60472200

CONTROL DATA
CORPORATION

STATE PROGRAMMING LANGUAGE REFERENCE MANUAL

CDC® COMPUTER SYSTEMS:
255X SERIES
NETWORK PROCESSOR UNITS
COMMUNICATIONS CONTROL PROGRAM (CCP)
COMMUNICATIONS CONTROL INTERCOM (CCI)
COMMUNICATIONS CONTROL MODULE (CCM)

CDC® HOST OPERATING SYSTEMS:
NOS 1
NOS/BE 1
MASTER/MCS III

REVISION RECORD

REVISION	DESCRIPTION
A	Original release.
(6-30-78)	
B	Revised to CCP 3.2, PSR Level 497. This revision obsoletes all previous editions.
(5-31-79)	
C	Revised to CCP/CCI, PSR Level 518 (HASP postprint).
(5-22-80)	

Publication No.
60472200

REVISION LETTERS I, O, Q AND X ARE NOT USED

Address comments concerning this manual to:

CONTROL DATA CORPORATION
Publications and Graphics Division
P. O. Box 4380-P
Anaheim, California 92803

or use Comment Sheet in the back of this manual.

LIST OF EFFECTIVE PAGES

New features, as well as changes, deletions, and additions to information in this manual, are indicated by bars in the margins or by a dot near the page number if the entire page is affected. A bar by the page number indicates pagination rather than content has changed.

PAGE	REV	PAGE	REV	PAGE	REV	PAGE	REV
Cover	-						
Title Page	-						
ii thru iv	C						
v/vi	B						
vii	C						
1-1/1-2	B						
2-1 thru 2-3	B						
3-1 thru 3-3	B						
4-1/4-2	B						
5-1 thru 5-11	B						
5-12 thru 5-15	C						
A-1 thru A-8	B						
A-9 thru A-11	C						
B-1	B						
B-2	C						
C-1	B						
D-1 thru D-8	B						
Index-1/Index-2	C						
Comment Sheet	-						
Mailer	-						
Back Cover	-						

PREFACE

The manual is intended to provide specific programming information for analyst-level personnel who wish to create or to modify the firmware-level (mux-level) message processing portions of a terminal interface program (TIP). These programs are called text processing state programs for downline messages and input state programs for upline messages. The programs are required for every TIP in a 255x Network Processor Unit using Communications Control Program (CCP), Communications Control INTERCOM (CCI) or Communications Control Module (CCM). There is also a set of modem state programs used in each of these systems.

This manual should be used in conjunction with the appropriate System Programmer's Reference Manual for CCP or CCI. Unless specified, all references to number are to decimal values; all references to bytes are to 8-bit bytes; all references to characters are to 8-bit ASCII-coded characters.

RELATED MANUALS

Additional information on state programs and on systems which use state programs can be found in the following documents:

Publication Title	Publication Number
Communications Control Program Version 3 System Programmer's Reference Manual	60474500
Communications Control INTERCOM Version 3 System Programmer's Reference Manual	60471160
Communications Control Module Version 3 Reference Manual	60470500
Macro Assembler Reference Manual Mass Storage Operating System	60361900

CDC manuals can be ordered from Control Data Literature and Distribution Services, 308 North Dale Street, St. Paul, Minnesota 55103

This product is intended for use only as described in this document. Control Data cannot be responsible for the proper functioning of undescribed features or parameters.

CONTENTS

APPENDIXES

INDEX

FIGURES

INTRODUCTION 1

State programs handle protocol dependent tasks (such as code and format conversion) for a terminal interface program (TIP). These state programs operate on the firmware (multiplex) level. All state programs are written using a set of macros called state instructions. These macros are a defined set of CYBER 18 macro assembly macros and are assembled using the CYBER 18 macro assembler.

Three types of state program are needed by every TIP:

- Text processing state programs convert the code/format of output messages; and in some cases the code/format of input messages. These state programs are called directly from the TIP and return control to the TIP when the message text is in terminal format and ready for output. (In the case of input text processing, the message is in host format and is ready to be passed to the host.)

- Input state programs convert code/format for input messages. These state programs are specified by the TIP to the multiplex subsystem, which controls the programs directly. One-pass input state programs convert the message to a form expected by the host. Two-pass input state programs demultiplex data from the circular input buffer to an input source buffer. The TIP then performs input text processing.

- Modem state programs are common to all TIPs. They are controlled by the multiplex subsystem and are used to set up modem/communications line adapter parameters, and to take status from the communications line adapter parameters, and branch on the basis of the communications line adapter status. Modem state programs need be considered only if a new line type is added to the system.

PROGRAM INTERFACE

All TIPs are written on two levels of processing: the OPS level and the firmware level. State programs run at the firmware level and interface with the OPS-level TIP by passing information to them through worklist entries and/or through the control block (MLCB and TPCB are described later).

Part of the message processing is handled by the firmware output data processor (ODP) or by the input data processor (IDP). Both programs are part of the multiplex subsystem. The ODP is interrupt driven by a microprogram that is activated when output data demands (ODD) are generated by the communications line adapters. The ODP's primary function is to obtain characters from line-oriented output buffers, transform this data into line frame formats, and transfer the line frames onto the multiplex output loop.

Output text processing is required when the output sent by the host and received by the OPS-level TIP requires special handling (e.g., character translation) before being output to the terminal. Text processing state programs analyze and reformat the output buffer data to terminal format and code. This processing must be completed before the TIP requests the multiplex subsystem to start output on the line.

The IDP is a multiplex subsystem level 1 microprogram which removes loop cell data from the circular input buffer (CIB), strips off the multiplex loop control fields, and packs the resulting characters into line-oriented input buffers. Prior to storing an input character into the buffer, an input state program determines whether any special action is required for that character. When all the input characters in the transmission are processed and the line-oriented input buffer is completed, a worklist entry is sent to the TIP at OPS-level. The IDP is interrupt driven by the multiplex loop interface adapter whenever a line frame is stored in the CIB. Unless its processing is preempted by an ODP interrupt, the IDP processes all active entries in the CIB prior to relinquishing control.

STATE PROGRAM STRUCTURE

The elements of a state program are as follows:

- State program instructions provide individual firmware operations. These basic elements of the language are defined in section 5 and summarized in appendix A.

- State processes consist of one or more state instructions.

- State programs consist of one or more state processes. A state program assembles as a sequential table of coded state instructions, but processing starts or stops only at state process boundaries. All state programs are reentrant.

- State pointer tables contain a pointer to every state process in the program. The state pointer table is constructed with a set of macros to create both the state process addresses and the state indexes. The macro has the advantage of forcing the programmer to use mnemonic names for the state and indexes, thus making the code more flexible should state processes be deleted or inserted.

In the example (figure 1-1) of the creation of a state pointer table, the state named P1 is state 1, as determined by its position in the table. Defining the macro UMPTR1 using the CYBER 18 macro assembler creates a symbol, USP1, which is equated to 1 and an address reference named UP1. Elsewhere in the program there must be a label UP1 which defines the address of a set of state instructions defining this state process. The choice of the prefix US and U is arbitrary; however, the following conventions are in use:

A and AS -	Async or TTY TIP
H and HS -	HASP TIP
M and MS -	Modem State Programs
V and VS -	Mode 4 TIP

```
UMPTR1  MAC     NM
        EQU     US ≠ NM ≠ (*-UISPTBL) creates state index
                mnemonic
        ADC     U ≠ NM ≠
        FMC
  *
        ENT     UISPTBL
  *
UISPTBL UMPTR1  ESRC     end of source
        UMPTR1  P1       first state process (index = 1)
        UMPTR1  P2
          .       .
          .       .
          .       .
        UMPTR1  PN       last state process (index = n)
```

(Note that each state pointer table has a unique entry address name, UISPTBL in this case, and thus each table has its own macro.)

Figure 1-1. State Pointer Table Creation

MANUAL FORMAT

The remainder of the manual describes input state programs, modem state programs and the state instructions.

For further CYBER 18 macro assembler information, see the macros description in the Macro Assembler Reference Manual.

INPUT STATE PROGRAMS

Prior to the start of an input operation, the appropriate TIP passes information to the multiplex subsystem so that the subsystem knows which input state pointer table to use for a given line. As the data passes into the circular input buffer (CIB), the specified input state program is called by the input data processor (IDP), to store characters into line-oriented buffers. These buffers are sent to the TIP for further processing.

FIRMWARE INTERFACE

When the IDP detects a data character in the CIB, it passes control to the designated input state process for the line/terminal. Prior to executing the first state input state instruction, the firmware loads a selected register with the current (untranslated) character. The contents of this register may be tested or changed by state instructions. This register is referred to as the current character.

The parity bit is stripped when the register is initially loaded, if parity stripping is specified. If a state instruction changes the character of this register, parity stripping is ignored.

PROGRAM CONTROL

The line determines the port table (NAPORT) to use. The dynamically allocated multiplex line control block (MLCB) is found through NAPORT. Within the MLCB, selection of the input state process to execute is found by combining the value of the input state process index with the input state pointer table entry which points to the associated input state process. Figure 2-1 shows these relationships.

DATA STRUCTURE FOR INPUT STATE PROGRAM: MLCB

The TIP causes the command driver of the multiplex subsystem to set up the fields in the multiplex line control block (MLCB). MLCB fields hold various control information for the data processing. A standard 16-word MLCB is provided for all systems using state programs. This MLCB variant is shown in figure 2-2. Other variants of the MLCB are used by some systems. See the appropriate system programmer's reference manual for definition of variant MLCB fields.

