IN THIS ROUTINE, TGETC IS USED TO GET THE TYPE AND LENGTH OF CONCAT SECTIONS, BYTE BY BYTE. THE TYPE OF THE FIRST SECTION ONLY IS READ IN TRANSM.

7.1.4 TGETC

TGETC IS CALLED FROM TPUTFL TO GET THE NEXT CHARACTER FROM THE INPUT

FILE. A NEW BUFFER IS READ IF NECESSARY. THE CHARACTER IS RETURNED IN THE RH BYTE OF COMMON VARIABLE CHAR (ZERO FILLED).

7.1.5 OUTPUT (T-) FILE FORMAT

```
FILE HEADER     WORD 0     ZERO
                WORD 1     ZERO
DATA RECORDS, MAX 44 WORDS.
                WORD 0     BITS 1-10 NO OF WORDS TO END OF
                           RECORD INCLUDING THIS WORD, BIT 16 =1

                WORD 1     NO OF WORDS TO END OF RECORD
                           INCLUDING  THIS  WORD (THEREFORE 1 LESS THAN
                           VALUE IN BITS 1-10 OF PREVIOUS WORD)

                WORD 2      ZERO

                WORD 3     )UP TO 40 DATA WORDS
                      .    )
                      .    )
                      .    )
                      .    )
                WORD N
                WORD N+1    CONTAINS ETB (ETX FOR LAST RECORD)
```

ZERO WORD

SECTION 7.2

7020 RECEIVE PROCESSING MODULE

7.2.1 INTRODUCTION

THE RECEIVE MODULE IS CALLED BY THE WORKSTATION TO PROCESS ALL OR PART OF A SYMBIONT-PRODUCED RECEIVE (R-) FILE FOR A GIVEN LINE. THE PROCESSED DATA IS WRITTEN TO PRINT OR PUNCH FILES.

CONTROL RETURNS TO THE WORKSTATION WHEN 10 BLOCKS OF THE RECEIVE FILE HAVE BEEN PROCESSED, WHEN LOGICAL END OF PRINT OR PUNCH FILE HAS BEEN DETECTED OR WHEN PHYSICAL END OF RECEIVE FILE HAS BEEN DETECTED. IN THIS WAY, OTHER WORKSTATION PROCESSING IS NOT HELD UP FOR TOO LONG.

7.2.2 INPUT/OUTPUT FILES

RECEIVE (R-) FILES

RECEIVE FILE NAMES ARE OF THE FORMAT:

RNXXXX
WHERE N IS THE LINE NUMBER
XXXX IS A SEQUENCE NUMBER STARTING FROM 0001

THE FORMAT OF THE RECEIVE FILE IS:

```
HEADER    (2  WORDS)
BLOCK 1   (41 WORDS)
BLOCK 2   (41 WORDS)
   .
   .
   .
BLOCK N   (41 WORDS)
```

```
WHERE HEADER   IS
      WORD 1   UNDEFINED
      WORD 2   000002 = PRINT FILE
               000006 = PUNCH FILE
```

AND EACH BLOCK OF 41 WORDS CONTAINS USEFUL DATA UP TO THE FIRST ETX OR ETB CHARACTER - THE REMAINDER OF THE BLOCK IS IGNORED.

S-FILES

AS THE RECEIVE FILES ARE PROCESSED, THEY CAN EITHER BE DELETED OR SAVED AS S FILES (WHERE THE LEADING R IN THE R-FILE NAME IS CHANGED TO S) IN THE Q UFD. THE OPTION IS SELECTED USING THE SAVE COMMAND.

PRINT FILES

PRINT OUTPUT FILES ARE AUTOMATICALLY PLACED ON THE SPOOL OUTPUT QUEUE (PRTXXX FILES IN UFD SPOOLQ).

PUNCH FILES

PUNCH OUTPUT FILES CAN BE:-

1. NOT TRANSLATED, LEFT IN Q UFD.
2. TRANSLATED, LEFT IN THE Q UFD.
3. TRANSLATED, ADDED TO THE SPOOL QUEUE.

THE OPTION IS SELECTED USING THE PUNCH COMMAND.

7.2.3 RECEIVE MODULE STRUCTURE

THE RECEIVE MODULE WAS WRITTEN ORIGINALLY IN THE FORM OF SUBROUTINES; THESE WERE CHANGED LATER TO DIRECT GOTOS. HOWEVER, THE SUBROUTINE STRUCTURE HAS BEEN RETAINED AND COMMENTS IN THE CODE INDICATE THE ORIGINAL STRUCTURE. THEREFORE THE FOLLOWING DESCRIPTION REFERS TO THESE SUBROUTINES.

THE RECEIVE MODULE IS MADE UP OF THE FOLLOWING ROUTINES.

RECV	ORGANISES OPENING, CLOSING, DELETING OR SAVING OF INPUT (R-) FILES AND OUTPUT (PRINT OR PUNCH) FILES. IT ALSO OUTPUTS STATISTICS FOR EACH PRINT AND PUNCH FILE PRODUCED.
PRINIT	CALLED BY RECV TO INITIALISE VARIABLES FOR PRINT OUTPUT.
PUINIT	CALLED BY RECV TO INITIALISE VARIABLES FOR PUNCH OUTPUT.
RPUTFL	CALLED BY RECV TO PROCESS ALL OR PART OF A RECEIVE FILE.
RGETC	CALLED BY RPUTFL AND RETURNS THE NEXT CHARACTER OF THE RECEIVE FILE UNTIL END OF FILE IS DETECTED.
EOFDET	CALLED BY RPUTFL FOR PRINT FILES ONLY, ONCE PER PRINT LINE TO CHECK FOR LOGICAL END OF PRINT FILE.
PUTCPR	CALLED BY RPUTFL TO PLACE A PROCESSED CHARACTER IN THE PRINT OUTPUT BUFFER.
PUTCPU	CALLED BY RPUTFL TO PLACE A PROCESSED CHARACTER IN THE PUNCH OUTPUT BUFFER.
PROUTP	CALLED BY RPUTFL TO WRITE A PRINT OUTPUT BUFFER TO DISC.
PUOUTP	CALLED BY RPUTFL TO WRITE A PUNCH OUTPUT BUFFER TO DISC.

RECV IS CALLED BY THE WORKSTATION; COMMON BLOCK VARIABLE RLINE IS SET TO THE NUMBER OF THE LINE FOR WHICH R- FILES ARE TO BE PROCESSED.

VARIABLES ARE INITIALISED IF THIS IS THE FIRST TIME RECV HAS BEEN CALLED FOR THIS LINE.

IF RECV HAS BEEN CALLED WITH INSTRUCTIONS TO CLOSE DOWN RECEIVE PROCESSING, THIS IS ONLY DONE IF END OF A RECEIVE FILE AND LOGICAL END OF PRINT AND PUNCH FILES HAVE ALL BEEN REACHED SIMULTANEOUSLY.

FOR THE TEST SITE, IT HAS ALSO BEEN ALLOWED TO CLOSE AFTER A SPECIAL 'HEADER' LINE, WHICH IS OUTPUT REPEATEDLY.

IF RECEIVE PROCESSING IS TO CONTINUE, THE NEXT R-FILE IS OPENED IF ONE IS NOT ALREADY OPEN. IF THE NEXT R-FILE IS NOT FOUND, CONTROL RETURNS TO THE WORKSTATION, SETTING THE TODO PARAMETER TO FALSE I.E. THERE IS NO MORE TO DO FOR THE MOMENT. IF THE R-FILE IS FOUND TO BE COMPLETELY EMPTY (COULD BE AS A RESULT OF THE RECOVER COMMAND) A MESSAGE IS OUTPUT AND THE NEXT FILE IS AUTOMATICALLY SEARCHED FOR.

IF A NEW R-FILE IS OPENED, THE SECOND WORD INDICATES WHETHER IT IS PART OF A PRINT OR PUNCH FILE; THE RELEVANT OUTPUT FILE IS OPENED IF NECESSARY.

RPUTFL IS THEN CALLED WHICH PERFORMS THE ACTUAL PROCESSING OF THE R-FILE. CONTROL RETURNS TO RECV WHEN 10 BLOCKS OF R-FILE HAVE BEEN PROCESSED, WHEN LOGICAL END OF PRINT OR PUNCH FILE OR PHYSICAL END OF R-FILE HAVE BEEN DETECTED. IF END OF FILE DETECTION HAS BEEN DISABLED, THEN LOGICAL END OF PRINT OR PUNCH FILE IS SET AT THE END OF EACH RECEIVE FILE.

IF END OF R-FILE HAS BEEN DETECTED, THE FILE IS CLOSED (AND RENAMED AS AN S-FILE IF IT IS TO BE SAVED); THE R-FILE NAME IN THE XY FILE IS UPDATED TO THE NEXT RECEIVE FILE. IF LOGICAL END OF PRINT OR PUNCH FILE HAS BEEN DETECTED, THE FILE IS CLOSED AND RELEVANT STATISTICS ARE OUTPUT.

IF A PRINT OR PUNCH FILE HAS NOT BEEN CLOSED, PROCESSING TIME FOR RECEIVE PROCESSING IS ACCUMULATED AND CONTROL RETURNS TO THE WORKSTATION.

RPUTFL

RPUTFL IS CALLED BY RECV. SEPARATE SECTIONS OF THIS ROUTINE ARE EXECUTED FOR PRINT, PUNCH/TRANSLATE AND PUNCH/NO TRANSLATE OUTPUT FILES.

FOR PRINT OUTPUT, RGETC IS CALLED TO GET A CHARACTER. IF THIS CHARACTER IS:-

1. A NL CHARACTER AND END OF LOGICAL PRINT FILE IS TO BE SEARCHED FOR, EOFDET IS CALLED; IF NOT, END OF LOGICAL PRINT FILE IS SET WHEN THE END OF AN R-FILE IS DETECTED BY RGETC.

2. A CR CHARACTER, THIS IS REPLACED BY CR, NL AS EVERYTHING UP TO AND INCLUDING THE FOLLOWING NL CHARACTER IS IGNORED BY OUR SPOOLER/PRINTER.

3. AN FF CHARACTER, THIS IS REPLACED BY NL, FF, NL.

4. AN HT CHARACTER, RGETC IS AGAIN CALLED TO GET THE FOLLOWING COUNT CHARACTER AND THAT NUMBER OF SPACES IS OUTPUT.

5. POUND, # OR $ CHARACTER, THEY ARE CONVERTED TO PRIME POUND,\ OR $ CHARACTERS.

ANY OTHER CHARACTERS ARE NOT CHANGED.

FOR PUNCH/TRANSLATE OUTPUT, THIS IS TREATED THE SAME AS PRINT OUTPUT EXCEPT THAT ROUTINE EOFDET IS NOT CALLED AT THE END OF A LOGICAL LINE. INSTEAD, END OF PUNCH FILE IS DETECTED WHEN MORE THAN 80 CONSECUTIVE NULL CHARACTERS HAVE BEEN PROCESSED (THEY ARE IGNORED). HOWEVER, IF NOTHING HAS BEEN WRITTEN TO THE FILE, THEN IT IS NO CLOSED.

FOR PUNCH/NO TRANSLATE OUTPUT, NO CHARACTER TRANSLATION TAKES PLACE; NULLS ARE NOT IGNORED AND END OF PUNCH FILE IS DETECTED WHEN MORE THAN 80 CONSECUTIVE NULL CHARACTERS HAVE BEEN PROCESSED.

IF END OF R-FILE IS DETECTED AND END OF FILE DETECTION HAS BEEN DISABLED, END OF LOGICAL OUTPUT FILE IS SET AND CONTROL RETURNS TO RECV. CONTROL ALSO RETURNS TO RECV IF LOGICAL END OF OUTPUT FILE IS DETECTED OR WHEN 10 BLOCKS OF THE R FILE HAVE BEEN PROCESSED.

RGETC

RPUTFL CALLS RGETC TO GET THE NEXT CHARACTER OF THE RECEIVE FILE. A NEW INPUT BUFFER IS READ THE FIRST TIME THIS ROUTINE IS CALLED AND WHENEVER THE INPUT BUFFER HAS BEEN EXHAUSTED.

IF PRINT OR PUNCH/TRANSLATE OUTPUT IS BEING PRODUCED, NULL CHARACTERS ARE IGNORED.

IF AN ETX OR ETB CHARACTER IS DETECTED, THIS IS NOT RETURNED; THE REMAINDER OF THE 41-WORD BLOCK IS IGNORED AND THE FIRST CHARACTER FROM THE NEXT BLOCK IS OBTAINED.

IF END OF R-FILE IS DETECTED, A FLAG IS SET AND CONTROL IS RETURNED TO RPUTFL. RGETC ALSO COUNTS THE NUMBER OF BLOCKS READ AND THE NUMBER OF CHARACTERS PROCESSED FOR STATISTICS OUTPUT LATER.

EOFDET

EOFDET IS CALLED ONCE PER LOGICAL PRINT LINE IF LOGICAL END OF PRINT FILE DETECTION IS ENABLED. IF END OF FILE IS DETECTED, A FLAG IS SET.

THIS ROUTINE ONLY DETECTS THE TEST SITE END OF FILE STRING. ARRAY EOFSTR CONTAINS THE STRING WITH WHICH THE BEGINNING OF EACH LINE IS COMPARED, HERE ^***6S AND VARIABLE EOFPRC EQUALS THE NUMBER OF

CHARACTERS TO BE COMPARED, HERE = 6.

THIS ROUTINE MUST BE ALTERED FOR OTHER END OF FILE STRINGS.

THIS ROUTINE ALSO TESTS EACH LINE FOR A HEADER LINE (BEGINNING WITH ^*6S) AS THE TEST SITE SENDS SEVERAL OF THESE AFTER EACH LOGICAL FILE. IF ONE OF THESE LINES IS DETECTED, A FLAG IS SET, RECEIVE PROCESSING IS ALLOWED TO TERMINATE ON REQUEST IF THIS IS THE LAST LINE WHICH HAS BEEN RECEIVED.

PUTCPR

PUTCPR IS CALLED FROM RPUTFL; THE CHARACTER IN VARIABLE CHAR IS PLACED IN THE NEXT CHARACTER POSITION OF THE PRINT OUTPUT BUFFER (CR AND NL ARE PADDED WITH NULL CHARACTERS IF THEY FALL IN THE LEFT HAND BYTE OF A WORD).

PUTCPU

AS PUTCPR BUT FOR PUNCH, NO NULL PADDING OF CR OR NL TAKES PLACE FOR PUNCH/NO TRANSLATE.