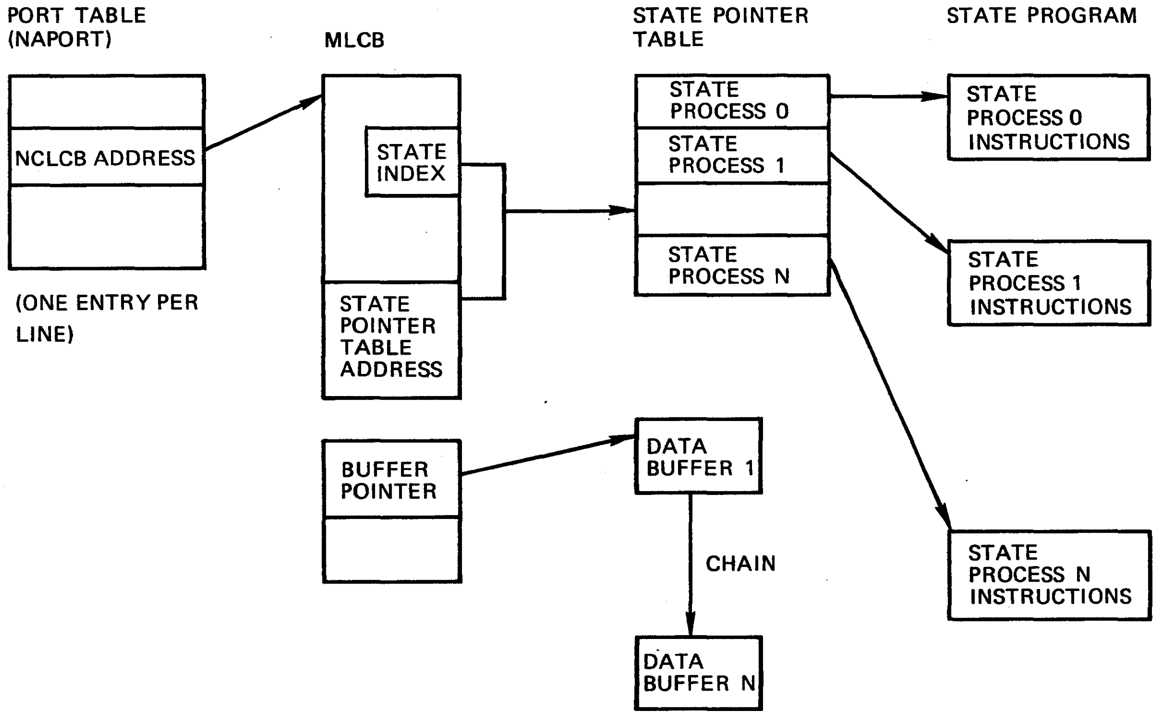


Figure 2-1. Locating an Input State Process

The TIP must never directly reference the MLCB. The fields within the MLCB may be changed only by the command driver or state instructions.

<table>
<tr><th></th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>0</th></tr>
<tr><td>0</td><td>F1</td><td>F2</td><td>F3</td><td>F4</td><td>F5</td><td>F6</td><td>F7</td><td>F8</td><td colspan="5">NCOCHR – NEXT OUTPUT CHARACTER</td></tr>
<tr><td>1</td><td>F9</td><td>F10</td><td>F11</td><td colspan="5">NCTIME – MULTIPLEX TIMER</td><td colspan="5">NCOBLCD – LCD OF OUTPUT BUFFER</td></tr>
<tr><td>2</td><td colspan="13">NCOBP – POINTER TO OUTPUT BUFFER</td></tr>
<tr><td>3</td><td>F12</td><td>F13</td><td>F14</td><td>F15</td><td>F16</td><td>F17</td><td>F18</td><td>F19</td><td>F20</td><td>F21</td><td colspan="3">NCISTAI – INPUT STATE PROGRAM INDEX</td></tr>
<tr><td>4</td><td colspan="8">NCCNTL – CHARACTER COUNT LIMIT</td><td colspan="5">NCCNT1 – CHARACTER COUNTER 1</td></tr>
<tr><td>5</td><td colspan="13">NCISPTA – POINTER TO INPUT STATE PROGRAM POINTERS TABLE</td></tr>
<tr><td>6</td><td colspan="13">NCIBP – POINTER TO INPUT BUFFER</td></tr>
<tr><td>7</td><td>F22</td><td>F23</td><td>F24</td><td>F25</td><td>F26</td><td>F27</td><td>F28</td><td>F29</td><td>F30</td><td>F31</td><td>F32</td><td colspan="2">NCCRCP – CRC POLYNOMIAL</td></tr>
<tr><td>8</td><td colspan="8">NCSCHR – SPECIAL CHARACTER</td><td colspan="5">NCIBFCD – FCD OF INPUT BUFFER</td></tr>
<tr><td>9</td><td colspan="13">NCCRCS – CRC ACCUMULATION</td></tr>
<tr><td>10</td><td colspan="3">NCZER1 – ZERO</td><td colspan="10">NCCNT2 – CHARACTER COUNTER 2</td></tr>
<tr><td>11</td><td colspan="3">NCZER2 – ZERO</td><td colspan="10">NCBLKL – BLOCK LENGTH (RECORDS)</td></tr>
<tr><td>12</td><td colspan="13">NCCXLTA – POINTER TO CODE TRANSLATE TABLE</td></tr>
<tr><td>13</td><td colspan="13">NCSCBA – POINTER TO FIRST BUFFER IN BLOCK</td></tr>
<tr><td>14</td><td colspan="8">NCBLCNT – NUMBER OF BUFFERS ALLOCATED</td><td colspan="5">NCSVWL – SAVED WORKLIST</td></tr>
<tr><td>15</td><td colspan="13"><b>RESERVED</b></td></tr>
</table>

Flags:

F1 = NCEOBL – end of block
F2 = NCNXOCA – next output character available
F3 = NCLCT – last character transmitted (CDCCP)
F4 = NCBCREQ – buffer chaining required
F5 = NCOMPRO – output message in progress
F6 = NCSP1 – not used
F7 = NCODDIN – ODD received
F8 = NCSP2 – not used
F9 = NCSUPCHAIN – suppress buffer chaining
F10 = NCOBT – generate output buffer terminated (OBT)
F11 = NCBZL – reset timer
F12 = NCRINCH – input character in right byte
F13 = NCCAREC – character received
F14 = NCRIGHTC – left/right source flag (1 = right)
F15 = NCINPRO – input message in progress
F16 = NCNOXL – code translation active

F17 = NCRPRT – strips parity bit
F18 = NCSCF – suppress chain flag
F19 = NCLASTCH – LCD of source buffer reached
F20 = NCEOSR – end of source buffer reached
F21 = NCSP3 – not used
F22 = NCUOP1
F23 = NCUOP2
F24 = NCUOP3
F25 = NCUOP4 } optional user flags
F26 = NCUOP5
F27 = NCUOP6
F28 = NCUOP7
F29 = NCUOP8
F30 = NCETX – Delay ETX worklist generation
F31 = NCMRTO – Modem response timed out
F32 = NCCARR – Line carrier type (1 = controlled; 0 = constant)

Figure 2-2. Standard MLCB

PROGRAM ORGANIZATION

An input state program consists of a maximum of 64 state processes. These states handle tasks such as data conversion, cyclic redundancy checksum generation, character compression, and message blocking. Since all state processes are reentrant, lines with a similar protocol (that is, controlled by a single TIP) share state processes.

The user must provide programs for the four reserved input state processes (0, 1, 2, and 3):

- State 0 handles parity errors and data transfer overruns.
- State 1 is called when DCD dropped is detected. This allows DCD dropped to be used as a logical ETX for controlled carrier lines.
- State 2 is called when the number of input buffers currently in use exceeds the system limit.
- State 3 is called when the buffer threshold is reached.

State 0 and state 1 are given control by the modem state program (regardless of the current input state) when the stated condition occurs. States 2 and 3 are called by the IDP to process buffer related condition when trying to store a new character which requires assigning a new buffer (note: the character is not stored). States 4 through 63 are defined by the TIP.

INTERFACE TO THE MODEM STATE PROGRAMS

This subsection describes the current interface; it by no means represents all the allowable interfaces to the modem state programs. When a data character and communications line adapter status occur in the same line frame of the CIB, the firmware transfers control to the current modem state process. A modem state program jumps to input state process 0 or 1 upon detecting status conditions for which the input state program should get control.

MLCB flags are used for communication between a modem state program and an input state program. Setting NCETX indicates the input state program has detected the end of the input transmission and wishes to wait for the carrier before continuing. Setting NCETX has meaning only if NCCARR is also set. NCCARR is set by the line initializer for a controlled carrier line and must not be altered. State instructions are available to set, clear, and test these flags.

Input state programs set the modem state index to the modem state process which handles status while input is in progress. That is, upon detecting start of input, the input state program changes the modem state index to point to the modem state process which handles status when inputting (MSTINP). Then, upon detecting end of transmission, the input state program sets the modem state index to the modem state process for idle (MSTIDL).

On controlled carrier-type lines, an output message cannot be transmitted until data carrier detect (DCD) drops on input. To eliminate the possibility of TIPs attempting to output before DCD drops during input, the input state program has the ability to terminate the input buffer and save the workcode in the MLCB (as opposed to building a worklist at termination time). The input state program then sets the NCETX user flag indicating that the workcode was saved. A worklist entry may be built immediately if the line type is not a controlled carrier line.

The modem state program jumps to input state process 1 when DCD drops while in the idle modem state. The input state can then send a worklist entry to the OPS level of the TIP. The TIP does not get control until DCD drops, eliminating the possibility of starting to output before DCD drops during input.

TEXT PROCESSING STATE PROGRAMS

Two kinds of text processing are provided by a system:

- Output text processing converts data from host format to data in terminal code/format. The processed data is placed in an output buffer (or chain of buffers) and the multiplex subsystem then sends the data to the terminal.

- Input text processing converts data from the source buffers to host code/format. The data was placed in the source buffers by the appropriate input state program.

Both types of text processing programs are called directly from the OPS-level TIP.

When handling characters for text processing state programs, the buffer containing data to be converted is called the source buffer. A character from this buffer is called the source character. The source character is placed in the current character register by the firmware.

DATA STRUCTURE, TPCB

The text processing control block (TPCB) contains information necessary to perform text processing. The first 19 words are standard in all systems but only the first 7 words plus a few named fields in other words are used by each TIP. Figure 3-1 shows the standard TPCB.

<table>
<tr><th></th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>0</td><td colspan="16">NCLCDFCD – SOURCE BUFFER LCD/FCD</td></tr>
<tr><td>1</td><td>F9</td><td>F10</td><td>F11</td><td colspan="5">NCTIME – MULTIPLEX TIMER</td><td colspan="8">NCOBLCD – LCD OF OUTPUT BUFFER</td></tr>
<tr><td>2</td><td colspan="16">NCSBP – SOURCE BUFFER POINTERS</td></tr>
<tr><td>3</td><td>F12</td><td>F13</td><td>F14</td><td>F15</td><td>F16</td><td>F17</td><td>F18</td><td>F19</td><td>F20</td><td>F21</td><td colspan="6">NCISTA1 – INPUT STATE PROGRAM INDEX</td></tr>
<tr><td>4</td><td colspan="8">NCCNTL – CHARACTER COUNT LIMIT</td><td colspan="8">NCCNT1 – CHARACTER COUNTER 1</td></tr>
<tr><td>5</td><td colspan="16">NCSPTA – POINTER TO STATE PROGRAMS POINTERS TABLE</td></tr>
<tr><td>6</td><td colspan="16">NCDBP – POINTER TO STATE PROGRAMS TABLE</td></tr>
<tr><td>7</td><td>F22</td><td>F23</td><td>F24</td><td>F25</td><td>F26</td><td>F27</td><td>F28</td><td>F29</td><td>F30</td><td>F31</td><td>F32</td><td colspan="5">NCCRCP – CRC POLYNOMIAL</td></tr>
<tr><td>8</td><td colspan="8">NCSCHR – SPECIAL CHARACTER</td><td colspan="8">NC1BFCD – FCD OF INPUT BUFFER</td></tr>
<tr><td>9</td><td colspan="16">NCCRCS – CRC ACCUMULATION</td></tr>
<tr><td>10</td><td colspan="3">NCZER1 – ZERO</td><td colspan="13">NCCNT1 – CHARACTER COUNTER 2</td></tr>
<tr><td>11</td><td colspan="3">NCZER2 – ZERO</td><td colspan="13">NCBLK1 – BLOCK LENGTH (RECORDS)</td></tr>
<tr><td>12</td><td colspan="16">NCCXLTA – POINTER TO CODE TRANSLATE TABLE</td></tr>
<tr><td>13</td><td colspan="16">NCFDBA – POINTER TO FIRST DESTINATION BUFFER</td></tr>
<tr><td>14</td><td colspan="8">NCBLCNT – NUMBER OF BUFFERS ALLOCATED</td><td colspan="8">NCSVWL – SAVED WORKLIST</td></tr>
<tr><td>15</td><td colspan="16">RESERVED</td></tr>
<tr><td>16</td><td colspan="16">NCDUMD</td></tr>
<tr><td>17</td><td colspan="16">NCDUME</td></tr>
<tr><td>18</td><td colspan="16">NCFSBA – FIRST STORAGE BUFFER ADDRESS</td></tr>
<tr><td>19</td><td colspan="16">RESERVED FOR TIP USAGE</td></tr>
<tr><td></td><td colspan="16"></td></tr>
<tr><td>31</td><td colspan="16">RESERVED FOR TIP USAGE</td></tr>
</table>

M-422

Figure 3-1. Standard TPCB

FIRMWARE INTERFACE

The procedure PTTPINF provides the PASCAL interface to the text processor. The procedure is called with one parameter specified with the control block to be used. The control block is a variable of type NCLCB.

The format of the call is PTTPINF (TPCB) where the TPCB is contained in a data buffer. A pointer variable of type B0BUFPTR is required to contain the address of the TPCB. Control is returned to the called with various control fields set in the TPCB.

TPCB INITIAL SET-UP

Prior to calling the firmware to perform text processing, the TIP prepares the TPCB. Three fields must be initialized:

- NCSPTA and NCSTAI point to the first text process to execute.
- NCFSBA specifies the first source buffer to be text processed.

Depending on the TIP and the type of data to be processed, several other fields need to be initialized:

- NCBLKL, NCCNT1, NCCNT2, and NCCNTL specify the counters (word count values and initialization values).
- NCSCHR contains the special character used by the SPCHEQ state instruction.
- NCCRCP selects the cyclic redundancy check (CRC) polynomial.
- NCSCF suppresses length chaining of the input source; and is used if a nonstandard buffer is used as the source.
- NCUOPS user option flags are set as appropriate. All other fields must be zero.
- TIP defined fields in words 19 to 31 may be set as needed.

TPCB SET-UP FOR RESTART

NCSBP and NCDBP fields can affect a restart condition (or the initial call) and are set to zero prior to calling the text processing state program.

- NCSBP - If this field is zero, the firmware obtains the first character from NCFSBA and sets all related flags to their proper state.

  If this field is nonzero, the firmware assumes a continuation. The next source character is obtained based on this word, NCRIGHTC, and NCEOSR. To determine the end of the source condition, the firmware expects the data to be in the data buffer and the LCD to be in the NCLCDFCD field.

- NCDBP - If this field is zero, the firmware gets a buffer, sets NCFDBA with the address of the buffer, and sets all flags to their proper state.

  If this field is nonzero, the firmware stores the next character based on this pointer and NCRINCH.

The TIP must also reset any of the initial parameters required by the restarted state program. If CRC is being accumulated, the field NCCRCS must be restored. The restart is typically used when the initial source is exhausted and the TIP must wait for more data to complete the destination block. If the TPCB is contained in a data buffer, no field need be changed except NCFSBA and NCSBP.

TPCB RETURN VALUES

On return to the calling program the TPCB will contain parameters as needed for the TIP to determine the actions performed by the state programs. The following fields are available:

- NCFSBA -Contains the address of the first destination buffers containing the processed data.
- NCVQPS -Contains the user-option flags being returned.
- The TIP defined fields in words 19 to 31 may contain any values, as needed.

If source data is to be fragmented into more than one destination block, some special processing is usually necessary. On return from test processing, the source buffers that have been completely processed should be released. The first source buffer containing data not yet processed should have its first character displacement (FCD) updated to point to the next character to be processed. The following fields may be used:

- NCSBP - Contains the address of the word containing the next source character to process.
- NCEOSR -is set to TRUE if the next source character is the first of the next buffer.
- NCRIGHTC - is set to TRUE if the next source character is in bits 7 to 0 of the word.

FILE 1 TEXT PROCESSING REGISTERS

A group of 16 firmware registers referred to as the file 1 text processing registers are initialized from the last 16 words of the TPCB before text processing is initiated.

The 16 file 1 registers are accessed by specifying a displacement to the selected file 1 register. Thus, a displacement of 0 selects the first text processing file 1 register and a displacement of 15 selects the last text processing file 1 register.

PROGRAM CONTROL

The text processing state process to be executed is determined by combining the value of the state process index with the state pointer table address. Both fields are in the TPCB. The selected text processing state pointer table entry points to the associated text processing state process. The process is the same as that shown in figure 2-1 except there is no port table and the TPCB takes the place of the MLCB.

The state pointer table address and state process index fields are set by the OPS-level TIP program. State processing instructions may change the processing index while executing state programs.

PROGRAM ORGANIZATION

A text processing state program consists of a maximum of 64 state processes. Since all state processes are reentrant, lines with a similar protocol may share state processes.

Text processing state process 0 is reserved for handling the end-of-source-reached condition and state process 2 is reserved for handling buffer overflow processing. States 1, and 3 through 63 are defined by the TIP.

MODEM STATE PROGRAMS 4

The modem state programs process modem status as a function of modem control signals. The programs, which are called by the firmware when communications line adapter status enters the subsystem, forward the logical communications line adapter status via a worklist entry to the multiplex level status handler (PTCLAS). PTCLAS analyzes the status and reports line conditions to the TIP through a worklist entry.

FIRMWARE INTERFACE

Communications line adapter status is passed by the multiplex subsystem to the circular input buffer (CIB). The CIB provides temporary buffering of input characters (section 2) and communications line adapter status. When the firmware's input data processor (IDP) detects communications line adapter status, it passes control to modem state process for that line.

PROGRAM CONTROL

The modem state program is entered by accesing the port table. A combination of the modem state index and the modem state program address selects the modem state pointer table entry which points to the associated modem state process. Figure 4-1 shows this relationship.

The modem state program address field is set by the multiplex subsystem when a line is initialized. The modem state index is changed by the multiplex subsystem, by an input state program, or by the modem state program. The multiplex subsystem sets the modem state index to the modem state process to be executed according to the command being issued. The input state programs control the setting of the modem state program index for handling status while input processing is in progess.

PROGRAM ORGANIZATION

The modem state program consists of a maximum of 16 state processes. There are modem state processes defined for each line type based on line condition. Thus, the modem state program can have one or more processes for each condition or one state process to handle more than one line condition, depending on the line type.

INTERFACE TO THE MULTIPLEX LEVEL STATUS HANDLER

The modem state program builds a worklist entry containing the communications line adapter status. The multiplex level worklist processor routes the worklist entry to the multiplex level status handler, PTCLAS. Upon receiving control, PTCLAS analyzes the status condition indicator and acts accordingly. The appropriate action may be to generate a CE error message, start a timer for modem response or communications line adapter status overflow, or make a worklist entry to the associated TIP.

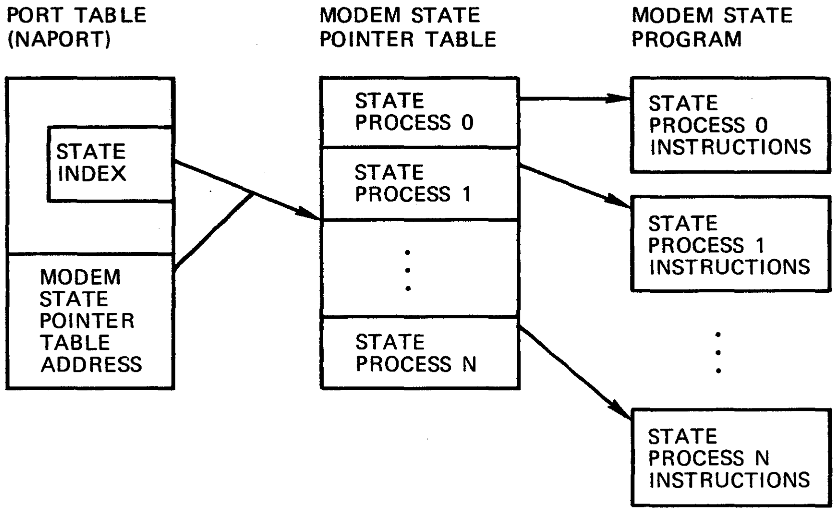

PORT TABLE (NAPORT)
STATE INDEX
MODEM STATE POINTER TABLE ADDRESS
MODEM STATE POINTER TABLE
STATE PROCESS 0
STATE PROCESS 1
STATE PROCESS N
MODEM STATE PROGRAM
STATE PROCESS 0 INSTRUCTIONS
STATE PROCESS 1 INSTRUCTIONS
STATE PROCESS N INSTRUCTIONS

Figure 4-1. Locating a Modem State Process

INTERFACE TO THE INPUT STATE PROGRAMS

When a data character and communications line adapter status occur in the same line frame of the CIB, the firmware transfers control to the current modem state process. The modem state program jumps to input state process 0 or 1 upon detecting status conditions for which the input state program gets control.

There are user flags in the multiplex line control block used for communication between the modem state program and input state program. Refer to the Input State Programs, Section 3.

Another user flag, MXCARR, is set by the line initializer when a controller carrier line is initialized.

The input states programs also set the modem state index to the modem state process which handles status while input is in progress. That is, upon detecting start of input, the input state program changes the modem state index to the modem state process which handles status when inputting (MSTINP). Then, upon detecting end of transmission, the input state program sets the modem state index to the modem state process for idle (MSTIDL).