SECTION 8

GRTS TRANSMIT AND RECEIVE

INTRODUCTION

IT IS THE PURPOSE OF THIS SECTION TO DESCRIBE THE DATA STRUCTURE AND THE CONCEPTS BEHIND CERTAIN ACTIONS TAKEN BY THE PROGRAM CODE. THIS SECTION DOES NOT PROPOSE TO EXPLAIN THE USE OF EACH INDIVIDUAL VARIABLE OR REASON FOR EACH LINE OF CODE AS THIS SHOULD BE FAIRLY CLEAR FROM THE COMMENTS IN THE PROGRAM. IT IS HOPED THAT BY A BETTER UNDERSTANDING OF THE PROGRAM AND DATA STRUCTURE, MUCH TIME WILL BE SAVED BY THE ANALYST WHO HAS TO MODIFY ANY PART OF GRTS TRANSMIT OR RECEIVE.

8.1 RECEIVE PROCESSING

THE RECEIVE PROCESSOR IS A MODULE OF THE WORK STATION. ITS PURPOSE IN LIFE IS TO PICK UP A FILE FROM THE RECEIVE QUEUE AND TRANSFORM IT INTO ONE WHICH A PRIME MACHINE CAN UNDERSTAND. CURRENTLY, ALL GRTS RECEIVE FILES ARE TREATED AS PRINT FILES AND ARE CONVERTED AS SUCH.

8.1.1 THE RECEIVE FILE

A RECEIVE FILE IS PRODUCED BY THE SYMBIONT AND CONTAINS ALL THE NECESSARY INFORMATION TO CONVERT TO PRIME STYLE.

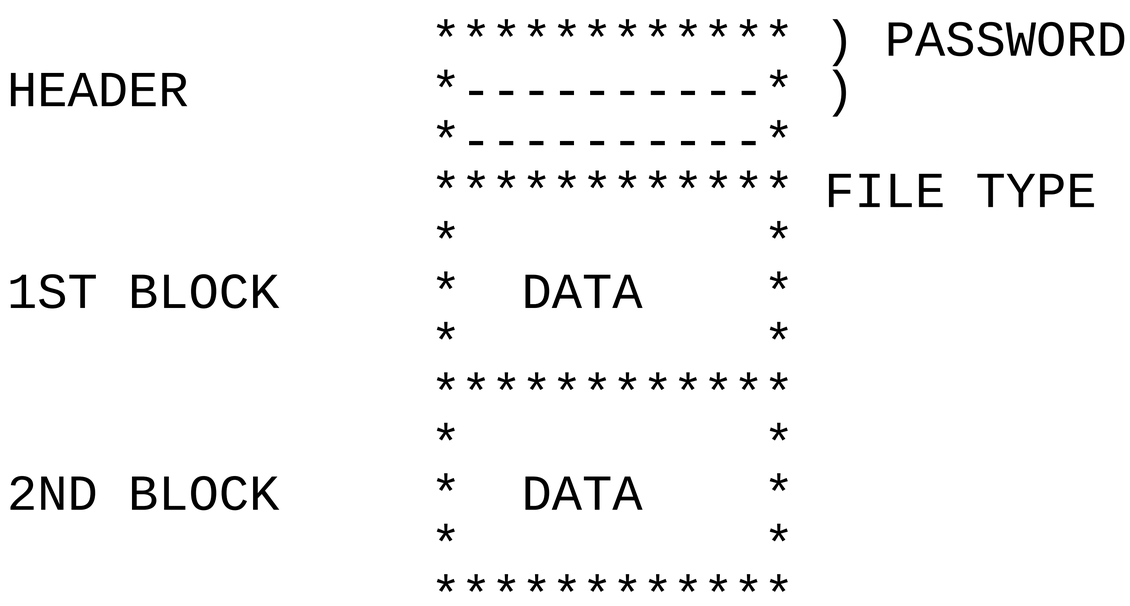


FIG. 8.1.1 RECEIVE FILE FORMAT

FIG. 8.1.1 GIVES THE SIMPLE LAYOUT OF THIS FILE.
FIG. 8.1.2 DESCRIBES THE DATA BLOCK.

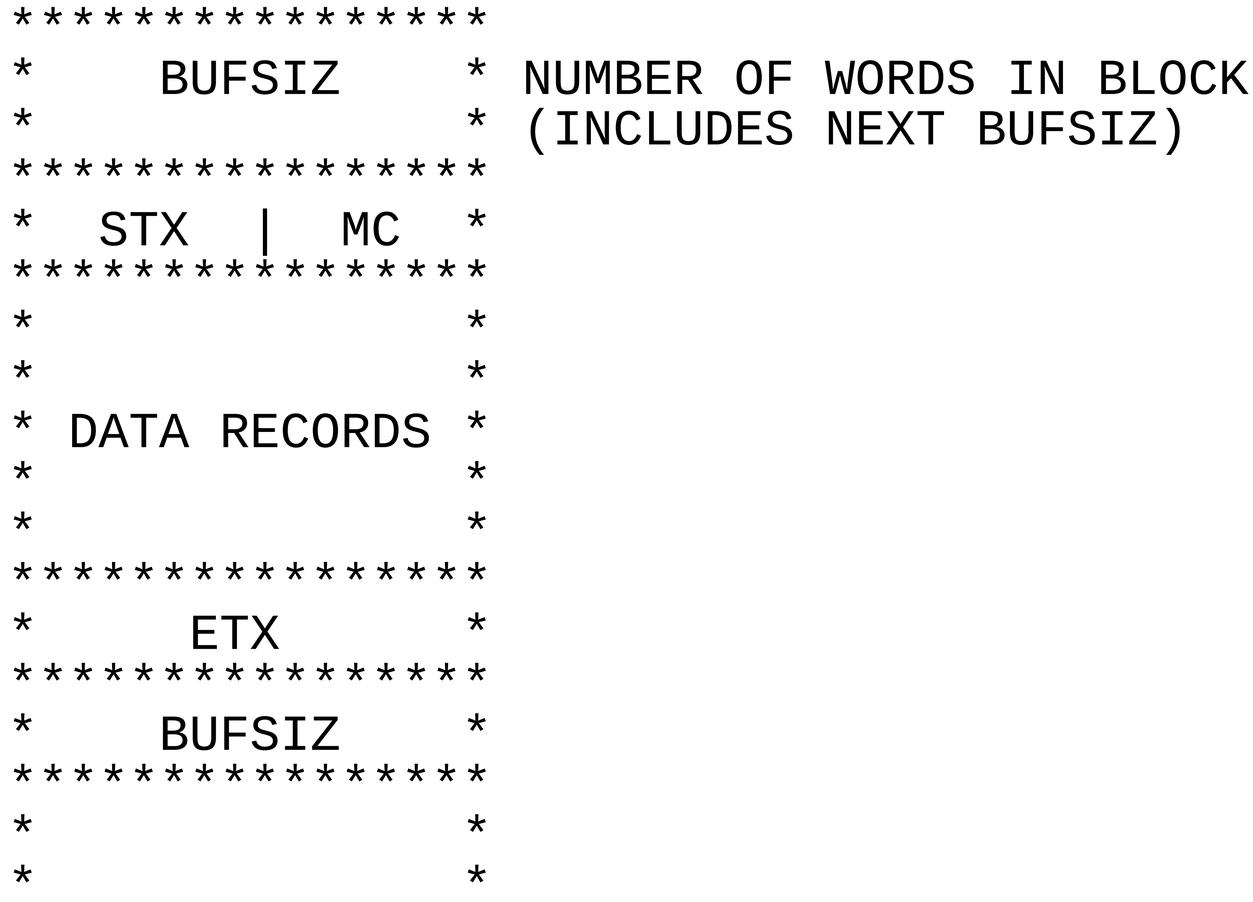


FIG. 8.1.2 RECEIVE FILE DATA BLOCKS

AT THE END OF THE FILE BUFSIZ IS ZERO. EACH DATA BLOCK CAN CONTAIN SEVERAL LINES OF PRINT FORMATTED AS FOLLOWS.

IT IS POSSIBLE FOR EITHER OR BOTH THE FIRST MEDIA CODE (MC) AND LAST RECORD SEPARATOR (RS) TO BE MISSING. THIS IS CALLED A SPLIT RECORD. IN OTHER WORDS, A BLOCK CAN ONLY BE A MAXIMUM OF 324 CHARACTERS BUT IT IS POSSIBLE FOR A RECORD TO SPAN TWO BLOCKS.

THE MC DEFINES WHAT TYPE OF DATA FOLLOWS. IF THE CODE IS OCTAL 314 THEN THE DATA IS FOR A PRINTER. IN THIS CASE THE CHARACTER IMMEDIATELY PRECEDING EACH RS IS CALLED A SLEW CHARACTER AND IS USED TO CONTROL THE SLEWING OF THE PRINTER.

IT IS ALSO POSSIBLE TO HAVE CHARACTERS COMPRESSED IN THE FOLLOWING MANNER:

WHERE X IS THE CHARACTER BEING COMPRESSED, US (UNIT SEPARATOR) INDICATES COMPRESSION AND COUNT IS THE NUMBER OF TIMES X OCCURS (NB. COUNT IS TRANSMITTED IN BCD AND REQUIRES CONVERSION).

8.1.2 END OF FILE DETECTION

IT IS QUITE POSSIBLE FOR ONE LOGICAL JOB TO BE IN ONE RECEIVE FILE. HOWEVER, IT IS ALSO POSSIBLE FOR ONE JOB TO SPAN SEVERAL RECEIVE FILES OR TO BE A PART OF A RECEIVE FILE. THE RECEIVE PROCESSOR IS AWARE OF THESE POSSIBILITIES AND WHEN INSTRUCTED TO, WILL TEST FOR THE END OF A LOGICAL FILE ANYWHERE WITHIN A PHYSICAL FILE. THIS IS ACCOMPLISHED BY MAINTAINING A BUFFER, HOLDING ONE CHARACTER PER WORD, AND WHEN AN RS IS DETECTED, THIS BUFFER IS INTERROGATED AND COMPARED WITH A KNOWN CHARACTER STRING. IF THE STRINGS MATCH THEN THE END OF LOGICAL FILE HAS BEEN FOUND. THIS IS TALKED ABOUT IN MORE DETAIL LATER.

8.1.3 DATA STRUCTURE

THE WORKSTATION IS COMPILED USING A PARAMETER IN RJECOM>PARAM CALLED NLINE. THIS PARAMETER DEFINES THE NUMBER OF SMLC LINES THAT THE WORKSTATION CAN HANDLE. MANY VARIABLES ARE THEREFORE DIMENSIONED TO NLINE. TO MAKE THIS PART OF THE DOCUMENT MORE READABLE, ONLY OCCASIONALLY WILL IT REFER TO A VARIABLE BY THIS DIMENSION.

BOTH INPUT AND OUTPUT FILES ARE BUFFERED TO PLACE AS SMALLER A LOAD ON THE FILE SYSTEM AS SEEMED REASONABLE. IN FACT THE INPUT BUFFER IS FILLED ONE RECEIVE BLOCK AT A TIME, WHILST THE OUTPUT BUFFER IS 1024 WORDS LONG.

FIG. 8.1.3 SHOWS THE INPUT AND OUTPUT BUFFERS WITH THEIR ASSOCIATED POINTERS.

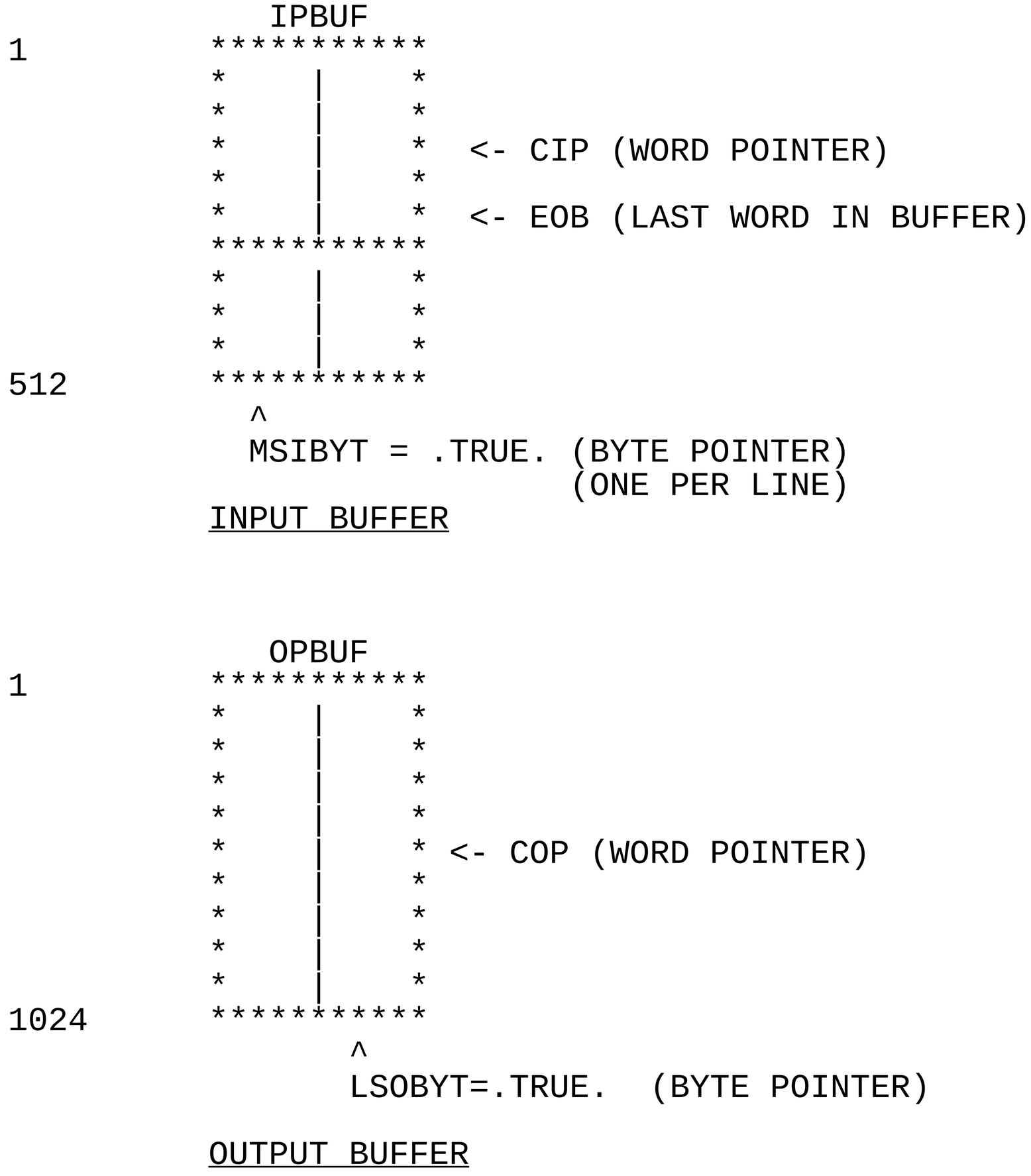


FIG. 8.1.3 INPUT AND OUTPUT BUFFERS

8.1.3.1 PROGRAM MODULES

THE FOLLOWING SUBROUTINES ARE USED FOR RECEIVE PROCESSING:-

```
     RECV       IN FILE RECEIV
     RPUTFL     IN FILE RPUTFL
                NOTE:  THIS SUBROUTINE CONTAINS TWO INTERNAL SUBROUTINES
                       RGETC AND RPUTC.
     GETBUF     IN FILE RSUBS
     GETNEW        DITTO
     GETOLD        DITTO
     RUSER         DITTO
     MOVCHR        DITTO
     EOFDET     IN FILE ENDDET
```

RECV IS THE LINK BETWEEN THE WORKSTATION AND THE RECEIVE PROCESSOR. IT HANDLES THE OPENING AND CLOSING OF FILES, INITIALISATION AND PRINTING OF STATISTICS AND CLEANING UP AFTER AN ERROR. IT IS CALLED WITH THE ARGUMENT TODO WHICH, IF TRUE ON RETURN, INDICATES TO THE WORKSTATION THAT THERE IS MORE PROCESSING TO DO ON THIS LINE. THE IDEA HERE IS TO RETURN FROM RECEIVE PROCESSING TO THE MAIN WORKSTATION SEVERAL TIMES BEFORE THE RECEIVE FILE HAS BEEN FULLY PROCESSED TO ENABLE THE WORK STATION TO PROCESS OTHER COMMANDS AND TO CHANGE LINES IF NECESSARY. THIS MEANS THAT ALL LINES GET A BIT OF PROCESSING ON A FREQUENT BASIS.

AFTER OPENING, OR ENSURING THAT INPUT AND OUTPUT FILES ARE OPEN, RECV ISSUES A CALL TO THE MAIN CHARACTER PROCESSING ROUTINE RPUTFL.

RPUTFL PROCESSES THE INPUT AND OUTPUT FILES CHARACTER BY CHARACTER. A CHARACTER IS OBTAINED FROM THE INPUT BUFFER BY AN INTERNAL SUBROUTINE RGETC. A CHECK IS MADE ON THE VALIDITY OF THE CHARACTER AND WHETHER THE END OF PHYSICAL INPUT FILE HAS BEEN REACHED. THE CHARACTER OBTAINED IS NOW PLACED IN THE OUTPUT BUFFER BY ANOTHER INTERNAL SUBROUTINE RPUTC. WHEN APPROPRIATE THE SAME CHARACTER IS PLACED IN THE END OF FILE DETECTION BUFFER. IF EITHER END OF LOGICAL, OR END OF PHYSICAL INPUT FILE IS DETECTED, THE OUTPUT BUFFER IS FLUSHED TO DISK.

RGETC AND RPUTC HAVE BEEN WRITTEN AS INTERNAL SUBROUTINES TO INCREASE THE PROCESSING RATE. EACH OF THESE ROUTINES IS CALLED APPROXIMATELY ONCE PER CHARACTER. THIS CALLING RATE PRODUCED AN OVERHEAD BECAUSE OF THE PCL (PROCEDURE CALL) MECHANISM.

GETBUF OBTAINS A NEW INPUT BUFFER. GETNEW, WHICH CONTAINS A DUPLICATE PIECE OF CODE AS THAT IN RGETC, MERELY PICKS UP THE NEXT CHARACTER FROM THE INPUT BUFFER AND UPDATES THE INPUT BUFFER POINTERS. GETOLD OBTAINS THE PREVIOUS INPUT BUFFER CHARACTER, BUT DOES NOT CHANGE THE POINTERS. RUSER IS A DUMMY ROUTINE AND IS INCLUDED TO ALLOW A USER TO INTERROGATE THE OUTPUT BUFFER BEFORE IT IS FLUSHED. MOVCHR MOVES WORDS FROM ONE ARRAY TO ANOTHER AND FINALLY, EOFDET IS CALLED ONCE EVERY END OF LINE TO CHECK FOR END OF LOGICAL FILE.

EOFDET IS INTENDED TO BE MODIFIED OR RE-WRITTEN FOR A SPECIFIC SITE. ITS PURPOSE IN LIFE IS TO INTERROGATE THE END OF FILE DETECTION BUFFER TO DETECT THE END OF A LOGICAL JOB. A FLAG IS SET TO INDICATE THIS EVENT.

THE INSERT FILE RJEGRTS>RECCOM HAS A ONE LINE DESCRIPTION FOR EACH VARIABLE UNIQUE TO GRTS RECEIVE. RJEGRTS>RPARAM CONTAINS PARAMETER DEFINITIONS AND RJECOM>RSTATS CONTAINS P50 ALL THE VARIABLES USED FOR GATHERING STATISTICS.

8.1.3.2 INPUT AND OUTPUT BUFFERS (SEE FIG.1.3)

THE INPUT BYTE POINTER, MSIBIT, IS TRUE WHEN THE NEXT CHARACTER REQUIRED IS IN THE LEFT HAND (MOST SIGNIFICANT) BYTE. IF MSIBYT IS TRUE THEN CIP MUST BE INCREMENTED TO PICK UP THE NEXT WORD TO BE PROCESSED. WHEN MISBYT GOES FALSE AND THE CHARACTER OBTAINED, CIP IS CHECKED TO SEE IF THE BUFFER IS EXHAUSTED. IF IT IS THEN CIP IS SET TO ZERO TO ENFORCE A NEW BUFFER TO BE READ THE NEXT TIME A CHARACTER IS REQUIRED.

THE OUTPUT BUFFER IS HANDLED SLIGHTLY DIFFERENTLY. COP POINTS TO THE WORD THAT CONTAINS THE NEXT FREE CHARACTER. THAT CHARACTER IS POINTED AT BY LSOBYT. WHEN LSOBYT GOES FALSE, COP IS CHECKED AGAINST 1024 WORDS. IF IT IS GREATER, THEN THE ENTIRE BUFFER IS FLUSHED TO THE OUTPUT FILE AND COP AND LSOBYT RESET.

8.1.3.3 DESCRIPTION OF PROGRAM MODULES

THIS SECTION EXPLAINS THE RELATIONSHIP BETWEEN SUBROUTINES AND WHAT EACH SUBROUTINE IS RESPONSIBLE FOR.

SUBROUTINE RECV DOES NOT EFFECT THE DATA STRUCTURE SUFFICIENTLY TO WARRANT A FULL DESCRIPTION. IT OPENS AND CLOSES FILES AND CHECKS FLAGS TO SEE IF LOGICAL OR PHYSICAL ENDS OF FILES HAVE BEEN FOUND. SEVERAL CALLS MAY BE MADE ON RECV TO PROCESS ONE FILE AND THE WORKSTATION OPERATOR MAY WISH TO END RECEIVE PROCESSING DURING CONVERSION. THIS IS SIGNIFIED BY WLINE BEING SET TO ZERO. ONLY WHEN IT IS CONVENIENT TO DO SO DOES RECV SET WLINE TO -1 TO TELL THE WORKSTATION IT HAS FINISHED.

RPUTFL IS CALLED BY RECV. IT PICKS UP A CHARACTER FROM IPBUF BY MAKING AN INTERNAL CALL TO RGETC.

RGETC MANIPULATES THE INPUT BUFFER, IPBUF AND ITS ASSOCIATED POINTERS. A CALL ON RGETC DOES NOT ALWAYS NEED TO ACCESS IPBUF AS THE NEXT CHARACTER OBTAINED MAY BE A CONSEQUENCE OF A PREVIOUS CHARACTER. FOR INSTANCE, THE PREVIOUS CHARACTERS MAY HAVE INDICATED COMPRESSION OF THE CHARACTER 'S'. IF IT IS REPEATED SIX TIMES, THEN SIX CALLS ARE MADE ON REGTC. A SLEW CHARACTER FOUND HAS THE SAME EFFECT. THE MAXIMUM INDIVIDUAL LINE SLEW CAN BE 15. IN THIS CASE, FIFTEEN CALLS ARE MADE ON RGETC TO RECOVER THESE NEW LINE CHARACTERS. BOTH THESE SPECIAL CASES ARE FLAGGED BY A LOGICAL VARIABLE 'SPECAL'.. IF TRUE THEN A CHECK IS MADE ON WHICH CHARACTER IS REQUIRED (NEW LINE OR COMPRESSED CHARACTER). THERE IS ALSO A COUNT ASSOCIATED WITH EACH CONDITION. THIS IS DECREMENTED UNTIL ZERO WHEN THE NEXT CHARACTER SHOULD COME FROM THE INPUT BUFFER.

EACH CHARACTER FOUND IN THE INPUT BUFFER IS CHECKED FOR RS, ETX, US AND SYNC. THE SYNC CHECK IS REDUNDANT WHEN A FORMATTER IS IN OPERATION. ON FINDING AN RS THE MEDIA CODE (MC) IS LOOKED AT TO ENSURE WE ARE

DEALING WITH A PRINT FILE. THIS BEING THE CASE GETOLD IS CALLED TO OBTAIN THE PREVIOUS CHARACTER WHICH IS, IN FACT, THE SLEW. THIS IS THEN TRANSLATED INTO A FORM FEED OR A NUMBER OF NEW LINES AND 'SPECAL' SET WHEN NECESSARY. THE OUTPUT BUFFER IS BACKED UP BY ONE CHARACTER TO REPLACE THE SLEW CHARACTER AND PROCESSING CONTINUES BY LOOKING AT THE NEXT CHARACTER AS THIS MAY WELL BE AN RS OR ETX AND FLAGS ARE SET ACCORDINGLY. PROCESSING NOW CARRIES ON AS NORMAL.

IF A US CHARACTER IS FOUND THEN THE PREVIOUS AND FOLLOWING CHARACTERS ARE OBTAINED TO GET THE CHARACTER AND COUNT RESPECTIVELY.

AN ETX FOUND AT THIS POINT MERELY INDICATES A NEW INPUT BUFFER IS REQUIRED AND IS CAUSED BY A SPLIT RECORD. RGETC IS RESPONSIBLE FOR MAINTAINING A FULL INPUT BUFFER SO THAT THE CALLING ROUTINE NEEDS ONLY CONCERN ITSELF WHEN AN END OF FILE IS DETECTED.

ON RETURNING TO RPUTFL A COUPLE OF CHECKS ARE MADE AND THE CHARACTER PLACED IN THE OUTPUT BUFFER BY AN INTERNAL CALL TO RPUTC. RPUTC IS MUCH SIMPLER THAN RGETC AND MERELY UPDATES THE OUTPUT BUFFER AND POINTERS, CHECKING FOR END OF BUFFER. IF THIS IS FOUND OBUFFL IS SET TRUE, CAUSING THE BUFFER TO BE FLUSHED IN RPUTFL ON RETURN.

EOFDET IS CALLED EVERY TIME AN RS IS FOUND. THE END OF FILE DETECTION BUFFER, ENDBUF, IS INTERROGATED FOR A SPECIFIC CHARACTER STRING OR STRINGS. WHEN FOUND THE FLAG ENDLOG IS SET TRUE TO INDICATE THE END OF A LOGICAL JOB. ENDBUF CONTAINS THE LAST LINE OF CHARACTERS FROM THE RECEIVE FILES, ONE CHARACTER PER WORD, LOCATED IN THE RIGHT HAND BYTES. THE BUFFER STARTS AT [ ] AND ENDS AT [ENDLPT]. IT IS THE USERS RESPONSIBILITY TO INCLUDE THE CORRECT CODE FOR EOFDET AS THE STRING INDICATING END OF JOB IS SITE DEPENDENT. THE VARIABLE EOFKEY IS DEFINED IN THE SITE DEFINITION FILE AND INDICATES WHICH SITE IS BEING USED. FOR INSTANCE, SITEA AND SITEB MAY BOTH BE GRTS SITES BUT HAVE DIFFERENT CHARACTER STRINGS REPRESENTING THE END OF A JOB. BOTH SITES MAY BE ACCESSED VIA SMLC LINE ZERO. IN THIS CASE SITEA'S SITE DEFINITION FILE WILL CONTAIN EOFKEY=1 AND SITEB'S SITE DEFINITION FILE WILL CONTAIN EOFKEY=2.

8.2 TRANSMIT PROCESSING

THE TRANSMIT PROCESSOR PERFORMS THE CONVERSION OF PRIME FILES TO GRTS PROTOCOL. AT PRESENT IT CATERS FOR STANDARD ASCII FILES, CONCAT FILES AND HANDLES COMPRESSION AND SPLIT RECORDS AS AN OPTION.

8.2.1 THE TRANSMIT FILE

THE TRANSMIT FILE HAS TO BE FORMATTED SUCH THAT THE SYMBIONT CAN PICK UP THE INFORMATION REQUIRED WITHOUT TOO MUCH EFFORT. IT IS FORMATTED IN BLOCKS SO THAT EACH BLOCK CAN BE TRANSMITTED IN ONE GO. FIG. 8.2.1 SHOWS THIS BLOCK STRUCTURE AND FIG. 8.2.2 SHOWS THAT EACH BLOCK MUST BEGIN WITH A FORMAT CODE. THIS CONTAINS INFORMATION ABOUT WHAT THE RECORD IS, I.E., TEXT OR SPECIAL CONTROL RECORD, COMPRESSION AND SPLIT RECORDS. THE MEDIA CODE IS SET TO SHOW STANDARD ASCII RECORDS. ALL THE DATA BLOCK ARE SEPARATED BY AN RS FOLLOWED BY A MC (OR ETX). ETX OCCURS WHEN TO ADD ANOTHER RECORD WOULD CAUSE THE BLOCK SIZE TO BE

GREATER THAN 324 CHARACTERS OR AT THE END OF A FILE. SPECIAL CONTROL RECORDS (THOSE BEGINNING WITH A $) ARE SENT ONE PER BLOCK.