On controlled carrier type lines, an output message cannot be transmitted until DCD drops following input. To eliminate the possibility of a TIP trying to output before DCD drops for the current input operation, the input state program has the ability to terminate the input buffer and to save the workcode in the multiplex line control block (as opposed to building the worklist at terminate time). The input state program sets the MXETX user flag indicating this saved workcode condition and sets the modem state index to idle (MSTIDL). A worklist entry is built immediately if the line type is not a controlled carrier line.

The modem state program jumps to input state process 1 when MXETX sets and DCD drops while in the idle modem state. The TIP does not get control until DCD drops, eliminating the possibility of starting output before DCD drops following input. When DCD drops, the TIP builds a worklist entry using the saved workcode and buffer address.

STATE INSTRUCTIONS

This section describes each state processing instruction in detail.

The general format for a state instruction is:

```
    MACRO NAME   PARAMETER1,
                 PARAMETER2,...,PARAMETERn
```

The number of parameters varies depending upon the state instruction. Note that this is the normal CYBER 18 macro assembler macro format. The macro name is followed by a blank. Parameters are separated by commas, and blanks within the parameter stream are ignored. Omitted parameters are delimited by commas; that is, PARAMETER1,,PARAMETER3 omits PARAMETER2.

Appendix A lists the state instructions by macro name in alphabetical order. Certain parameters are common to several state instructions. These parameters are listed separately in figure 5-1.

The instructions are functionally grouped in nine categories as follows:

- Handling assignable counters
- Character manipulation
- Index manipulation
- Skips
- Processing communications line adapter status
- Flag control
- Worklist handling
- Text processing
- Miscellaneous

HANDLING ASSIGNABLE COUNTER

Two general purpose counters, character counter 1 (CC1) and character counter 2 (CC2), are usd in state programs for tasks such as packetizing and character expanding. CC1 is an 8-bit counter whose value may range from 0-255; CC2 is a 12-bit counter whose value may range from 0-4095. Both counters are maintained in the control block (MLCB or TPCB).

INITIALIZE CHARACTER COUNTER

This state instruction initializes either of two character counters that are maintained in the control block. Character count 1 is initialized from the line control block field NCCNTL. Character count 2 is initialized from the line control block NCBLKL field.

Macro Call

```
    INTCC     COUNT,ACTION
```

Initializes the specified character counter.

Usage

The initialize character counter instruction resets control block NCCNT1 or NCCNT2 with the values set in the fields NCCNTL or NCBLKL, respectively. For input state programs, NCCNTL and NCBLKL are set by issuing an ENABLE or INPUT command to the command driver. For text processing programs, the values are set in the TPCB before calling the firmware.

SET CHARACTER COUNTER

This two-word state instruction sets either character count 1 or count 2 to a specified value.

Macro Call

```
    SETCC     COUNT,CV
```

Sets character count (COUNT) to value (CV).

MASK AND SET CHARACTER COUNTER

This two-word state instruction masks, using a logical AND, a specified value to the current (untranslated) character. The result is stored in the selected character counter.

Macro Call

```
    CHRCC     COUNT,IMASK
```

Sets designated character counter (COUNT).

Nonstandard Parameters

IMASK	8-bit mask

SET CHARACTER COUNTER WITH MOD FUNCTION

This two-word state instruction performs a modulus function by repeatedly subtracting a given modulo value until the result is negative. The modulo value is then added to the negative number and the result is stored in the specified character counter.

Macro Call

```
    MODCC     COUNT,CV
```

ACTION

Selects a character related and/or process control action.

Symbolic Name	Value	Description
Not specified	0	Default
–	0	Execute next instruction
EXIT	1	Discard character and exit
STOREXIT	2	Store character and exit
CRCSTOREX	3	Accumulate CRC, store character, and exit
CRCEXIT	4	Accumulate CRC, discard character, and exit
CRCNT	5	Accumulate CRC, execute next instruction

CHAR

Defines an 8-bit character.

COUNT

Symbolic Name	Value	Description
Not specified	0	Error
–	1	Count 1
–	2	Count 2

CRCA

Symbolic Name	Value	Description
Not specified	0	Default
		Store character and do not accumulate CRC
CRCA	1	Store character and accumulate CRC

CV

Count value (must not be zero).

DD

Sets the destination displacement to the file 1 register.

Symbolic Name	Value	Description
Not specified	0	File 1 register (first)
–	0-15	File 1 register (first through 16th)

EOT

Symbolic Name	Value	Description
Not specified	0	Default
–	0	Reset EOT flag
EOT	1	Set EOT flag

EP

This determines the worklist control block (WLCB) or translation table to be used. This label is associated with this instruction so that the address of the appropriate translation table or OPS-level WLCB may be supplied by the link editor at a later time. If the WLCB parameter is not specified or is 0, the multiplex WLCB is used.

LABEL

The name associated with the state instruction to receive control. The label must be on an instruction that is within N locations forward or back from this instruction. N is defined in each label using instruction.

SD

Sets the source displacement to the file 1 register.

Symbolic Name	Value	Description
Not specified	0	File 1 register (first)
–	0-15	File 1 register (first through 16th)

VALUE

The hexadecimal value to be used.

WC

Specifies the workcode.

Symbolic Name	Value (hexadecimal)	Description	
Not specified	0	Default	
–	0	Use saved workcode	Multiplex or OPS-level
–	1-7F	Use given workcode	

WL

This parameter is not used; however, space must be allocated for it in the parameter string.

Figure 5-1. Standard Macro Parameter Definitions

INCREMENT CHARACTER COUNTER

This state instruction increments (by one) either character count 1 or count 2 of the control block. Counter recycles if incremented when full.

Macro Call

```
ICC     COUNT,ACTION
```

Increment the specified character count (COUNT).

DECREMENT CHARACTER COUNTER

This state instruction decrements (by one) either character count 1 or count 2 of the control block. When the specified character count reaches zero the processor skips to the designated instruction. While the character count is not zero, the specified action exit is performed. If the count is zero when this instruction is executed, the count is set to minus one. This value is treated as a large positive number for subsequent operations.

Macro Call

```
DCC     COUNT,LABEL,ACTION
```

Decrement the specified character count (COUNT).

Usage

This is used to store or discard a fixed number (count) of characters. When the last character in the string is processed, the state program skips to the selected label to continue processing.

COMPARE CHARACTER COUNTER TO A VALUE

This two-word state instruction compares the selected character counter to a specified value.

character count = value: execute next instruction

character count ≠ value: skip

Macro Call

```
CNTNE   COUNT,CV,LABEL
```

Use specified character count (COUNT).

Labeled instruction is within ±8 instructions of macro.

COMPARE CHARACTER COUNTER TO BLOCK LENGTH

This two-word state instruction compares the block length with either character count 1 or count 2.

block length ≠ count: skip
block length = count: execute next instruction

Macro Call

```
BLCNE   COUNT,LABEL
```

Uses the specified character count (COUNT) for the comparison.

The label must be on an instruction that is within 8 locations forward from this instruction.

Usage

The block length for this comparison is obtained from the control block field, NCBLKL.

STORE CHARACTER COUNTER IN BUFFER

This state instruction stores either character count 1 or count 2 of the control block into the third word of the first destination buffer (following the flag word).

Macro Call

```
STORC   COUNT,ACTION
```

Store specified character count (COUNT) into the buffer.

Usage

The third word of the first destination buffer is used to communicate one counter value to the OPS-level TIP. Thus it is useful only during input state processing as the TIP is unable to access the control block.

CHARACTER MANIPULATION

These instructions store, replace, and add characters. The character is translated or altered during the operations.

STORE CHARACTER

This state instruction stores the current character into the destination buffer. If the translate flag is set, the current character is translated before it is stored.

Macro Call

```
STORE   CRCA
```

REPLACE CHARACTER

This state instruction takes the specified character and establishes it as the current (untranslated) character.

Macro Call

```
RCHAR   CHAR,ACTION
```

Usage

If the CRC is being accumulated and the existing current character is to be included in the CRC, it must be available to the encoder before executing this character instruction. This is accomplished by executing a previous instruction with an exit action parameter of CNCNT to accumulate the CRC.

When this instruction is executed during input processing, the current character received from the line is lost. For text processing, the current character is saved in the first file 1 register (displacement = 0) and may be restored, if desired. The saved copy of the character does not have the parity bit stripped regardless of the parity strip option. If the CRC accumulation is specified as an exit action with this instruction, the replacing character is CRC encoded.

NOTE

> RCHAR must exit to perform translation, CRC encoding, and character storing. ADDC does not allow CRC encoding or translating.

REPLACE AND STORE CHARACTER

This combination of two state instructions takes a specified character, establishes it as the current character, and stores it into the destination buffer.

Macro Call

```
RPLACE   CHAR,CRCA
```

Usage

The instruction produce the following code:

```
RCHAR    CHAR
STORE    CRCA
```

If the CRC is being accumulated and the existing current character is to be included in the CRC, it must be available to the encoder before executing this character instruction. This is accomplished by executing a previous instruction with an exit action parameter of CNCNT to accumulate in the CRC.

When this instruction is executed during input processing, the current character received from the line is lost. For text processing, the current character is saved in the first file 1 register (displacement = 0) and is restored, if desired. The saved copy of the character does not have the parity bit stripped even if the parity strip option is set. If the CRC accumulation is specified as an exit action with this instruction, the replacing character is CRC encoded.

This macro provides a shorthand method of coding to place a character into the destination buffer. The character is translated and CRC is adjusted. Control returns to the next state instruction.

ADD (INSERT) A CHARACTER

This state instruction inserts a given character into the destination buffer. Character CRC accumulation and translation is not performed.

Macro Call

```
ADDC    CHAR,ACTION
```

NOTE

> The exit action is performed on the current character and not the inserted character.

EXPAND (REPEAT) CHARACTER

This state instruction expands either a given character or the current character by placing it in the destination buffer. Character count 1 specifies the number of times the character is to be expanded.

Character translation is performed if the translation flag is set; however, CRC accumulation is not available.

NOTE

> When the initial value of character counter 1 is zero or is greater than 80, expansion is not performed. The next state instruction is executed.

Macro Calls

RADDC   CHAR

Expands the given character (CHAR).

CHRPT   Expands the current character.

INDEX MANIPULATIONS

Some macros manipulate the following state program indices:

Index	Location	Field
Modem	Port table (NAPORT)	NAMSI
Input state	MLCB	NCISTAI
Text processing state	TPCB	NCSTAI

SET MODEM STATE INDEX

This state instruction sets the modem state index in the port table to a specified value.