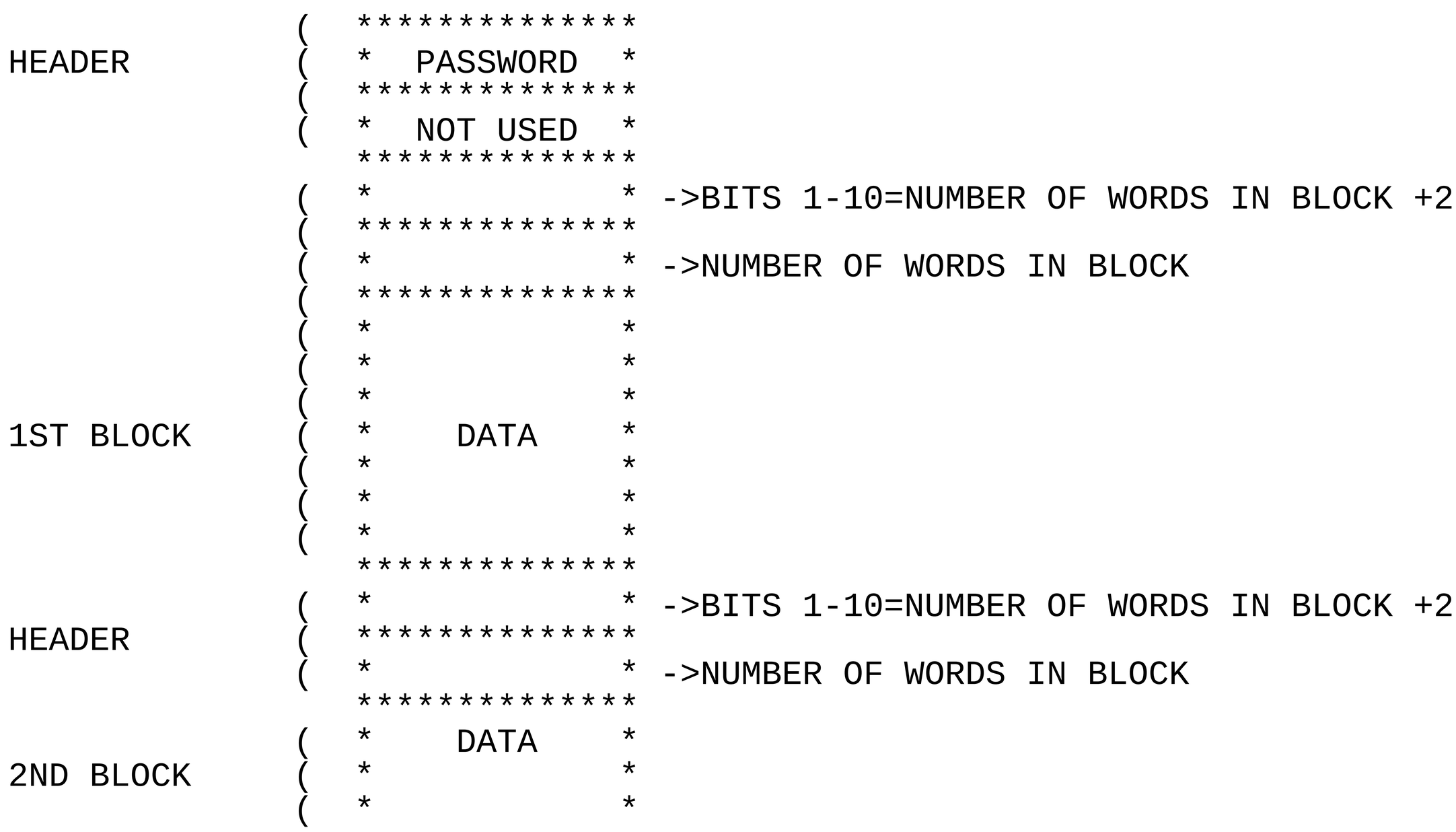


FIG. 8.2.1 TRANSMIT FILE FORMATS

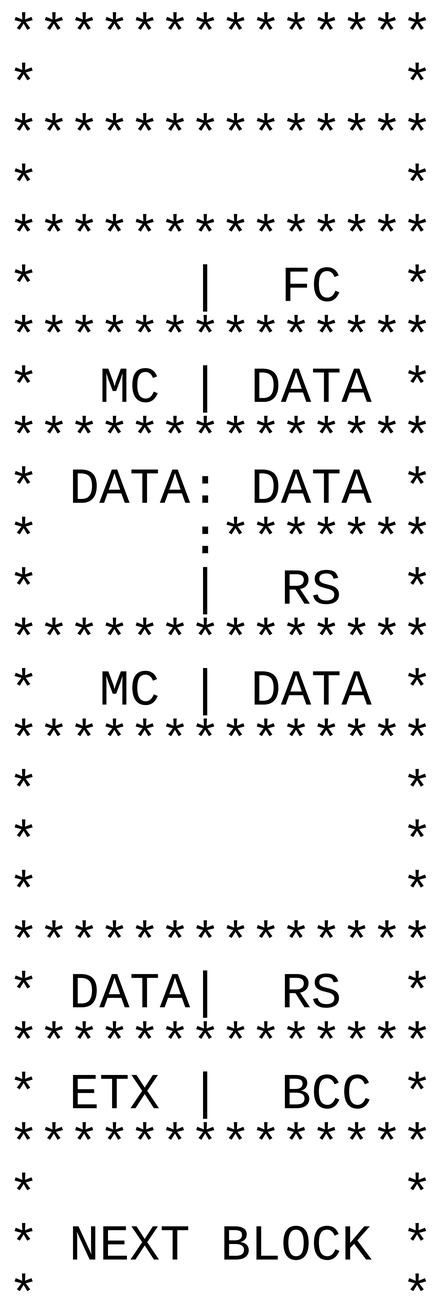


FIG. 8.2.2 TRANSMIT DATA BLOCK

8.2.3 DATA STRUCTURE

ALL TRANSMIT MODULES ARE COMPILED USING A PARAMETER NAMED NLINE IN FILE RJECOM>PARAM WHICH CONTROLS THE NUMBER OF SMLC LINES THE ENTIRE WORKSTATION CAN HANDLE AT ANY ONE TIME. IN THE FOLLOWING SECTIONS REFERENCE WILL NOT BE MADE TO A VARIABLES DEMENSION, MERELY ITS PURPOSE.

8.2.3.1 PROGRAM MODULES

TRANSMIT PROCESSING CONSISTS OF THE FOLLOWING SUBROUTINES:

```
     TRANSM      IN FILE TRANSM
     TPUTFL      IN FILE TPUTFL
     TOOBIG     )
     CPYRND     )
     ENDLIN     )
     BEGREC     ) IN FILE TSUBS
     TGTNEW     )
     TGTBUF     )
     TINIT      )
     ENDREC     )
```

NB. TPUTFL CONTAINS TWO INTERNAL SUBROUTINES TGETC AND TPUTC. THEY ARE INTERNAL TO SAVE THE OVERHEAD OF A PCL INSTRUCTION. BOTH THESE ROUTINES ARE CALLED APPROXIMATELY ONCE PER CHARACTER.

TRANSM IS THE LINK BETWEEN THE WORKSTATION (OR RJ STATION) AND THE TRANSMIT PROCESSOR. IT HAS THREE ARGUMENTS WHICH ARE THE FILENAME (TREENAME FOR RJ), NAMELENGTH AND A THIRD ARGUMENT WHICH IS AT PRESENT UNUSED. THIS SUBROUTINE ENSURES INPUT AND OUTPUT FILES ARE OPEN AND WRITES A TWO WORD HEADER TO A NEW TRANSMIT FILE. THE FILE IS PROCESSED BY CALLING TPUTFL AND ON RETURN, TRANSM CLOSES ALL FILES AND PRINTS STATISTICS WHERE REQUIRED.

TPUTFL IS IN THE ROUTINE CALLED TO CONVERT THE OPENED INPUT FILE. IT ACCOMPLISHES THIS BY CALLING RGETC TO GET A CHARACTER FROM THE INPUT FILE, CHECKS FOR COMPRESSION AND SPECIAL CONTROL RECORDS, AND OUTPUTS TO THE OUTPUT BUFFER BY A CALL TO RPUTC.

ENDLIN IS CALLED AT THE END OF A LINE TO DECIDE IF THE TRANSMIT BLOCK HAS EXCEEDED 324 CHARACTERS, IF SO TOOBIG IS CALLED. BEGREC AND ENDREC HANDLE THE HEADERS ETC. REQUIRED AT THE BEGINNING AND ENDING OF TRANSMIT BLOCKS, (SORRY FOR THE AMBIGUITY OF NAMES). TGTNEW OBTAINS A NEW CHARACTER FROM THE INPUT BUFFER AND IS MERELY A REPEAT OF A SMALL SECTION OF CODE IN TGETC. TGTBUF GETS A NEW INPUT BUFFER AND TINIT IS CALLED TO INITIALISE VARIABLES PRIOR TO CONVERSION.

A ONE LINE DESCRIPTION OF ALL THE COMMON VARIABLES MAY BE FOUND IN RJEGRTS>TRACOM. STATISTICS VARIABLES ARE IN RJECOM>STATTX.

8.2.3.2 INPUT AND OUTPUT BUFFERS (SEE FIG.8.2.3)

BOTH BUFFERS MAINTAIN WORD AND BYTE POINTERS FOR THE NEXT AVAILABLE

CHARACTER OR EMPTY BYTE.

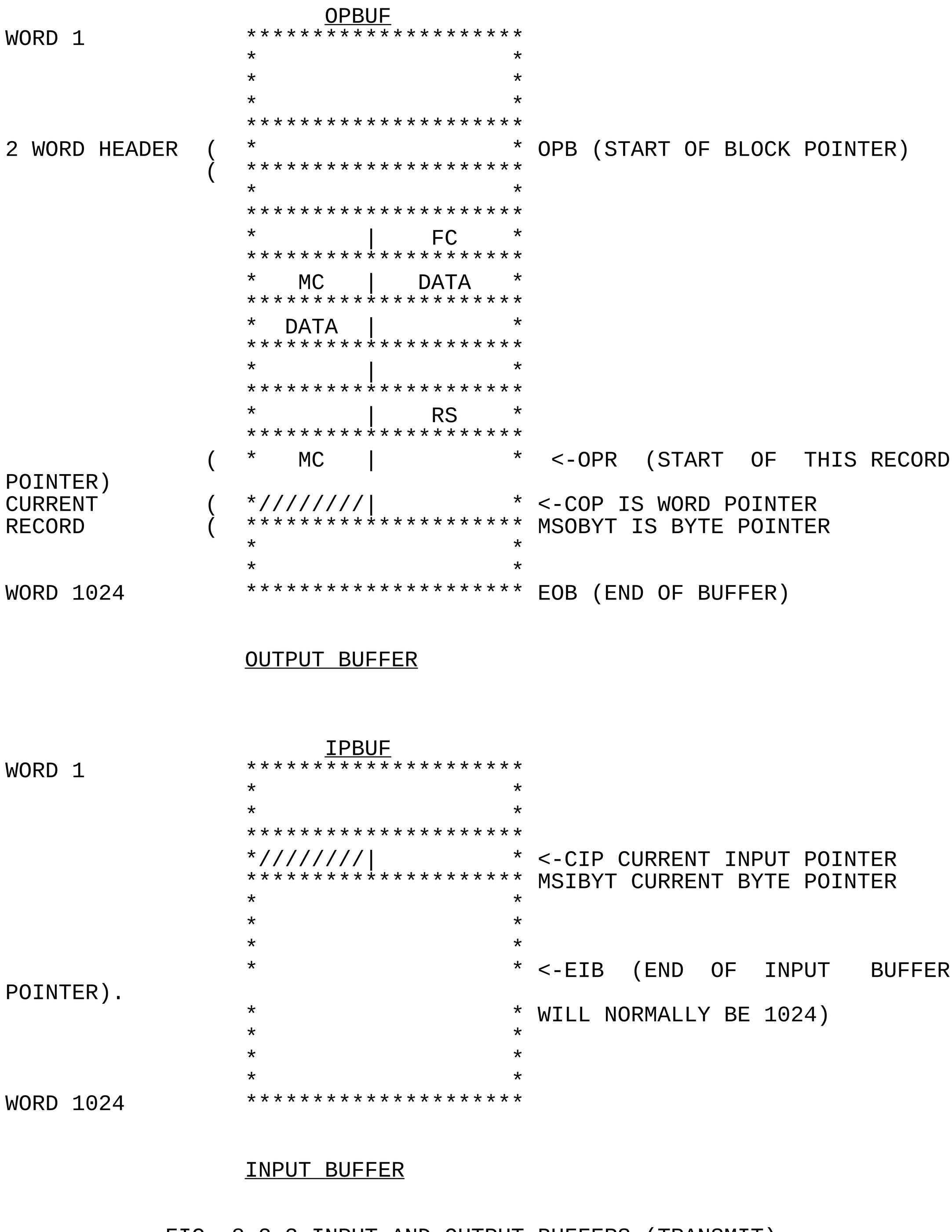


FIG. 8.2.3 INPUT AND OUTPUT BUFFERS (TRANSMIT)

THE INPUT BUFFER, IPBUF, IS SIMPLE IN OPERATION. TGETC IS RESPONSIBLE FOR KEEPING THE BUFFER FULL AND EIB WILL NORMALLY BE 1024 EXCEPT FOR THE LAST BUFFER WHICH CAN BE ANYTHING FROM 0 TO 1024 IN LENGTH.

THE OUTPUT BUFFER IS A MUCH MORE COMPLEX BEAST. WHEN TRANSMIT PROCESSING BEGINS FOR A NEW FILE, WORDS ONE AND TWO ARE LEFT BLANK FOR THE BLOCK HEADER. THE BUFFER IS NOW ADDED TO UNTIL MORE THAN 324 CHARACTERS ARE CONTAINED BETWEEN THE START AND END DATA CHARACTERS. AT THIS POINT ALL LOCATIONS AT AND FOLLOWING OPR ARE MOVED DOWN BY FOUR WORDS. THIS LEAVES ENOUGH ROOM FOR AN ETX, A TWO WORD BLOCK HEADER AND A FC. THESE FOUR LOCATIONS ARE FILLED IN WITH RELEVANT DATA. THE TWO LOCATIONS POINTED TO BY OPB (IN THIS CASE WORDS ONE AND TWO) ARE FILLED IN WITH THE NUMBER OF WORDS FOR THIS BLOCK (SEE FIG.8.2.1). OPB IS RESET TO THE NEW BLOCK HEADER AND OPR TO THE NEW CURRENT RECORD. COP IS ALSO UPDATED. THE NEXT BLOCK MAY NOW BE PROCESSESD.

NOW CONSIDER THE CASE WHEN TO MOVE DOWN BY FOUR WORDS WOULD CAUSE THE BUFFER TO OVERFLOW.