Macro Calls

MSTATE   STATE,ACTION

Sets the modem state index to the specified value (STATE).

MJUMP   STATE

Sets the modem state index to the specified value (STATE) then executes this modem state program.

Nonstandard Parameters

STATE Determines the new modem state program index.

Symbolic Name	Value (hexadecimal)	Description
Not specified	0	Default index
---	0-F	Index
MSTCHK	0	Check hard error
MSTERR	1	Error
MSTLNI	2	Line Initialized
MSTENB	3	Enable
MSTIDL	4	Idle
MSTOUT	5	Output
MSTINP	6	Input

Usage

The MSTIDL and MSTINP symbolic names are used by input state programs exclusively. All the other symbolic names are used by modem state programs only.

SET INPUT/TEXT PROCESSING STATE INDEX

This state instruction sets the state program index in the control block to a specified value.

Macro Call

STATE STATE,ACTION

Sets the state program index to the specified value (STATE).

Nonstandard Parameters

STATE Sets the state value.

Symbolic Name	Value (hexadecimal)	Description
Not specified	0	Default. Does not change the index.
---	0-3F	State value

Usage

Changing the state index does not affect the current state process execution. The macro changes states based on incoming character patterns.

JUMP TO INPUT/TEXT PROCESSING STATE

This state instruction executes a given state and optionally updates the control block state program index with the given state.

Macro Calls

JUMP STATE,RTN

RTRN Jumps to the current state process.

Nonstandard Parameters

STATE Sets the state value.

Symbolic Name	Value (hexadecimal)	Description
Not specified	0	Default. Does not change the index.
---	0-3F	State value

RTN

Symbolic Name	Value (hexadecimal)	Description
Not specified	0	Default
---	0	Update state index
---	1	Do not update state index

Usage

The jump instruction allows a state program to pass control to a state process to continue the processing of the current character. The RTN option allows the programmer to suppress changing the state index, so that the next input or source character is processed by the previous state process. The RTN option also provides a method for calling a simple subroutine. If the state parameter is zero, the firmware jumps to the state specified by the state index. The RTRN instruction jumps to the state process indicated by the current value of the state index. Processing begins at the first instruction of this current state.

SKIPS

If the label parameter is within 128-255 locations from the associated state instruction and the instruction is located within 128 locations from the beginning of the program, an informative diagnostic message is produced and the instruction assembles correctly. This is an assembler limitation.

SKIP

This state instruction transfers control by skipping forward or backward.

Macro Calls

SKIP LABEL

Skip forward or backward.

SKIPB LABEL

Skip backward.

The label must be on an instruction that is within ±255 locations from this instruction.

SKIP IF CRC IS EQUAL

This state instruction tests either an 8-bit or 7-bit block check character (BCC) against the accumulated CRC. An equal condition causes the processor to skip to the instruction specified. An unequal condition causes the next state instruction to be executed.

NOTE

> When comparing a hexadecimal (16-bit) CRC polynomial, the first BCC character is accumulated by a state instruction that relinquishes control with a CRCEXIT parameter.

Macro Call

CRCEQ SB,LABEL

Nonstandard Parameters

SB Specifies BCC format

Symbolic Name	Value (hexadecimal)	Description
Not specified	0	Default
B8	0	8-bit BCC
B7	1	7-bit BCC

The label must be on a state instruction that is within 8 locations forward from this instruction.

SKIP IF STATE IS LESS THAN VALUE

This state instruction compares the current state index (input, text, or modem) with a specified value to determine the subsequent state process instruction to perform.

Current state < value: skip

Current state ≥ value: execute next instruction

Macro Calls

STATLS STATE,LABEL

Compares the current state index to the specified value (STATE). The current state is defined in the control block and is either an input state or text processing state.

MSTLS STATE,LABEL

Compares the current modem state index to the specified value (STATE).

Nonstandard Parameters

STATE Specifies the comparison value.

Symbolic Name	Value (hexadecimal)	Description
Not specified	0	Default
---	0-1F	Modem state values
---	0-3F	Input and text processing state values

The label must be on a state instruction that is within 8 locations forward from this instruction.

SKIP IF CHARACTER IS NOT EQUAL

This state instruction compares the current (untranslated) character with a specified character to determine the subsequent state process instruction to perform.

Current character ≠ char: skip

Current character = char: execute next instruction

Macro Call

CHARNE CHAR,LABEL

The label must be on an instruction that is within 8 locations forward from this instruction.

SKIP IF SPECIAL CHARACTER EQUALS CURRENT CHARACTER

This state instruction compares the special character (NCSCHR) to the current (untranslated) character to determine the subsequent state instruction to perform.

Special character ≠ current character: action parameter

Special character = current character: skip

Macro Call

```
SPCHEQ   LABEL,ACTION
```

This instruction must be within 255 locations forward from this instruction.

Usage

This instruction compares an incoming character against a changing value in the line control block. This may be the case if a line has multiple types where different control characters are used for each terminal.

SKIP IF CHARACTER IS LESS THAN OPERAND

This state instruction compares the current (untranslated) character to a specified value to determine the subsequent state process instruction to perform.

Current character < value: skip

Current character ≥ value: execute next instruction

The label must be on an instruction that is within 8 locations forward from this instruction.

PROCESSING CLA STATUS

Each type of communications line adapter (async, sync and HDLC) has its own status words. For these tests, the two status words (8 bits each) are packed into a single computer word (16 bits) with the first communications line adapter status word in the upper half word and the second communications line adapter status word in the lower half word. The three words are defined in figure 5-2.

TEST CLA STATUS

This two-word state instruction checks for a specific positive line status by performing an AND. If the check is satisfied, the next state instruction is executed. Otherwise, the processor skips to a designated instruction.

Macro Call

```
TSTCLA   CMASK,LABEL
```

Nonstandard Parameters

CMASK   Communications line adapter status mask (16 bits). See figure 5-2.

The label must be on a state instruction that is within 8 locations forward from this instruction.

Usage

This instruction is used in input and modem state programs only.

COMPARE CLA STATUS

This two-word state instruction checks the line status for any selected negative line status condition(s) by performing an exclusive AND with the mask followed by an exclusive OR with the mask. If the test result is zero, the next state instruction is executed. If the result is non-zero, the processor skips to the labelled instruction. The communications line adapter status word 1 and word 2 are packed into the upper half and lower half word (of one word) respectively for this check.

Macro Call

```
CMPCLA   CMASK,LABEL
```

Nonstandard Parameters

CMASK   Communications line adapter status mask (16 bits). See figure 5-2.

The label must be on a state instruction that is within 8 locations forward from this instruction.

Usage

This instruction is used in input and modem state programs only.

FLAG CONTROL

These macros control the setting/resetting of various flags in the control block (MLCB or TPCB) and destination buffers.

SET/RESET TRANSLATE FLAG

This state instruction sets or resets the translate flag (NCNOXL) in the control block. Setting the flag causes the current character to be translated before it is stored into the destination buffer. Translation is not performed if the translation address (NCCXLTA) is nil.

Macro Calls

```
SETRAN   ACTION
```

Sets the translation flag.

```
RSTRAN   ACTION
```

Resets the translation flag.

SET/RESET MESSAGE IN PROCESS FLAG

This state instruction sets or resets the input message in process flag maintained in the control block.

Macro Calls

```
SETINP   ACTION
```

Sets the flag.

```
RSTINP   ACTION
```

Resets the flag.

	15				11				7				3			0
Async CLA	CTS	DSR	DCD	RI	SDCD	SQD	ILE	OLE	PES	DTO	FES	–	–	–	–	–
Sync CLA (Mode 4)	CTS	DSR	DCD	RI	QM	SQD	ILE	OLE	PES	DTO	–	NCNA	–	–	–	–
HDLC CLA	CTS	DSR	DCD	RI	QM	SQD	ILE	OLE	FCSE	DTO	ABT	NCNA	LCR	RC1	RC2	RC3

where

ABT	–	Abort
CTS	–	Clear to send
DCD	–	Data carrier detect
DSR	–	Data set ready
DTO	–	Data transfer overrun
FCSE	–	Frame check sequence error
FES	–	Framing error status
HDLC	–	High-level data link control
ILE	–	Input loop error
LCR	–	Last character received
NCNA	–	Next character not available
OLE	–	Output loop error
PES	–	Parity error
QM	–	Quality monitor
RC1 RC2 RC3	–	Reason codes
RI	–	Ring indicator
SDCD	–	Secondary data carrier detector
SQD	–	Signal quality detector

Figure 5-2. CLA Status Bit Assignment

Usage

This instruction is used in input state programs to indicate whether input is active or not active to the macro level TIP. The ASYNC/TTY TIP uses this bit to indicate that a character timeout has occurred.

OPERATE ON USER FLAGS

This state instruction sets, resets or tests the flags in the control block. If any of the tested flags are set, the processor skips to the labelled state instruction. if the tested flag is not set, the next state instruction is executed.

Macro Calls

SETMXF MFLAGS,ACTION

Set user flags (MFLAGS).

RSTMXF MFLAGS,ACTION

Reset user flags (MFLAGS).

TSTMXT MFLAGS,LABEL

Skip (to LABEL) if any user flags (MFLAGS) are set.

Nonstandard Parameters

MFLAGS The 11 user flags in the control block. The flags NCETX, NCMRTP and NCCARR are reserved for modem state use.

Symbolic Name	Value (hexadecimal)	Description
NCUOP1	400	bit 15
NCUOP2	200	bit 14
NCUOP3	100	bit 13
NCUOP4	080	bit 12
NCUOP5	040	bit 11
NCUOP6	020	bit 10
NCUOP7	010	bit 09
NCUOP8	008	bit 08
NCETX	004	bit 07
NCMRTP	002	bit 06
NCCARR	001	bit 05

The label must be on a state instruction that is within 8 locations forward from this instruction.

Usage

The flags are used to record events during processing and to indicate special processing. The initial value of the flags is set for input state processing by calls to the command driver. For text processing the various flags are set on entry and tested on exit for communication between the firmware and the OPS-level portions of the TIP.

SET FLAGS IN THE DESTINATION BUFFER

This state instruction sets selected bits (bits 7 to 1) in the flag word of either the first destination buffer or the current destination buffer. Any bits set at a prior time remain set.

Macro Call

SETFLG FLAGS,BUFF,ACTION

Nonstandard Parameters

FLAGS Selects flags.

Symbolic Name	Value (hexadecimal)	Description
Not specified	0	Default
---	2-7E	Flag bits

BUFF Selects flag word to operate upon.

Symbolic Name	Value (hexadecimal)	Description
Not specified	0	Default
FRST	0	First buffer
CURN	1	Current buffer

Usage

This instruction allows the input state program to record data events in the flag bits of the buffer for communication with the OPS-level portion of the TIP.

SET/RESET PARITY FLAG

This state instruction sets or resets the parity flag in the control block. Setting the flag causes the firmware to strip off the high order bit (bit 7) of the current (untranslated) character before executing the first state instruction. This instruction does not affect the present current character, but rather the next and subsequent current characters until the parity bit resets. During text processing, the setting of the parity flag does not affect the character saved in the file 1 registers.

Macro Calls

SETPAR ACTION

Set the parity flag.

RSTPAR ACTION

Reset the parity flag.

Usage

Stripping the parity bit is advantageous when performing character translation. A translation table contains 128 entries, instead of 256, when translation is used in conjunction with the SETPAR macro.

WORKLIST HANDLING

These instructions build worklists or set a workcode in the appropriate control block (MLCB or TPCB).

TERMINATE INPUT BUFFER

This two-word state instruction terminates input and either builds a worklist entry or stores the workcode in the MLCB. When specified, the end of transmission flag (EOT) in the flag word of the current buffer is set. If a worklist entry is built, the state program determines if it is processed at the multiplex (interrupt level 3) or OPS level. This is done by the selection of the worklist control block.

Macro Calls

TIBWL WC,WL,EOT,ACTION,EP

Terminats the input buffer and builds a worklist entry.

TIBSWC WC,EOT,ACTION

Terminates the input buffer and saves the workcode in the MLCB.

Usage

These instructions are used primarily for input state processing to set the LCB in the final buffer and to signal end of input via a workcode to the OPS-level portion of the

TIP. For text processing, the LCB is also set in the last buffer with the TIBSWC instruction. The creation of a workcode is unnecessary as the text processing is done at OPS level.

The address of the worklist control block is calculated by the Link Edit program. The control blocks are arranged in an array of multiword entries. The origin of the array is an entry point (BYWLCB) which allows the following calculations:

(EP) = BYWLCB + (WLINDEX - (B0FSWL))*
/BYWSIZE

where

BYWLCB = address of worklist control block array

WLINDEX = index of worklist to receive the entry

/BYSIZE = length of worklist entry

The EOT flag is set when the input data is to be transmitted to the host via a coupler. Input state programs are not required to set this bit.

BUILD EVENT WORKLIST

This two-word state instruction generates a worklist entry. Two worklist formats are available. One format places a given workcode and the input buffer pointer from the MLCB into the worklist. The other format obtains the workcode and the first buffer address from the MLCB. Format of a worklist to the OPS-level TIP is as follows:

15	7 0
	Workcode
Line Number	
Current IBP or first buffer address	

Macro Call

BLDWL WC,WL,ACTION,EP

Usage

If the WC parameter is zero, the workcode is the last one saved by TIBSWC. This instruction is used for input state and modem state processing only. The address of the worklist control block is calculated by the Link Edit program. The control blocks are arranged in an array of multiword entries. The origin of the array is an entry point (BYWLCB) which allows the following calculations:

(EP) = BYWLCB + (WLINDEX - (B0FSWL))*
/BYWSIZE

where

BYWLCB = address of worklist control block array

WLINDEX - index of worklist to receive the entry

/BYWSIZE = length of worklist entry

BUILD CLA STATUS WORKLIST ENTRY

This state instruction generates the following communications line adapter status worklist entry to the multiplex level.

15	7 0
SCI	01
Line Number	
SW1	SW2

SCI Status condition indicator
SW1 Status Word 1
SW2 Status Word 2

Macro Call

BLK01 SCI,ACTION

Nonstandard Parameters

SCI Status condition indicator

Symbolic Name	Value (hexadecimal)	Description
Not specified	0	Default
---	0	Pass status to TIP
---	1	Line initialized
---	2	Line enabled
---	3	Hard error(s)
---	4	Soft output error(s)
---	5	Soft input error(s)
---	6	Start modem response time-out (10 sec)
---	7	Stop modem response timeout
---	8	Communications line adapter status overflow
---	9	Communications line adapter status overflow timeout
---	A	Modem response timeout
---	B	Break (FES - from an error status)

Usage

This instruction is used for modem state processing only.

TEXT PROCESSING MACROS

These instructions, used by the text processor, use file 1 registers to modify the current character or perform calculations.

OPERATE ON FILE 1 REGISTER

This state instruction operates on two file 1 registers by either adding, subtracting, or comparing the registers. When adding or subtracting, the result is stored in the register designated by the destination displacement parameter.

Macro Calls

TPADDR SD,DD

Add the contents of the source file 1 register to the contents of the destination file 1 register and store the result in the destination file 1 register.

TPSUBR SD,DD

Subtract the contents of the source file 1 register from the contents of the destination file 1 register and store the result in the destination file 1 register.

TPCMPR SD,DD

Compare the contents of the source file 1 register to the contents of the destination file 1 register. The result determines the next instruction to execute.

(source)   (destination) go to P+1
(source) = (destination) skip to P+2
(source)   (destination) skip to P+3

P is the program address counter.

Usage

This instruction gives the state program a basic computation capability. It is used primarily for text processing.

SET REGISTER VALUE

This state instruction increments or decrements the contents of the selected file 1 register by a specified value.

Macro Calls

TPINCR SD,VALUE

Increment the selected file 1 register by the specified value.

TPDECR SD,VALUE

Decrement the selected file 1 register by the specified value.

Nonstandard Parameters

VALUE Specifies the amount to increment or decrement.

Symbolic Name	Value (hexadecimal)	Description
Not specified	0	Increment by 0 or decrement by 0
---	0-7	Value to increment/ decrement

SAVE/RESTORE TEXT PROCESSING CONDITIONS

This state instruction provides the user with the ability to look ahead before processing the data in a source buffer. The mark function saves the current source and destination buffer pointers, flags, and CRC accumulation; this includes all the necessary information required to get/store the next character in the respective buffer. The information is stored in file 1 registers by the firmware. Two levels of marking are allowed. The backup function restores the information from the file 1 registers for the specified level.

Macro Calls

TPMARK LV

Mark the source and destination buffers at the indicated level.

TPBKUP LV,SRC,DST

Back up to the specified buffer/level.

Nonstandard Parameters

LV Specifies the marking level.

Symbolic Name	Value (hexadecimal)	Description
Not specified	0	Default to level 1
LEVEL1	0	Level 1
LEVEL2	1	Level 2

SRC Specifies the source buffer.

Symbolic Name	Value (hexadecimal)	Description
Not specified	0	Default - null
SRC	1	Source buffer

DST Specifies the destination buffer.

Symbolic Name	Value (hexadecimal)	Description
Not specified	0	Default - null
DST	2	Destination buffer

Usage

This instruction is used in text processing state programs only. Several protocols require a look ahead on the source data to determine the correct transform for the data. Thus, the program records a position in the data and subsequently returns when the correct transform is known.

For TIPs which require that lines not cross transmission block boundaries, the position at the end of a line (or start of a line) is marked. Then, in the event that the line being processed does cross transmission block boundaries, the user can back up to the end of the last line (or start of the current line). Another application is to mark the beginning of a string when compressing characters.

STORE CHARACTER FROM FILE 1 REGISTER

This state instruction, used for text character processing, has two functions:

- It transfers a character from the file 1 register in the register reserved for untranslated characters.
- It stores a character in the destination buffer and optionally accumulates the CRC. If the translate flag in the MUXLCB is on, the character is translated before it is stored. The CRC is accumulated after translation. When the translate flag is off, the untranslated character is stored. Either the left or right byte of the selected file 1 register is stored.

Macro Calls

TPSTLC SD,CRCA

Store the left byte of the file 1 register (SD) in the destination buffer.

TPSTRC SD,CRCA

Store the right byte of the file 1 register (SD) in the destination buffer.

TPRSTL SD

Restores the untranslated character register from the left byte of the file 1 register (SD).

TPRSTR SD

Restores the untranslated character register from the right byte of the file 1 register (SD).

Usage

The restoration of the untranslated character may be accomplished with any file 1 register. However, the restoration is usually done with the first file 1 register (displacement is 0) which contains the current source character. Caution should be used as this copy of the source character does not have the parity bit set to zero even when the parity strip option is selected. The parity bit is always as it is in the source data.

EXIT TEXT PROCESSING

This state instruction causes an exit from the text processing state program and returns to OPS-level processing.

Macro Call

TPEXIT Exit text processing.

Usage

This macro is used to leave text processing after the end of source condition is detected.

INSERT TEXT PROCESSING CHARACTER

This text processing state instruction inserts a character in a destination buffer near a previously marked position.

Macro Call

TPINSR L,S,CHAR,I

Nonstandard Parameters

L Mark level

Symbolic Name	Value (hexadecimal)	Description
Not specified	1	Insert character at a position relative to the level 1 mark
---	2	Insert character at a position relative to the level 2 mark
---	other	Illegal. Causes error message: LEVEL MUST BE ONE OR TWO

C Character source

Symbolic Name	Value (hexadecimal)	Description
Not specified	0	Default Insert character supplied with this instruction
CURNT	1	Insert current source character
other	other	Illegal. Causes error message: ILLEGAL CHARACTER SOURCE

Note that if the symbolic name for CHAR is label, the character associated with the label will be used rather than the CHAR supplied with the instruction.

I Index to position where character is to be inserted

Symbolic Name	Value (hexadecimal)	Description
Not specified	$0 - 7F_{16}$	Determines position of character to be inserted relative to the mark
---	other	Illegal. Causes error message: INDEX OUT OF RANGE

Usage

This instruction is used in text processing state programs only.

MISCELLANEOUS MACROS

SET TRANSLATION TABLE ADDRESS

This two-word state instruction stores the address of a translation table into the control block.

Macro Call

STRNTB TA,ACTION

Set translation table address directly.

STRNTE ACTION,EP

Set up entry point for translation address to be assigned by the link edit program.

Nonstandard Parameters

TA Address of the translation table.

RESET TIMER

This input processing state instruction sets the line control timer (BLTIME) with a specified value for the associated line.

Macro Call

RSTIME TIME,ACTION

Parameters

TIME Sets a time interval for the subsystem timer.

Symbolic Name	Value (hexadecimal)	Description
Not specified	0	Default
---	1-FF	Number of half seconds

Usage

This instruction gives an input state program the ability to set the line timer based on input data. An application sets a short timeout value for the interval between output terminate and start of input. Once input is detected the timer clears, permitting the receipt of the message. This allows for quick detection of a no response condition.

BACKSPACE

This state instruction backspaces the destination buffer pointer one character at a time. Should the pointer cross buffer boundaries while backspacing, the firmware releases the unused destination buffer. However, if backspace is performed on the first character of the first destination buffer, the firmware does not release this buffer.