TO OVERCOME THIS, THE ROUTINE CPYRND IS CALLED. THIS FLUSHES THE OUTPUT BUFFER UP TO OPB-1, AND COPIES OPB THROUGH COP TO LOCATIONS ONE AND ONWARDS. ALL THE POINTERS ARE RESET AND PROCESSING CONTINUES. THE BUFFER CANNOT BE FLUSHED PAST OPB BECAUSE THE NUMBER OF WORDS FOR THE CURRENT BLOCK ARE NOT KNOWN AT THIS TIME.

IF HANDLING SPLIT RECORDS A SLIGHTLY DIFFERENT APPROACH IS TAKEN WHEN A BLOCK BECOMES GREATER THAN 324 CHARACTERS. IN THIS SITUATION, THE BLOCK IS TRUNCATED AT 324 CHARACTERS IN LENGTH AND A NEW BLOCK STARTED. CLEARLY A MORE SIMPLE OPERATION.

8.2.3.3 DESCRIPTION OF PROGRAM MODULES

TRANSM HAS ALREADY BEEN DESCRIBED IN ENOUGH DETAIL.

TPUTFL CONVERTS THE INPUT FILE BY FETCHING A CHARACTER, MANIPULATING IT AND PLACING IT INTO THE OUTPUT BUFFER. BUFFER READING AND WRITING IS NOT UNDER ITS DIRECT CONTROL.

AFTER EVERY CHARACTER IS FETCHED A TEST IS MADE TO SEE IF COMPRESSION IS ALLOWED. IF IT IS, THE CHARACTER IS COMPARED WITH THE LAST ONE (LSTCHR). IF THESE MATCH THEN COUNT IS INCREMENTED AND THE NEXT CHARACTER IS FETCHED. IF THEY DO NOT MATCH THEN THE LAST CHARACTER IS OUTPUT, WITH A COMPRESSION COUNT IF REQUIRED, AND THE NEW CHARACTER SAVED, AGAIN IN LSTCHR. AT THIS POINT THE CODE REJOINS NON COMPRESSION CODE AND TESTS ARE MADE TO ESTABLISH IF THIS IS THE FIRST CHARACTER ON A NEW LINE. IF SO, IT IS TESTED AGAINST A DOLLAR SIGN. IF IT MATCHES, THE CURRENT BLOCK IS TERMINATED, AND A NEW ONE STARTED. (DOLLAR INDICATES SPECIAL CONTROL RECORD). OTHERWISE THE BLOCK IS CONTINUED AS USUAL.

WHEN A NEW LINE CHARACTER IS FOUND, ENDLIN IS CALLED TO TERMINATE THE CURRENT LINE.. THIS IS DONE BY APPENDING AN RS AND MC TO THE OUTPUT BUFFER. A CHECK HAS TO BE MADE FOR THE BUFFER OVERFLOWING.

EACH CHARACTER IS OBTAINED FROM THE INPUT FILE BY TGETC. THIS ROUTINE

CHECKS EACH CHARACTER FOR RHT (RELATIVE HORIZONTAL TAB) AND RVT (RELATIVE VERTICAL TAB). ON DETECTION THE FLAG SPECAL IS SET TO INDICATE ON THE NEXT CALL TO TGETC THAT THE CHARACTER TO BE RETURNED IS NOT IN THE INPUT BUFFER BUT IS A SPACE OR NEW LINE.

TGETC IS MADE MORE COMPLEX BEAUSE IT HANDLES THE SPECIAL CONCAT FILE.

A CONCAT FILE CAN BE A MIXTURE OF TEXT AND JCL MADE UP LINE BY LINE OR FILE BY FILE BY THE USER. THE FILE CONSISTS OF 002, 002 OCTAL IN THE FIRST TWO BYTES INDICIATING A CONCAT FILE FOLLOWED BY A ONE WORD TYPE CODE, FOLLOWED BY A TWO WORD WORD COUNT. THE TYPE AND COUNT HEAD EACH CONCAT BLOCK. TGETC IS TOLD BY TGTBUF (THE ROUTINE THAT GETS A NEW INPUT BUFFER) THAT THIS FILE IS A CONCAT FILE. TGETC IS THEN ENTIRELY RESPONSIBLE FOR MAKING SURE THAT A CORRECT CONCAT FORMAT EXISTS.

NO OTHER ROUTINE REALISES CONCAT IS IN OPERATION. CERTAIN OPERATIONS MEANINGFUL ONLY TO CONCAT ARE ALWAYS CARRIED OUT, E.G., DECREMENTING THE CONCAT COUNT. HOWEVER, IF A BRANCH IS TAKEN BECAUSE OF A CERTAIN CONDITION, E.G., COUNT GOES TO ZERO, THE CONCAT FLAG IS CHECKED AND IF CONCAT IS NOT IN OPERATION, THEN NORMAL PROCESSING CONTINUES. THIS IS QUICKER THAN ALWAYS CHECKING THAT CONCAT IS IN OPERATION BEFORE PERFORMING A CONCAT FUNCTION.

TPUTC IS RESPONSIBLE FOR ADDING CHARACTERS TO THE OUTPUT BUFFER AND ENSURING THAT IT DOES NOT OVERFLOW.

TOOBIG HANDLES THE CONDITION OF A BLOCK BEING GREATER THAN 324 CHARACTERS AS DESCRIBED IN SECTION 8.2.3.2.

THE PURPOSE OF ALL OTHER ROUTINES HAVE BEEN DESCRIBED IN SECTION 8.2.3.2.

SECTION 9. THE HASP TRANSMIT/RECEIVE MODULES

INTRODUCTION

THIS SECTION IS CONCERNED WITH THE HOUNSTON AUTOMATIC SPOOLING PROGRAM (HASP) THAT IS SUPPORTED BY IBMS JES2, AND ITS IMPLEMENTATION ON PRIME.

THERE ARE 3 SUB-SECTIONS TO THIS SECTION, AS FOLLOWS:-

1. TRANSMIT MODULE
2. RECEIVE MODULE

APPENDIX A. MULTI-LEAVING IN RJE

IN BOTH TRANSMIT AND RECEIVE THE FOLLOWING PROGRAMMING TECHNIQUE IS USED:-

THE TWO SUBROUTINES MOST FREQUENTLY USED ARE THE GET AND PUT CHARACTER ROUTINES. (TGETC, TPUTC FOR TRANSMIT AND RGETC, RPUTC FOR RECEIVE). ORIGINALLY THESE WERE 'CLOSED' SUBROUTINES AND WERE CALLED BY:-

```
     CALL TGETC (CHAR) 1* FOR EXAMPLE
```

AS EACH CALL TOOK APPROX. 25US TO SET UP, THIS APPROACH WAS CONSIDERED UNREASONABLE, AND SO A SERIES OF 'OPEN' SUBROUTINE CALLS REPLACED THEM. TO ACCOMMODATE THIS (AND TO AID READABILITY) A VARIABLE IS INITIALISED TO THE ADDRESS OF THE FIRST LOCATION IN THE 'OPEN' SUBROUTINE.

```
     E.G.   TGETC$ = $11000 /* (11000 IS THE LABEL AT THAT LOCATION)
```

THEN, WHEREVER TGETC WOULD HAVE BEEN CALLED, THE FOLLOWING CODE IS USED INSTEAD.

```
         RTADR = $XXXX /*(XXXX IS RETURN LABEL)
         GOTO TGETC$
   XXXX CONTINUE
```

THE CHARACTER IS RETURNED IN VARIABLE CHAR WHICH MAY THEN BE ASSIGNED TO ANY OTHER COMPATIBLE VARIABLE.

SIMILARLY, TO OUTPUT A CHARACTER, THE CALL

```
     CALL TPUTC (CHAR)
```

IS REPLACED BY CHAR = CHAR

```
         RTNADR = $XXXX
         GOTO TPUTC$
    XXX CONTINUE
```

THE 'RETURN' STATEMENT THAT WAS USED IN THE 'CLOSED' SUBROUTINE IS REPLACED BY 'GOTO RTNADR'.

9.1 THE TRANSMIT MODULE

THE TRANSMIT MODULE IS ONLY ONE SUBROUTINE (TRANSM) BUT WITHIN THAT ROUTINE ARE TWO 'OPEN SUBROUTINES', ONE AT LABEL 11000 (REFERRED TO AS TGET$ AND THE OTHER AT 21000 (TPUTC$).

TRANSM IS CALLED BY THE WORKSTATION OR THE SEND UTILITY. IN THE CASE OF THE FORMER WHENEVER AN OPMODE LINE OR AN OPMESS MESSAGE IS TO BE SENT TO THE MAINFRAME, BUT ITS MAIN USE IS WHEN EITHER UTILITY WISHES TO SEND A FILE TO THE MAINFRAME (BY USING THE SEND OR CONCAT COMMAND).

THE ACTIONS INVOKED (WHICH ARE DIFFERENT FOR EACH COMMAND) ARE DETERMINED BY THE 3 PARAMETERS OF THE TRANSM CALL AND ONE OTHER FACTOR WHICH WILL BE DISCUSSED LATER. CONSIDER THE CALL

```
CALL TRANSM (IPNAME,IPLNTH,TYPE)
```

WHERE IPNAME IS THE NAME OF THE FILE TO BE SENT (IN THE CASE OF CONCAT AND OPMESS THIS IS A DUMMY FILE CREATED BY THE WORKSTATION). IN THE CASE OF OPMODE, IPNAME HOLDS THE ACTUAL MESSAGE

IPLNTH IS THE LENGTH OF THE NAME HELD IN IPNAME (IN THE CASE OF OPMODE THIS IS THE LENGTH OF THE MESSAGE TO BE OUTPUT).

TYPE:0 - NORMAL TRANSMIT OF CONCAT FILE. THE FILE NAME IS HELD IN IPNAME AND THE RESULTING OUTPUT IS TO GO IN THE NEXT 'T' FILE.

:1 - OPMODE FILE. THE MESSAGE TO BE SENT IS HELD IN IPNAME AND THE RESULTING OUTPUT IS TO GO INTO 'Z' FILE.

:2 - OPMESS FILE. THE FILE NAME IS HELD IN IPNAME AND THE RESULTING OUTPUT IS TO GO INTO 'Z' FILE.

THE OTHER FACTOR MENTIONED EARLIER DETERMINES WHETHER OR NOT A CONCATENATION HAS TAKEN PLACE BY THE WORKSTATION. IF CONCAT HAS TAKEN PLACE THEN THE WORKSTATION PLACES THE UNIQUE VALUE I.E. WORD 1 OF THE FILE ':001002'.

IF THIS UNIQUE VALUE IS PRESENT THEN THE FOLLOWING THREE WORDS ARE CONSIDERED AS A HEADER, AS FOLLOWS:

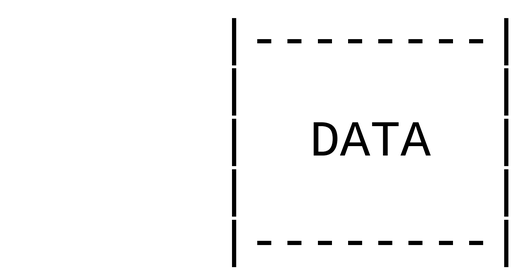
WORD 1 - TYPE (0 = TEXT, 1 = BINARY): NOTE HASP WILL TREAT JCL AND TEXT THE SAME. THE CONTEXT WILL DETERMINE ITS INTERPRETATION AT THE MAINFRAME.

WORDS 2 AND 3 - A 32 BIT WORD COUNT OF THE NUMBER OF WORDS TO THE NEXT 3 WORD HEADERS. NOTE* AS THE HEADER ONLY APPEARS AFTER A NEWLINE CHARACTER, THIS COUNT NEED NOT BE COMPARED UNTIL THE END OF PREVIOUS RECORD IS DETECTED.

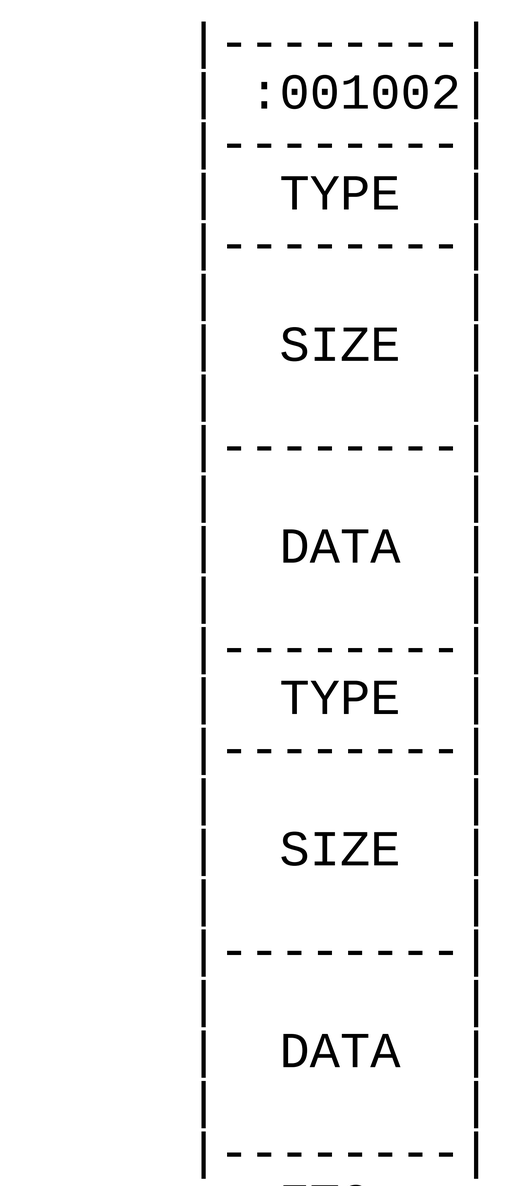
TO SUMMARISE THE DIFFERENT TYPES OF INPUT FILE TO BE PROCESSED THE FOLLOWING DIAGRAMATIC REPRESENTATION IS USED.

SEND: (IPNAME HOLDS THE FILENAME. IPLNTH FILENAMES LENGTH TYPE = 0).

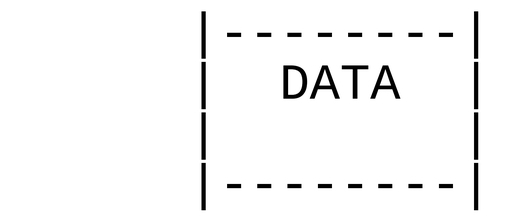
OUTPUT GOES TO NEXT 'T' FILE).