Macro Call

BKSPAC

RESYNC A SYNCHRONOUS LINE

This state instruction sends a resync command to the communications line adapter instructing it to discard all characters until a sync character is detected.

Macro Call

RESYNC ACTION

Usage

This instruction is used by input state programs for processing synchronous lines.

SET CRC VALUE

This state instruction initializes the cyclic redundancy checksum (CRC) value in the control block for communications lines that require encoding and decoding.

Macro Call

```
INTCRC   ICRC,ACTION
```

Nonstandard Parameters

ICRC   Sets the initial CRC value.

Symbolic Name	Value (hexadecimal)	Description
Not specified	0	Default
ZCRC	0	Set to zero
OCRC	1	Set to all 1's

ALLOCATE A NEW BUFFER

This state instruction gets a new buffer and sets the buffer FCD field. The user-supplied FCD is always an even number. The LCD of the old buffer is updated and a chain to the new buffer is established. If a buffer has not been established, this instruction effectively does a no-op.

Macro Call

```
ALNBUF   FCD,ACTION
```

Parameters

FCD   Defines a displacement to the first data character of the new buffer. This value must be an even number between 4 and $7C_{16}$. An even number forces the first character into the left character position of the word.

Usage

This instruction is used to end an old message, then start a new buffer when a new message is detected, or to break up the data into packets.

NO OPERATION

This state instruction provides the mechanism for specifying the action parameter exclusively. (The action parameter is normally specified as one of the parameters for a state instruction.)

Macro Call

```
NOPR     ACTION
```

MOVE FIELD

This state instruction is used only in text character processing. it allows the user to move specified fields from (1) a file 1 register to another file 1 register, (2) the control block (16 words) to a file 1 register, or (3) a file 1 register to the control block (16 words).

Macro Calls

TPMOVE   SD,DD

Moves the contents (16 bits) of a file 1 register (SD) to another file 1 register (DD).

TPST   SD,DD

Moves the contents (16 bits) of a file 1 register (SD) to the specified (DD) control block word.

TPSTR   SD,DD

Moves the contents of the right byte of the file 1 register (SD) to the right byte of the specified (DD) control block word.

TPSTL   SD,DD

Moves the contents of the right byte of the file 1 register (SD) to the left byte of the specified (DD) control block word.

TPLD   SD,DD

Moves the contents (16 bits) of the specified (SD) control block word to the selected file 1 register (DD).

TPLDR   SD,DD

Moves the right byte of the specified (SD) control block word to the right byte of the designated (DD) file 1 register.

TPLDL   SD,DD

Moves the left byte of the specified (SD) control block word to the right byte of the designated (DD) file 1 register.

Usage

These instructions are useful for moving TPCB fields into the file 1 registers where they can be operated on by the add, subtract, and compare register instructions. They are also used for setting and resetting TPCB fields with user-supplied information in the file 1 registers.

STORE BLOCK LENGTH CHARACTER

This state instruction sets the block length count in the character count 1 (NCCNT1) field of the control block with the current character minus an adjustment.

Macro Call

```
SBLC      ADJ,ACTION
```

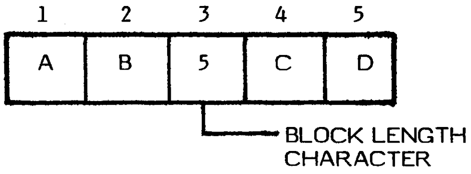
Parameters

ADJ Specifies an adjustment to the start of the block.

Symbolic Name	Value (hexadecimal)	Description
Not specified	0	Default
---	0-FF	Adjustment

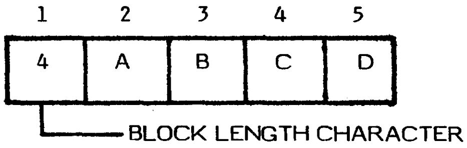
Usage

The adjustment is required if (1) the block length character is included in the block length count, or (2) the block length character is not the first character in the block.



ADJUSTMENT = 3

An adjustment is not required when the block length character is not included in the block length count.



ADJUSTMENT = 0

SUMMARY OF STATE INSTRUCTIONS A

In this appendix, the state instructions are listed alphabetically. The one or two-word macro-assembler packing of the instruction (including its parameter list) is also shown.

Note that the ACTION code always appears in bits 5, 6, and 7 of word 1. If the execution/exit action to be taken is specified by the TIP writer, the label ACTION is used; otherwise, the fixed action code is given. See figure 5-1 for ACTION codes.

The control block of the MLCB (input state processing) or the TPCB (upline or downline text processing).

File 1 registers are numbered 1 to 16; they are indexed 0 to 15.

MACRO **PARAMETERS** **PARAMETER LIST FORMAT**

ADDC CHAR,ACTION Add a character

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>09</td><td>08</td><td>07</td><td>06</td><td>05</td><td>04</td><td>03</td><td>02</td><td>01</td><td>00</td></tr>
<tr><td colspan="8">CHAR</td><td colspan="3">ACTION</td><td colspan="5">11_{16}</td></tr>
</table>

ALNBUF FCD,ACTION Allocate a new buffer

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>09</td><td>08</td><td>07</td><td>06</td><td>05</td><td>04</td><td>03</td><td>02</td><td>01</td><td>00</td></tr>
<tr><td colspan="8">FCD</td><td colspan="3">ACTION</td><td colspan="5">18_{16}</td></tr>
</table>

BKSPAC Backspace

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>09</td><td>08</td><td>07</td><td>06</td><td>05</td><td>04</td><td>03</td><td>02</td><td>01</td><td>00</td></tr>
<tr><td colspan="11">0</td><td colspan="5">$1D_{16}$</td></tr>
</table>

BLCNE COUNT,LABEL Skip if counter value unequal to block length

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>09</td><td>08</td><td>07</td><td>06</td><td>05</td><td>04</td><td>03</td><td>02</td><td>01</td><td>00</td></tr>
<tr><td>A1</td><td>1</td><td colspan="6">0</td><td colspan="3">A7</td><td colspan="5">$1C_{16}$</td></tr>
<tr><td colspan="16">0</td></tr>
</table>

A1 = count -1 A7 = label - *-2

Macro takes the form BLC1NE or BLC2NE where A1 = 0 or 1

BLDWL WC,WL,ACTION,EP Build worklist entry with given workcode

<table>
<tr><td></td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>09</td><td>08</td><td>07</td><td>06</td><td>05</td><td>04</td><td>03</td><td>02</td><td>01</td><td>00</td></tr>
<tr><td></td><td>0</td><td colspan="7">WC</td><td colspan="3">ACTION</td><td colspan="5">03_{16}</td></tr>
<tr><td>EP</td><td colspan="16">WLCB ADDRESS</td></tr>
</table>

WL is ignored but is present in macro call

MACRO	PARAMETERS	PARAMETER LIST FORMAT
BLDWL	WC,WL,ACTION,EP	Build worklist entry with workcode in control block

<table>
<tr><th></th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>09</th><th>08</th><th>07</th><th>06</th><th>05</th><th>04</th><th>03</th><th>02</th><th>01</th><th>00</th></tr>
<tr><td></td><td>1</td><td colspan="7">0</td><td colspan="3">ACTION</td><td colspan="5">03_{16}</td></tr>
<tr><td>EP</td><td colspan="16">WLCB ADDRESS</td></tr>
</table>

WL is ignored, but must be present in the macro call

BLD01	SCI,ACTION	Build CLA status worklist

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>09</th><th>08</th><th>07</th><th>06</th><th>05</th><th>04</th><th>03</th><th>02</th><th>01</th><th>00</th></tr>
<tr><td colspan="8">SCI</td><td colspan="3">ACTION</td><td colspan="5">16_{16}</td></tr>
</table>

CHARLS	CHAR,LABEL	Skip if character < operand

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>09</th><th>08</th><th>07</th><th>06</th><th>05</th><th>04</th><th>03</th><th>02</th><th>01</th><th>00</th></tr>
<tr><td colspan="8">CHAR</td><td colspan="3">A2</td><td colspan="5">$0A_{16}$</td></tr>
</table>

A2 = label - *-1

CHARNE	CHAR,LABEL	Skip if character ≠ operand

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>09</th><th>08</th><th>07</th><th>06</th><th>05</th><th>04</th><th>03</th><th>02</th><th>01</th><th>00</th></tr>
<tr><td colspan="8">CHAR</td><td colspan="3">A2</td><td colspan="5">$0C_{16}$</td></tr>
</table>

A2 = label - *-1

CHRCC	COUNT,IMASK	Mask and set character counter

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>09</th><th>08</th><th>07</th><th>06</th><th>05</th><th>04</th><th>03</th><th>02</th><th>01</th><th>00</th></tr>
<tr><td>A1</td><td colspan="2">0</td><td>1</td><td colspan="7">0</td><td colspan="5">$1C_{16}$</td></tr>
<tr><td colspan="16">IMASK</td></tr>
</table>

A1 = count -1

Macro takes the form of CHRCC1IMASK and CHRCC2IMASK where A1 = 0 or 1

CHRPT		Expand current character

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>09</th><th>08</th><th>07</th><th>06</th><th>05</th><th>04</th><th>03</th><th>02</th><th>01</th><th>00</th></tr>
<tr><td colspan="8">0</td><td colspan="3">7</td><td colspan="5">11_{16}</td></tr>
</table>

MACRO — PARAMETERS — PARAMETER LIST FORMAT

CMPCLA — CMASK,LABEL — Compare CLA status

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>09</th><th>08</th><th>07</th><th>06</th><th>05</th><th>04</th><th>03</th><th>02</th><th>01</th><th>00</th></tr>
<tr><td colspan="8">0</td><td colspan="3">A7</td><td colspan="5">15_{16}</td></tr>
<tr><td colspan="16">CMASK</td></tr>
</table>

A7 = label - *-2

CNTNE — COUNT,CV,LABEL — Skip if character counter does not equal CV

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>09</th><th>08</th><th>07</th><th>06</th><th>05</th><th>04</th><th>03</th><th>02</th><th>01</th><th>00</th></tr>
<tr><td>A1</td><td colspan="3">0</td><td>1</td><td colspan="3">0</td><td colspan="3">A7</td><td colspan="5">10_{16}</td></tr>
<tr><td colspan="16">CV</td></tr>
</table>