CONCAT : (IPNAME HOLDS THE NAME OF THE DUMMY FILE PRODUCED BY THE WORKSTATION, IPLNTH THE FILENAMES LENGTH TYPE = 0. OUTPUT GOES TO NEXT 'T' FILE).



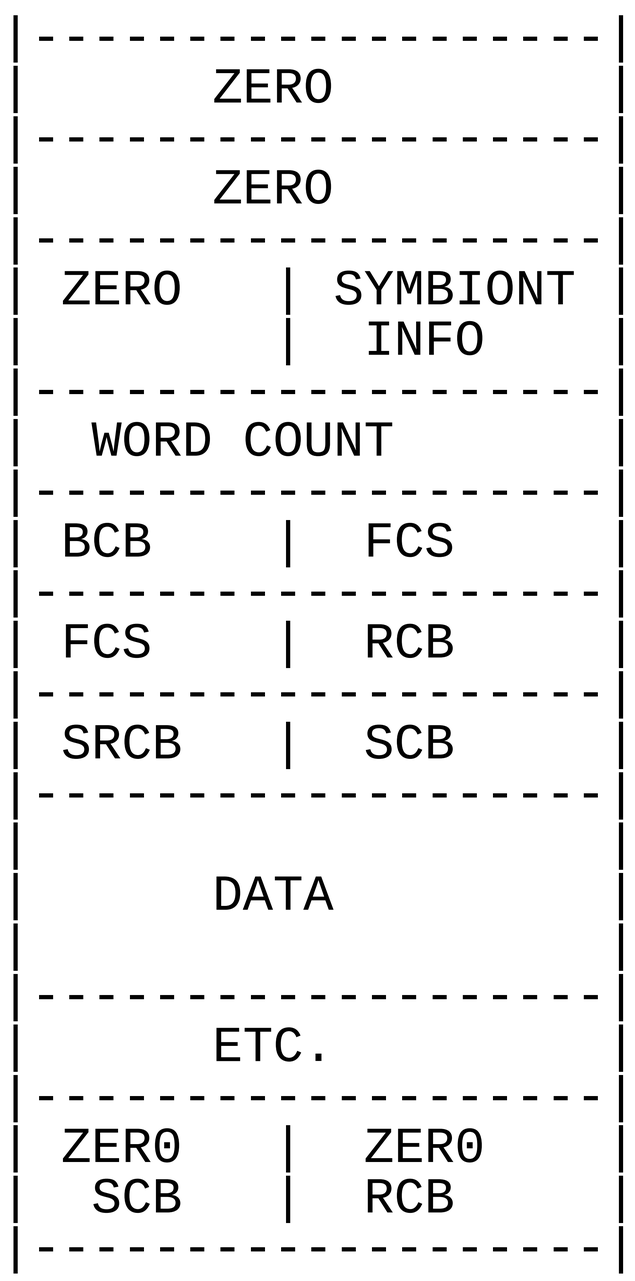
OPMESS : (IPNAME HOLDS THE NAME OF THE DUMMY FILE PRODUCED BY THE WORKSTATION. IPLNTH THE FILENAMES LENGTH TYPE= 2. THE OUTPUT GOES TO THE 'Z' FILE).



OPMODE : (IPNAME HOLDS THE MESSAGE TO BE SENT. IPLNTH HOLDS THE LENGTH OF THE MESSAGE. TYPE = 1. THE OUTPUT GOES TO THE 'Z' FILE).

HAVING DESCRIBED THE FORMAT OF THE INPUT TO BE TREATED, THE OUTPUT MAY

NOW BE CONSIDERED. APPENDIX A DESCRIBES THE FORMAT OF THE BLOCKS AND THIS (ALONG WITH NECESSARY HEADERS) IS SHOWN BELOW.



THE FOLLOWING IS AN EXAMPLE OF A TYPICAL TRANSMISSION BLOCK.

BYTES		
	DLE	)
	STX	)
	BCB	) LEFT FOR SYMBIONT TO FILL
	FCS	)
	FCS	)
	RCB	RECORD CONTROL BYTE FOR RECORD 1
	SRCB	SUB-RECORD CONTROL BYTE FOR RECORD 1
	SCB	STRONG CONTROL BYTE FOR RECORD 1
	DATA	CHARACTER STRING
	SCB	STRING CONTROL BYTE FOR RECORD 1
	DATA	CHARACTER STRING
	SCB	TERMINATING SCB FOR RECORD 1
	RCB	RCB FOR RECORD 2
	SRCB	SRCB FOR RECORD 2
	SCB	SCB FOR RECORD 2
	DATA	CHARACTER STRING
	SCB	TERMINATING SCB FOR RECORD 2
	RCB	TRANSMISSION BLOCK TERMINATOR
	DLE	)
	ETB	) LEFT FOR SYMBIONT TO FILL

HAVING DESCRIBED THE INPUT AND OUTPUT FORMATS, THE BEST WAY TO DESCRIBE HOW THESE ARE TRANSFORMED FROM ONE TO THE OTHER IS FROM THE INSIDE OUT. THAT IS TO SAY, IT IS EASIER IF THE SCB PRODUCTION IS DESCRIBED FIRST AND THEN COMMENT ON ANY "BELLS AND WHISTLES" AS AND WHEN THEY ARISE.

THE STRING CONTROL BYTE IS DEFINED IN APPENDIX A, BUT ESSENTIALLY IT IS USED AS A MEANS OF COMPRESSING DATA TO INCREASE DATA TRANSFER EFFICIENCY. IT FALLS INTO 1 OF 4 CATEGORIES.

1. "THE NEXT 'N' CHARACTERS ARE DIFFERENT".
2. "THE NEXT CHARACTER IS TO BE REPEATED 'N' TIMES".
3. "INSERT 'N' BLANKS INTO THE RECORD HERE".
4. "THIS IS THE END OF RECORD".

APART FROM '4' WHICH IS PRODUCED UPON THE DETECTION OF A NEWLINE (:212) CHARACTER, THE SCBS ARE PRODUCED RETROSPECTIVELY. THAT IS THEY CANNOT BE PRODUCED UNTIL A CHANGE FROM ONE OF THE 1ST THREE TYPES (ABOVE) TO ANOTHER, OCCURS.

THIS IS ACHIEVED IN TRANSM WITH THE AID OF TWO 'LOGICALS' DPLCTE AND BLANK, AND THE NEXT TWO UNPROCESSED CHARACTERS FROM THE INPUT STREAM - CHAR1 AND CHAR2 RESPECTIVELY.

THE FLAG DPLCTE IS SET .TRUE. IF THE CURRENT CHARACTER STREAM CONTAINS DUPLICATE CHARACTERS AND BLANK IS SET .TRUE. IF THE DUPLICATE CHARACTERS ARE BLANK (I.E. :240). THE VALUE OF BLANK HAS NO MEANING IF DPLCTE .EQ. .FALSE.

BY USING THE DPLCTE FLAG A TEST CAN BE MADE TO CHECK FOR THE FOLLOWING CONDITIONS.

```
     IF (DPLCTE) GOTO 4000
        (CHAR1.NE.CHAR2)GOTO 5000 /* ANOTHER DIFFERENT
         GOTO 6000    /* 1ST OF A DUPLICATE STRING
4000 CONTINUE
        IF (CHAR1.NE.CHAR2) GOTO 7000/* END DUPLICATE STRING
           GOTO 8000              /*ANOTHER DUPLICATE
```

WHERE A NEWLINE IS DETECTED THE FLAG ENDREC IS SET AND EITHER THE 'ANOTHER DUPLICATE' (8000) OR 'ANOTHER DIFFERENT' (5000) SECTION OF CODE IS RE-ENTERED TO TIDY UP AND THIS IS FOLLOWED BY THE OUTPUT OF AN SCB OF VALUE 0 (ZERO SCB) WHICH SIGNIFIES END OF RECORD. BECAUSE THESE TWO SECTIONS CAN RETURN EITHER TO THE MAIN LOOP FOR HANDLING SCBS OR TO THE 'END OF RECORD' HANDLING CODE, THE SAME CONCEPT OF 'OPEN SUBROUTINES' IS USED AS THAT ADOPTED FOR THE TGETC$ AND TPUTC$ ROUTINES. FOR THIS CODE HOWEVER, THE VARIABLE RTNADR HOLDS THE RETURN ADDRESS.

HAVING OUTPUT A ZERO SCB THE NEXT RECORD IS TO BE CONSIDERED AND HENCE DPLCTE AND BLANK NEED TO BE RE-INITIALISED. THIS IS ACHIEVED BY EXAMINING THE 1ST TWO CHARACTERS OF THE RECORD AND SETTING THE FLAGS ACCORDINGLY. SO THAT THE GENERAL LOOP FOR SCBS NEED NOT HAVE ANY SPECIAL TESTS WITHIN IT, THE INPUT BUFFER POINTER IS MOVED BACK ONE. ON ENTRY TO THE LOOP, THEREFORE, DPLCTE AND BLANK ARE CORRECT, THE LAST CHARACTER OF THE RECORD IS IN PLACE AND THIS ENABLES THE MAIN SCB LOOP TO PICK UP THE SECOND CHARACTER OF THE RECORD (AGAIN) AND TREAT BOTH

CHARACTERS AS IF THEY OCCURRED ANYWHERE IN THE LOOP.

THE SRCB ALWAYS TAKES THE DEFAULT VALUE OF :200 AND THE RCB WILL HAVE A VALUE OF :222 FOR OPERATOR MESSAGES OR :223 FOR THE INPUT RECORDS (I.E. THOSE ACTIONED BY SEND OR CONCAT).

TGETC$

AS MENTIONED EARLIER, THIS PIECE OF CODE (STARTING AT LABEL 11000) BEGAN LIFE AS A SUBROUTINE AND CALLED IN THE 'NORMAL' WAY (I.E. BY A FORTRAN CALL). IT IS NOW AN OPEN SUBROUTINE AND ITS FUNCTION IS TO PROVIDE (IN VARIABLE CHAR) THE NEXT CHARACTER IN THE INPUT BUFFER. IF THIS BUFFER IS EXHAUSTED (I.E. THE BYTE POINTER - IPSET - POINTS PAST THE LAST VALID CHARACTER IN THE BUFFER) THEN THIS ROUTINE READS MORE DATA INTO THE BUFFER FROM THE INPUT FILE. IF BOTH THE INPUT BUFFER AND INPUT FILE ARE EXHAUSTED THEN THE EOFSEND FLAG IS SET .TRUE.

THIS ROUTINE ALSO SKIPS THE RIGHT HAND BYTE OF ANY WORD WHICH HAS, AS ITS LEFT HAND BYTE, THE NEWLINE CHARACTER.

TPUTC$

AS WITH TGETC$ THIS ROUTINE HAS BEEN MADE AN 'OPEN' SUBROUTINE. ITS FUNCTION IS TO PLACE THE CHARACTER 'CHAR' INTO THE OUTPUT BUFFER. WHENEVER THERE ARE ENOUGH CHARACTERS TO FORM A BLOCK THE HEADERS AND TRAILERS ARE MANIPULATED AND THE BLOCK OUTPUT TO DISC.

THE SIZE OF A BLOCK IS DETERMINED BY THE USER 'BLOCK' COMMAND, AND THE NUMBER OF RECORDS WITHIN A BLOCK IS ENTIRELY DEPENDENT UPON THE NUMBERS OF CHARACTERS (INCLUDING RCBS, SRCBS AND SCBS) WITHIN EACH RECORD. BECAUSE OF THIS VARIABILITY, THE FOLLOWING TECHNIQUE IS ADOPTED.

GIVEN 1. RCBPSN HOLDS THE POSITION WITHIN THE CURRENT BLOCK OF THE LAST RCB.
2. OPOSET IS A BYTE POINTER TO WHERE THE NEXT CHARACTER IS TO BE OUTPUT.
3. BLKSZE HOLDS THE BLOCKSIZE ALLOWED FOR THIS INSTALLATION.

THEN 1. IF OPOSET .LE. BLKSZE GOTO 5 /* WILL NOT OVERFLOW.
2. WRITE OUT THE OUTPUT BUFFER UPTO BUT NOT INCLUDING RCBPSN.
3. COPY DATA NOT OUTPUT (I.E. RCBPSN TO OPOSET-1) TO BEGINNING OF OUTPUT BUFFER.
4. REASSIGN OPOSET AND RCBPSN
5. PLACE CHARACTER IN OUTPUT BUFFER AT OPOSET.

THERE ARE SOME "BELLS AND WHISTLES" TO THIS AND THESE ARE SHOWN IN THE CODE.

AT PRESENT, TRANSM PROCESSES AN ENTIRE FILE BEFORE RETURNING CONTROL TO THE WORKSTATION. HOWEVER, AT THE BEGINNING AND END OF THE ROUTINE, THE SET OF VARIABLES ARE ASSIGNED TO ARRAY ELEMENTS (E.G. OPUNIT = OTPUNT(TLINE)). THIS IS TO ENABLE THE NEXT PHASE OF THE TRANSM TO PRODUCE AN INTEGRATED NUMBER OF OUTPUT BLOCKS (BUT NOT NECESSARILY THE ENTIRE INPUT FILE) BEFORE RETURNING TO THE WORKSTATION. THIS MEANS THAT SUCCESSIVE CALLS TO TRANSM MAY BE FOR DIFFERENT LINES AND CONSEQUENTLY DATA SPECIFIC TO A LINE NEED BE KEPT (AS IS THE CASE WITH RECEIVE AT PRESENT).

9.2 RECEIVE MODULE

THE RECEIVE MODULE IS LESS COMPLEX THAN ITS TRANSMIT COUNTERPART AS IT HAS THE MORE COMPLICATED FILE (I.E. THE HASP RECEIVE FILE) TO GUIDE IT THROUGH ITS PROCESS. HOWEVER, LIKE THE TRANSMIT MODULE, THE RECEIVE MODULE HAS ITS GET AND PUT CHARACTER ROUTINES (RGETC$ AND RPLETC$) CODED AS OPEN SUBROUTINES. THE TECHNIQUE HAS BEEN EXTENDED TO PROCESSING OPERATOR MESSAGES AND ALSO TO THE HANDLING OF PRINTER CARRIAGE CONTROL AS DEFINED BY THE SUB-RECORD CONTROL BYTE (SRCB).

THERE IS ONLY ONE FORTRAN SUBROUTINE (OTHER THAN RECV ITSELF) THAT OF EOFDET, WHICH IS DESIGNED SUCH THAT THE USER MAY MODIFY IT TO ACCOMMODATE ANY END OF FILE DETECTION MECHANISMS REQUIRED BY A PARTICULAR INSTALLATION. THIS ROUTINE IS BASED ON A STRING COMPARE BETWEEN THE LAST RECORD IN RECEIVE FILE AND A STANDARD END OF FILE STRING. FACILITIES ARE INCLUDED TO ALLOW A GIVEN NUMBER OF SUCH STRINGS TO OCCUR BEFORE THE END OF FILE, DIFFERENT STRINGS FOR DIFFERENT SITES ON THE SAME EMULATOR AND DIFFERENT STRINGS FOR PRINT AND PUNCH FILES OF THE SHARE SITE. THIS ROUTINE HAS BEEN KEPT SEPARATE FOR EASE OF MODIFICATION.

THE RECV MODULE, WHEN CALLED, ESTABLISHES WHETHER OR NOT AN 'R' FILE IS ALREADY OPEN. IF NOT, THE NEXT ONE IS OPENED AND EXAMINED TO ENSURE IT IS NOT A NULL FILE (AS THIS MAY HAVE BEEN CREATED BY THE RECOVER COMMAND). IF ONE IS ALREADY OPEN THE FILE SYSTEM POINTER IN THAT FILE IS ADJUSTED TO POINT TO THE NEXT UNPROCESSED RECORD.

A BLOCK COUNT IS INITIALISED SO THAT ONLY THE NEXT 10 BLOCKS OF THE FILE WILL BE PROCESSED. NEXT, A ONE WORD COUNT IS PICKED UP (THE COUNT ITSELF IS ONLY RELEVANT IF ZERO, AS THIS INDICATES THE END OF THE CURRENT R FILE). THE BCB AND TWO FCS BUFFERS ARE SKIPPED OVER AND THE RCB IS PICKED UP, THIS SUPPLIES BOTH THE STREAM AND DEVICE TYPE OF THE CURRENT RECORD. AS ONLY ONE STREAM IS ALLOWED FOR EACH DEVICE TYPE A CHECK IS MADE TO ENSURE THAT THIS STREAM DOES NOT DIFFER FOR ONE THAT MAY ALREADY BE OPEN FOR THIS DEVICE.

THE FOLLOWING BYTE IS THE SRCB AND IS ONLY EXAMINED FOR PRINT DATA WHERE IT SUPPLIES CARRIAGE CONTROL INFORMATION EITHER TO BE ACTIONED UPON NOW OR AFTER THE OUTPUT OF THE NEXT RECORD.

IT IS NOW THAT THE MOST FREQUENTLY ACCESSED PART OF THE CODE IS ENTERED, THE DECODING OF THE STRING CONTROL BYTE (SCB). THE ACCEPTABLE VALUES OF THE SCB ARE DESCRIBED IN APPENDIX A. THE PROCESS OF DECODING IT IS STRAIGHTFORWARD AND NEEDS ONLY THE EXPLANATION SUPPLIED WITH THE CODE.

UPON DETECTION OF A ZERO SCB (I.E. ONE WHOSE VALUE IS ZERO) THIS CODE RETURNS TO THE RCB EXAMINATION WHICH IF ITSELF IS ZERO, SIGNIFIES END OF BLOCK AND THE END OF BLOCK COUNTER IS DECREMENTED BY ONE.

WHEN EITHER THE BLOCK COUNT REACHES ZERO OR IF THE INPUT FILE IS EXHAUSTED, AN EXIT PATH IS TAKEN. IN THE CASE OF THE LATTER THE INPUT FILE IS EITHER DETECTED OR HAS ITS NAME CHANGED TO AN 'S' FILE DEPENDING UPON THE SAVE COMMAND.

IN BOTH CASES THE OUTPUT BUFFER IS FLUSHED TO DISC AS A SUBSEQUENT CALL

TO RECV MAY BE FOR A DIFFERENT SITE, THE SAME OUTPUT BUFFER IS USED BY ALL SITES FOR EFFICIENCY.

AS WITH THE FACILITY TO BE SUPPLIED WITH TRANSMIT, (AT A LATER RELEASE) RECEIVE CAN HANDLE UP TO 4 LINES SIMULTANEOUSLY. THIS IS FACILITATED BY HOLDING THE POINTERS TO THE INPUT BUFFER (FOR A GIVEN LINE) ACROSS SUCCESSIVE CALLS IN SEPARATE ARRAYS.

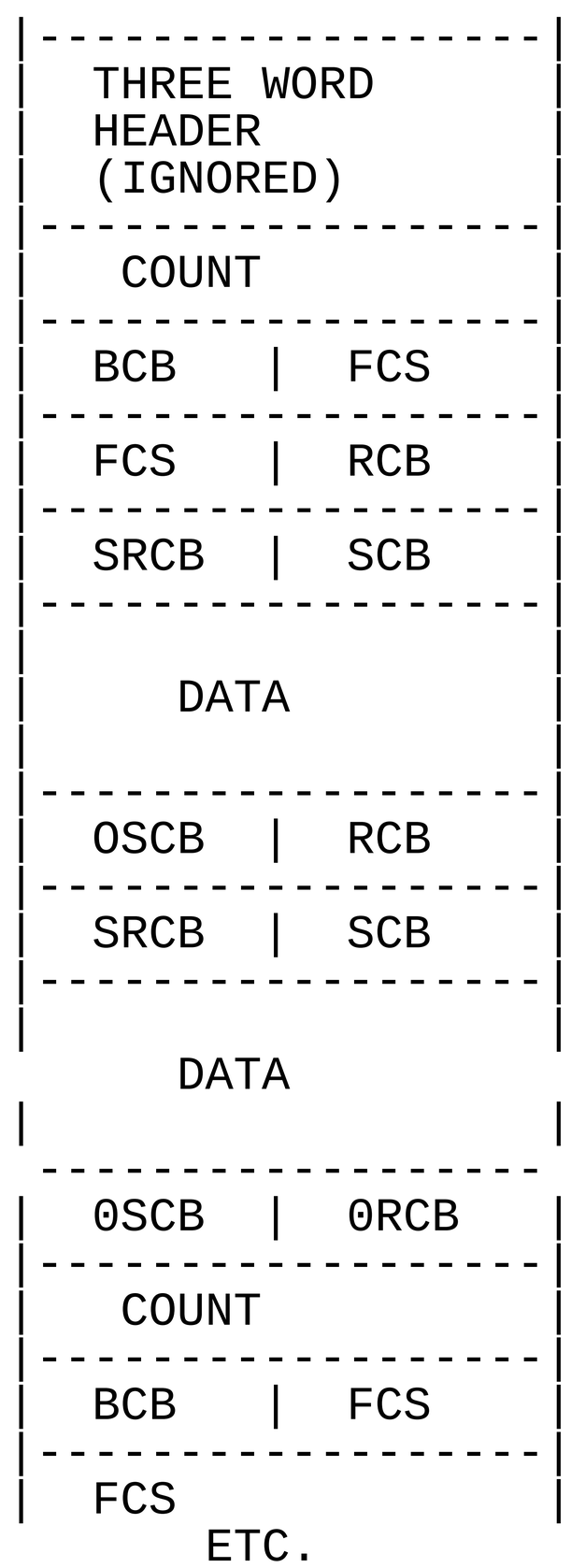
RGETC$

THIS ROUTINE RETURNS THE NEXT CHARACTER REQUIRED (IN VARIABLE CHAR) IRRESPECTIVE OF WHETHER IT IS IN THE BUFFER OR A FILE, I.E. WHENEVER THE BUFFER IS EXHAUSTED THE FILE IS READ, AND THE BUFFER FILLED.

RPUTC$

THIS ROUTINE OUTPUTS THE CHARACTER (HELD IN THE VARIABLE CHAR) TO THE OUTPUT BUFFER. WHENEVER THIS BUFFER IS FILLED IT IS AUTOMATICALLY WRITTEN AWAY TO DISC. THIS CONDITION IS FLAGGED IF THE CURRENT BYTE POINTER (OPOSET) EXCEEDES THE ARRAY SIZE. A WRITE TO DISC MAY BE FORCED BY TEMPORARILY SETTING THE ARRAY SIZE VARIABLE (BLKSZE) TO -1.

THE FORMAT OF THE INPUT FILE IS AS FOLLOWS:



NOTE. COUNT ALWAYS BEGINS ON A WORD BOUNDARY

SECTION 9. APPENDIX A MULTI LEAVING IN RJE

THE HASP MULTI LEAVING PROCESS FOR WHICH THE TRANSMIT AND RECEIVE SUBROUTINES WERE WRITTEN, IS DESIGNED TO WORK AS RJE COMPATIBLE WITH IBM'S JES2. THE PURPOSE OF THIS APPENDIX IS TO DESCRIBE THE FORMAT OF THE FILES WHICH THE TRANSMIT AND RECEIVE SUBROUTINE WILL PROCESS. WHENEVER A DEVIATION FROM THE STANDARD IS EXPECTED THIS WILL BE NOTED.

THE FILES PRODUCED BY THE RECEIVE SUBROUTINE (AND INDEED ACCEPTED AS INPUT BY THE TRANSMIT SUBROUTINE) ARE PRIME FILES OF THE FOLLOWING FORMAT:

EACH RECORD IS COMPRISED OF A SERIES OF CHARACTERS, TERMINATED BY A NEWLINE CHARACTER. IF THE NEWLINE CHARACTER IS ON A WORD BOUNDARY (I.E. LEFT HAND BYTE), THEN THE FOLLOWING BYTE IS REGARDED AS A 'GARBAGE' CHARACTER AND CONSEQUENTLY IGNORED. THIS ENSURES THAT ALL RECORDS BEGIN ON A WORD BOUNDARY.

RECEIVE WILL PRODUCE A FORM FEED (FF$ = :214 IN THE SUBROUTINE RECV) IN RESPONSE TO A REQUEST FOR A 'TOP OF FORM'. APART FROM THIS, THE ONLY ACCEPTABLE NON-PRINTING CHARACTERS (TO EITHER ROUTINE) ARE THE NEWLINE CHARACTER (NL$ = :212) AND THE RELATIVE HORIZONTAL TAB (RHT$ = :221).

THE FILES PRODUCED BY THE TRANSM SUBROUTINE (AND ACCEPTED AS INPUT BY THE RECV SUBROUTINE) ARE A SUBSET OF THOSE DELIVERED BY IBM'S MULTI LEAVING TECHNIQUE.

THE TECHNIQUE ITSELF IS NOW DISCUSSED FOLLOWED BY THREE EXPLICIT BREAKDOWNS OF:

(1) MULTI LEAVING AS PRODUCED BY TRANSMIT
(2) MULTI LEAVING ACCEPTABLE TO RECEIVE AND
(3) MULTI LEAVING UNACCEPTABLE TO EITHER.

THE BASIC ELEMENT USED IN MULTI LEAVING IS THAT OF THE <u>CHARACTER STRING</u>. ONE OR MORE CHARACTER STRINGS MAY THEN BE BUILT UP TO FORM A RECORD. FOR EFFICIENT DATA TRANSMISSION EACH RECORD IS REDUCED TO A SERIES OF CHARACTER STRINGS EACH OF WHICH MAY BE OF ONE OF TWO TYPES.

1. VARIABLE LENGTH, <u>NON-IDENTICAL</u> SERIES OF CHARACTERS.
2. VARIABLE LENGTH, <u>IDENTICAL</u> SERIES OF CHARACTERS.

DUE TO THE HIGH FREQUENCY OF STRINGS COMPRISING ONLY OF SPACE (BLANK) CHARACTERS A SPECIAL CASE OF '2' (ABOVE) IS MADE.

THE TYPE AND LENGTH OF A CHARACTER STRING IS HELD IN AN 8-BIT FIELD (CALLED THE STRING CONTROL BYTE - SCB) WHICH PRECEEDS THE STRING.

THE SCB OF VALUE 0 INDICATES THE END OF RECORD.

CONSIDER THE FOLLOWING RECORD, WHICH IS TO BE TRANSMITTED

'TRANSMITTED '

WHAT FOLLOWS IS A BREAKDOWN OF THE ABOVE RECORD INTO THE THREE CATEGORIES DESCRIBED.

```
'TRANSMIT'  NON-DUPLICATE CHARACTER STRING OF 7 CHARS.
'TT'        DUPLICATE STRING (NON-BLANK) OF 2 CHARS.
'ED'        NON-DUPLICATE CHARACTER STRING OF 2 CHARS.
'  '        DUPLICATE STRING (BLANK) OF 2 CHARS.
```

THE STRING IS BROKEN DOWN INTO 4 DISCREET SECTIONS, EACH OF WHICH MUST, WHEN TRANSMITTED, BE PRECEEDED BY AN SCB, THE LAST STRING MUST ALSO BE FOLLOWED BY A ZERO SCB TO INDICATE END OF RECORD.

THE MESSAGE IS, THEREFORE TRANSMITTED AS:-

SCB1	'TRANSMIT'	SCB2	'T'	SCB3	'ED'	SCB4	SCB5

```
WHERE: SCB1 = 'A NON-DUPLICATE STRING OF 7 CHARS FOLLOWS'
       SCB2 ='THE FOLLOWING CHARACTER IS TO BE REPEATED TWICE'
       SCB3 = 'A NON-DUPLICATE STRING OF 2 CHARS FOLLOWS'
       SCB4 = 'THERE ARE NOW 2 BLANKS TO BE INSERTED'
       SCB5 = 'END OF RECORD'.
```

TO ALLOW MULTIPLE RECORDS OF VARIOUS TYPES (E.G. BINARY, EBCDIC, PRINT RECORD, PUNCH RECORDS ETC.) TO BE GROUPED TOGETHER IN A SINGLE TRANSMISSION BLOCK, EACH RECORD IS PRECEEDED BY TWO EIGHT-BIT FIELDS.

1. RECORD CONTROL BYTE (RCB)
2. SUB-RECORD CONTROL BYTE (SRCB)

THE RECORD CONTROL BYTE IDENTIFIES THE TYPE AND FUNCTION OF THE RECORD FOLLOWING (E.G. PRINT OR PUNCH).

THE SUB-RECORD CONTROL BYTE IS USED TO SUPPLY ADDITIONAL INFORMATION CONCERNING THE RECORD TO THE RECEIVING PROGRAM. (E.G. IN THE TRANSMISSION OF DATA TO BE PRINTED, THE SRCB CAN BE INITIALISED FOR CARRIAGE CONTROL INFORMATION).

THE OTHER CONTROL FIELDS (THE BLOCK CONTROL BYTE - BCB, AND THE FUNCTIONAL CONTROL SEQUENCE - FCS) EXIST, BUT THEIR INFORMATION IS OF RELEVANCE TO THE SYMBIONT, ALTHOUGH ALLOWANCE IS MADE FOR THEM WHEN PRODUCING/EXAMINING FILES.

THE THREE FIELDS SCB, RCB AND SRCB WILL NOW BE DISCUSSED AS MENTIONED EARLIER.

MULTI LEAVING AS PRODUCED BY TRANSMIT

STRING CONTROL BYTE (SCB) - TRANSMIT

THE TRANSMIT SUBROUTINE PRODUCED THE FULL COMPLIMENT OF SCBS WITH ONE EXCEPTION (SEE NOTE BELOW).

THE SCB IS AN 8-BIT BYTE DEFINED AS

```
  - - - - - - - -
 | O K L J J J J J |   (LETTER REPRESENTATIONS ARE IBMS)
  - - - - - - - -

BIT MEANING:-  O=0 END OF RECORD (KLJJJJJ=0)
               O=1 ALL OTHER SCBS

                 K=0 DUPLICATE CHARACTER STRING
                   L=0 DUPLICATE CHARACTER IS BLANK
                   L=1 DUPLICATE CHARACTER IS NON-BLANK
                   JJJJJ DUPLICATION COUNT

                 K=1 NON-DUPLICATE CHARACTER STRING
                   LJJJJJ CHARACTER STRING LENGTH
```

NOTE: (1) A SPECIAL CASE IS MADE OF O = 1 AND KLJJJJJ = 0. (SEE UNSUPPORTED FUNCTIONS LATER IN THIS APPENDIX).
(2) SEE SRCB IN 'UNSUPPORTED FUNCTIONS' FOR DETAILS OF COUNT UNITS.

RECORD CONTROL BYTE RCB - TRANSMIT

THE TRANSMIT SUBROUTINE IS ONLY EXPECTED TO SUPPORT A SUBSET OF THE RCBS, HOWEVER, THERE IS ONE FUNCTION (MULTIPLE STREAMS) WHICH IS UNSUPPORTED. (SEE UNSUPPORTED FUNCTIONS LATER IN THIS APPENDIX).

THE RCB IS AN 8-BIT BYTE DEFINED AS:

```
  - - - - - - - -
 | O I I I T T T T |
  - - - - - - - -

BIT MEANING:- O = 0 END OF TRANSMISSION BLOCK (IIITTTT = 0)
              O = 1 ALL OTHER RCBS

                TTTT = 0010 SEND MESSAGE TO IBM OPERATOR
                     = 0011 SEND RECORD TO IBM. (THIS MAY BE BINARY OR
                            EBCDIC - ITS CONTEXT DETERMINES WHICH).
                TTTT = ANY OTHER VALUE - SEE 'MULTI LEAVING ACCEPTABLE TO
                       RECEIVE' OR 'UNSUPPORTED FUNCTIONS' LATER IN THIS
                       APPENDIX.

                III  = STREAM IDENTIFIER (THIS ENABLES, SAY, MANY CARD
                       READERS TO EXIST AT A REMOTE LOCATION EACH RUNNING
                       UNDER A DIFFERENT STREAM). THE FEATURE IS NOT
                       SUPPORT - SEE 'UNSUPPORTED FUNCTIONS' LATER IN
                       THIS APPENDIX.
```

SUB RECORD CONTROL BYTE SRCB - TRANSMIT

THE TRANSMIT ROUTINE IS ONLY EXPECTED TO SUPPORT THE SRCBS FOR INPUT RECORDS AND OPERATORS COMMAND. AS DOCUMENTATION OF SCRBS FOR OPERATOR COMMANDS IS VIRTUALLY NON-EXISTANT, THE DEFAULT IS ALWAYS USED (:200).

FOR INPUT RECORDS THE SRCB IS DEFINED AS FOLLOWS:

```
  - - - - - - - -
 | O M M B R R R R |
  - - - - - - - -
```

BIT MEANING:- O=1 (MUST ALWAYS BE ON).
MM = 00 SCB COUNT UNITS = 1
MM <> 00 UNSUPPORTED FUNCTION - SEE LATER IN APPENDIX.
BRRRR = RESERVED (TRANSMIT SETS THESE TO ZERO).

MULTI LEAVING ACCEPTABLE TO RECEIVE

STRING CONTROL BYTE (SCB) - RECEIVE

THE RECEIVE ROUTINE ACCEPTS THE FULL COMPLIMENT OF SCBS WITH THE SAME EXCEPTION AS FOR TRANSMIT ROUTINE.

SEE SCB IN 'MULTI LEAVING PRODUCED BY TRANSMIT' EARLIER.

RECORD CONTROL BYTE (RCB) - RECEIVE

THE RECEIVE SUBROUTINE IS ONLY EXPECTED TO SUPPORT A SUBSET OF THE RCBS, HOWEVER, THERE IS ONE FUNCTION (MULTIPLE STREAMS) WHICH IS UNSUPPORTED. (SEE UNSUPPORTED FUNCTIONS LATER IN THIS APPENDIX).

THE RCB IS AN 8-BIT BYTE DEFINED AS

```
  - - - - - - - -
| O I I I T T T T |
  - - - - - - - -
```

BIT MEANING:-O = 0 END OF TRANSMISSION BLOCK (IIITTTT = 0)
O = 1 ALL OTHER RCBS

TTTT = 0001 DISPLAY MESSAGE ON PRIME TERMINAL
= 0100 RECEIVE PRINT RECORD
= 0101 RECEIVE PUNCH RECORD.

TTTT = ANY OTHER VALUE - SEE 'MULTI LEAVING PRODUCED BY TRANSMIT' OR 'UNSUPPORTED FUNCTIONS' ELSEWHERE IN THIS APPENDIX.

III = STREAM IDENTIFIER (THIS ENABLES, SAY MANY PRINTERS, TO EXIST AT A REMOTE LOCATION, EACH RUNNING UNDER A DIFFERENT STREAM). THE FEATURE IS NOT SUPPORTED - SEE 'UNSUPPORTED FUNCTIONS' LATER IN THIS APPENDIX.

SUB RECORD CONTROL BYTE (SRCB) - RECEIVE

THE RECEIVE ROUTINE IS ONLY EXPECTED TO SUPPORT THE SRCBS FOR:

1. OPERATOR MESSAGE DISPLAY REQUEST
2. PRINT RECORD.
3. PUNCH RECORD.

ONCE AGAIN THE DOCUMENTATION FOR 'OPERATOR MESSAGE DISPLAY REQUESTS' IS NON EXISTANT AND SO THE RECEIVE ROUTINE IGNORES THESE SRCBS AND UPON A ZERO SCB (END OF RECORD) OUTPUTS THE MESSAGE TO THE TERMINAL, TERMINATED WITH A SINGLE NEWLINE (:212).

FOR PRINT RECORDS THE SRCB IS DEFINED AS FOLLOWS:

```
  - - - - - - - -
| O M C C C C C C |
  - - - - - - - -
```

BIT MEANINGS:- O = 1 (MUST BE ON)
NOTE: RECEIVE DOES NOT CHECK THIS BIT
IT ASSUMES O = 1
M = O NORMAL CARRIAGE CONTROL
M = 1 RESERVED FOR FUTURE USE.
NOTE: RECEIVE DOES NOT CHECK THIS BIT
IT ASSUMES M = 1
CCCCCC = CARRIAGE CONTROL INFORMATION
= 1000NN SPACE IMMEDIATELY NN LINES
= 11NNNN SKIP IMMEDIATELY TO CHANNEL NNNN
= 0000NN SPACE NN LINES OFFER PRINT
= 01NNNN SKIP TO CHANNEL NNNN AFTER PRINT
= 000000 SUPPRESS SPACE

NOTE: CCCCCC = 001100 (LOAD 3211 FCB IMAGE)
IS NOT SUPPORTED - SEE
'UNSUPPORTED FUNCTIONS' LATER IN
THIS APPENDIX.

FOR PUNCH RECORDS THE SRCB IS DEFINED AS FOLLOWS:

| O | M | M | B | R | R | S | S |

BIT MEANINGS:- O = 1 (MUST BE ON)
NOTE: RECEIVE DOES NOT CHECK THIS BIT.
IT ASSUMES O = 1

SS = PUNCH STACKER SELECT INFORMATION.
NOTE: DOCUMENTATION FOR THIS HAS NOT BEEN
OBTAINED AND SO RECEIVE DOES NOT CHECK
THIS.
B = O NORMAL EBCDIC CARD IMAGE.
NOTE: B = 1 IS UNSUPPORTED - SEE
'UNSUPPORTED FUNCTIONS' LATER IN THIS
APPENDIX.

MM = 00 SCB COUNT UNITS = 1
MM <> 00 UNSUPPORTED FUNCTION - SEE LATER IN THIS
APPENDIX.

RR = RESERVED FOR FUTURE EXPANSION

NOTE: RECEIVE ASSUMES THAT THE SRCB FOR PUNCH RECORDS IS :200, IT DOES NOT CHECK FOR ANY OTHER VALUE.

UNSUPPORTED FUNCTIONS (TRANSMIT/RECEIVE)

STRING CONTROL BYTE (SCB)

TWO POINTS TO NOTE HERE:

1. AN SCB OF :200 IS NEVER PRODUCED BY TRANSMIT NOR IS IT ACCEPTABLE TO RECEIVE. THE FORMER IS ONLY A COMMENT BUT IF RECEIVE WERE TO RECEIVE AN SCB OF VALUE :200 THEN THE OUTCOME IS UNDEFINED.

2. THE COUNT (EITHER DUPLICATION COUNT OR CHARACTER STRING LENGTH) MAY THEORETICALLY BE EXPRESSED IN UNITS OF 1, 2 OR 4. HOWEVER, AT THE TIME OF WRITING THE DOCUMENTATION INDICATED THAT JES2 DID NOT UTILISE ANY COUNT UNIT OTHER THAN 1. THEREFORE THAT IS THE ONLY COUNT UNIT ACCEPTABLE TO THE RECEIVE (IT WILL IGNORE ANY INSTRUCTIONS TO DO OTHERWISE) AND THE ONLY ONE PRODUCE BY TRANSMIT.

RECORD CONTROL BYTE (RCB)

THE FORMAT OF THE RCB IS AS FOLLOWS:

```
  - - - - - - - -
| O I I I T T T T |
  - - - - - - - -
```

THE FOLLOWING IS NOT SUPPORTED BY TRANSMIT/ RECEIVE

TTTT = 0000 - THE ONLY CONTROL RECORDS SUPPORTED BY JES2 ARE THOSE HANDLED BY DIFFERENT PARTS OF THE EMULATOR SOFTWARE.
= 0110 - DATA SET RECORD NOT SUPPORTED BY JES2.
= 0111 - TERMINAL MESSAGE ROUTING REQUEST NOT SUPPORTED BY JES2. = 1000-1100 - RESERVED FOR FUTURE USER
= 1101-1111 - AVAILABLE FOR LOCAL MOD.

SUB RECORD CONTROL BYTE (SRCB)

THE FOLLOWING SRCBS ARE NOT SUPPORTED.

FOR PRINT RECORDS ON SRCB = :214 (LOAD 3211 FCB IMAGE) IS NOT SUPPORTED. THIS FUNCTION IS SUBSTITUTED BY THE USE OF THE VFC COMMANDS AVAILABLE FROM A TERMINAL.

FOR PUNCH RECORDS ANY SRCB <> :200 IS NOT SUPPORTED. JES2 DOES NOT SUPPORT COLUMN BINARY NOR SCB COUNT UNITS <>1.

FOR INPUT RECORDS THE TRANSMIT SUBROUTINE ONLY PRODUCED AN SCB COUNT UNIT = 1 (JES2 DOES NOT SUPPORT THE OTHERS).

THE CHARACTER IS RETURNED IN VARIABLE CHAR WHICH MAY THEN BE ASSIGNED TO ANY OTHER COMPATIBLE VARIABLE.

SIMILARLY, TO OUTPUT A CHARACTER, THE CALL

```
CALL TPUTC(CHAR)
```

IS REPLACED BY:

```
      CHAR = CHAR
      RTNADR = $XXXX
      GOTO TPUTC$
XXXX CONTINUE
```

THE 'RETURN' STATEMENT THAT WAS USED IN THE 'CLOSED' SUBROUTINE IS REPLACED BY 'GOTO RTNADR'.

SECTION 10

THE SYMBIONTS

THE SYMBIONTS USED IN THE NEW EMULATORS ARE IN GENERAL THE SAME SYMBIONTS AS USED BY THE CURRENT EMULATORS. THE WORKSTATION SYMBIONT INTERFACE IS NOW THE SAME FOR ALL EMULATORS.

THE 1004 SYMBIONT HAS BEEN MODIFIED TO REMOVE THE OPERATOR INTERFACE AND INCLUDE THE WORKSTATION SYMBIONT STANDARD INTERFACE. THE 1004 SYMBIONT MAY NOW RUN AS A PHANTOM USER.

THE 7020 AND GRTS EMULATORS HAVE NOT BEEN RELEASED WITH MASTER DISC SOFTWARE PREVIOUSLY. THE 7020 AND GRTS SYMBIONTS ARE OF THE SAME STRUCTURE AS THE OTHER SYMBIONTS.

ALL THE SYMBIONTS HAVE THE FOLLOWING FEATURES:

1) WHEN THE SYMBIONT IS RESUMED IT OPENS A FILE CALLED 'SYMLOCK'. THIS IS USED AS A FLAG BY THE WORKSTATION TO CHECK IF THE SYMBIONT IS RUNNING OR NOT, OR WHETHER IT HAS CRASHED.

2) WHEN A LINE IS ENABLED, A FLAG IS SET INDICATING THE LINE IS IN USE.

3) WHENEVER A LINE IS DISABLED, THE ENABLE FLAG IS UNSET. IF, AT THIS TIME, ALL LINES ARE DISABLED, THE 'SYMLOCK' FILE IS CLOSED AND DELETED, AND THE SYMBIONT DIES PEACEFULLY BY CALLING EXIT.

THE COMMAND FILE USED TO START THE SYMBIONT IS CALLED PH_REM. LINES TO BE ASSIGNED TO THIS SYMBIONT MUST BE DECLARED IN PH_REM. A COMMAND OUTPUT FILE, CALLED COMO_XXXXX (WHERE XXXXX REPRESENTS AN EMULATOR E.G. 200UT, 1004 ETC) IS OPENED WHENEVER PH_REM IS IN USE. COMO_XXXXX SHOULD BE EXAMINED IF THE SYMBIONT CRASHES UNEXPECTEDLY.