A1 = count -1 A7 = label - *-2

Macro also takes the form CNT1NE CV,LABEL and CNT2NE CV, LABEL where A1 = 0 or 1

CRCEQ — SB,LABEL — Skip if CRC equal

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>09</th><th>08</th><th>07</th><th>06</th><th>05</th><th>04</th><th>03</th><th>02</th><th>01</th><th>00</th></tr>
<tr><td>SB</td><td colspan="7">0</td><td colspan="3">A2</td><td colspan="5">05_{16}</td></tr>
</table>

A2 = label - *-1

DCC — COUNT,LABEL,ACTION — Decrement count

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>09</th><th>08</th><th>07</th><th>06</th><th>05</th><th>04</th><th>03</th><th>02</th><th>01</th><th>00</th></tr>
<tr><td>A1</td><td>0</td><td colspan="6">A2</td><td colspan="3">ACTION</td><td colspan="5">06_{16}</td></tr>
</table>

A1 = count -1 A2 = label - *-1

Macro takes the forms DCC1 LABEL,ACTION and DCC2 LABEL,ACTION where A1 = 0 or 1

ICC — COUNT,ACTION — Increment count

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>09</th><th>08</th><th>07</th><th>06</th><th>05</th><th>04</th><th>03</th><th>02</th><th>01</th><th>00</th></tr>
<tr><td>A1</td><td>1</td><td colspan="6">0</td><td colspan="3">ACTION</td><td colspan="5">06_{16}</td></tr>
</table>

A1 = count -1

Macro takes the forms ICC1 ACTION and ICC2 ACTION where A1 = 0 or 1

INTCC — COUNT,ACTION — Initialize count

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>09</th><th>08</th><th>07</th><th>06</th><th>05</th><th>04</th><th>03</th><th>02</th><th>01</th><th>00</th></tr>
<tr><td>A1</td><td colspan="7">0</td><td colspan="3">ACTION</td><td colspan="5">07_{16}</td></tr>
</table>

A1 = count -1

Macro takes the form INTCC1 ACTION and INTCC2 ACTION where A1 = 0 or 1

MACRO	PARAMETERS	PARAMETER LIST FORMAT
INTCRC	ICRC,ACTION	Set CRC initial value

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>09</td><td>08</td><td>07</td><td>06</td><td>05</td><td>04</td><td>03</td><td>02</td><td>01</td><td>00</td></tr>
<tr><td>A3</td><td colspan="3">0</td><td colspan="4">2</td><td colspan="3">ACTION</td><td colspan="5">$1F_{16}$</td></tr>
</table>

A3 = ICRC

JUMP	STATE,RTN	Jump to state

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>09</td><td>08</td><td>07</td><td>06</td><td>05</td><td>04</td><td>03</td><td>02</td><td>01</td><td>00</td></tr>
<tr><td>0</td><td>1</td><td colspan="6">STATE</td><td colspan="3">0</td><td colspan="5">08_{16}</td></tr>
</table>

JUMP	STATE	Update state index and jump

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>09</td><td>08</td><td>07</td><td>06</td><td>05</td><td>04</td><td>03</td><td>02</td><td>01</td><td>00</td></tr>
<tr><td>1</td><td>0</td><td colspan="6">STATE</td><td colspan="3">0</td><td colspan="5">08_{16}</td></tr>
</table>

MJUMP	STATE	Set modem state and execute

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>09</td><td>08</td><td>07</td><td>06</td><td>05</td><td>04</td><td>03</td><td>02</td><td>01</td><td>00</td></tr>
<tr><td>1</td><td colspan="3">0</td><td colspan="4">STATE</td><td colspan="3">0</td><td colspan="5">19_{16}</td></tr>
</table>

MODCC	COUNT,CV	Set count with modulus function

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>09</td><td>08</td><td>07</td><td>06</td><td>05</td><td>04</td><td>03</td><td>02</td><td>01</td><td>00</td></tr>
<tr><td>A1</td><td colspan="10">0</td><td colspan="5">$1C_{16}$</td></tr>
<tr><td colspan="16">CV</td></tr>
</table>

A1 = count -1

MSTATE	STATE,ACTION	Set modem state index

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>09</td><td>08</td><td>07</td><td>06</td><td>05</td><td>04</td><td>03</td><td>02</td><td>01</td><td>00</td></tr>
<tr><td colspan="4">0</td><td colspan="4">STATE</td><td colspan="3">ACTION</td><td colspan="5">19_{16}</td></tr>
</table>

MSTLS	STATE,LABEL	Skip if modem state < operand

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>09</td><td>08</td><td>07</td><td>06</td><td>05</td><td>04</td><td>03</td><td>02</td><td>01</td><td>00</td></tr>
<tr><td>1</td><td colspan="7">STATE</td><td colspan="3">A2</td><td colspan="5">$0B_{16}$</td></tr>
</table>

A2 = label - *-1

NOPR	ACTION	No operation (execute ACTION only)

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>09</td><td>08</td><td>07</td><td>06</td><td>05</td><td>04</td><td>03</td><td>02</td><td>01</td><td>00</td></tr>
<tr><td colspan="8">0</td><td colspan="3">ACTION</td><td colspan="5">00_{16}</td></tr>
</table>

MACRO **PARAMETERS** **PARAMETER LIST FORMAT**

RADDC CHAR Expand (add) current character

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>09</td><td>08</td><td>07</td><td>06</td><td>05</td><td>04</td><td>03</td><td>02</td><td>01</td><td>00</td></tr>
<tr><td colspan="8">CHAR</td><td colspan="3">6</td><td colspan="5">11_{16}</td></tr>
</table>

RESYNC ACTION Resync the line

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>09</td><td>08</td><td>07</td><td>06</td><td>05</td><td>04</td><td>03</td><td>02</td><td>01</td><td>00</td></tr>
<tr><td colspan="4">0</td><td colspan="4">1</td><td colspan="3">ACTION</td><td colspan="5">$1F_{16}$</td></tr>
</table>

RCHAR CHAR,ACTION Replace character

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>09</td><td>08</td><td>07</td><td>06</td><td>05</td><td>04</td><td>03</td><td>02</td><td>01</td><td>00</td></tr>
<tr><td colspan="8">CHAR</td><td colspan="3">ACTION</td><td colspan="5">02_{16}</td></tr>
</table>

RPLACE CHAR,CRCA Replace and store character with CRC

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>09</td><td>08</td><td>07</td><td>06</td><td>05</td><td>04</td><td>03</td><td>02</td><td>01</td><td>00</td></tr>
<tr><td colspan="8">CHAR</td><td colspan="3">0</td><td colspan="5">02_{16}</td></tr>
<tr><td colspan="8">0</td><td colspan="3">3</td><td colspan="5">12_{16}</td></tr>
</table>

RPLACE CHAR Replace and store character without CRC

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>09</td><td>08</td><td>07</td><td>06</td><td>05</td><td>04</td><td>03</td><td>02</td><td>01</td><td>00</td></tr>
<tr><td colspan="8">CHAR</td><td colspan="3">0</td><td colspan="5">02_{16}</td></tr>
<tr><td colspan="8">0</td><td colspan="3">2</td><td colspan="5">12_{16}</td></tr>
</table>

RSTIME TIME,ACTION Reset timer

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>09</td><td>08</td><td>07</td><td>06</td><td>05</td><td>04</td><td>03</td><td>02</td><td>01</td><td>00</td></tr>
<tr><td colspan="8">TIME</td><td colspan="3">ACTION</td><td colspan="5">$1A_{16}$</td></tr>
</table>

RSTINP ACTION Reset input in progress flag

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>09</td><td>08</td><td>07</td><td>06</td><td>05</td><td>04</td><td>03</td><td>02</td><td>01</td><td>00</td></tr>
<tr><td colspan="8">0</td><td colspan="3">ACTION</td><td colspan="5">$1F_{16}$</td></tr>
</table>

RSTMXF MFLAGS,ACTION Reset user flags

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>09</td><td>08</td><td>07</td><td>06</td><td>05</td><td>04</td><td>03</td><td>02</td><td>01</td><td>00</td></tr>
<tr><td colspan="7">0</td><td>1</td><td colspan="3">ACTION</td><td colspan="5">17_{16}</td></tr>
<tr><td colspan="11">MFLAGS</td><td colspan="5">0</td></tr>
</table>

MACRO PARAMETERS PARAMETER LIST FORMAT

RSTPAR ACTION Reset parity flag

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
0								ACTION			$0F_{16}$				

RSTRAN ACTION Reset translate flag

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
0					1	0		ACTION			$0F_{16}$				

RTRN Jump to current state process

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
1	0										08_{16}				

SBLC ADJ,ACTION Store block length in character counter 1

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADJ								ACTION			09_{16}				

SETCC COUNT,CV Set count

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
A1	0	1	0								$1C_{16}$				
CV															

A1 = count -1

Also the forms SETCC1 CV and SETCC2 CV

SETFLG FLAGS,BUFF,ACTION Set flags in buffer

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
FLAGS							A4	ACTION			13_{16}				

A4 = buffer (0 = first 1 = current)

SETINP ACTION Set input in progress flag

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
1	0							ACTION			$1F_{16}$				

SETMXF MFLAGS,ACTION Set user flags

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
0								ACTION			17_{16}				
MFLAGS											0				

MACRO **PARAMETERS** **PARAMETER LIST FORMAT**

SETPAR ACTION Set parity flag

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
1	0							ACTION			$0F_{16}$				

SETRAN ACTION Set translation flag

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
1	0				1	0		ACTION			$0F_{16}$				

SKIP LABEL Skip forward

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
A9								1			00_{16}				

A9 = label - *

SKIPB LABEL Skip backward

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
B1								0			00_{16}				

B1 = * - label

SPCHEQ LABEL,ACTION Skip if special character equals current character

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
A2								ACTION			$0D_{16}$				

A2 = label - * -1

STATE STATE,ACTION Set next state

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
0		STATE						ACTION			08_{16}				

STATLS STATE,LABEL Skip if state < operand

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
0	STATE							A2			$0B_{16}$				

A2 = label - *-1

MACRO PARAMETERS PARAMETER LIST FORMAT

STORC COUNT,ACTION Store count

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>09</th><th>08</th><th>07</th><th>06</th><th>05</th><th>04</th><th>03</th><th>02</th><th>01</th><th>00</th></tr>
<tr><td>A1</td><td colspan="7">0</td><td colspan="3">ACTION</td><td colspan="5">14_{16}</td></tr>
</table>

A1 = count -1

Also STORC1 ACTION and STORC2 ACTION

STORE Store character without CRC

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>09</th><th>08</th><th>07</th><th>06</th><th>05</th><th>04</th><th>03</th><th>02</th><th>01</th><th>00</th></tr>
<tr><td colspan="8">0</td><td colspan="3">2</td><td colspan="5">12_{16}</td></tr>
</table>

STORE CRCA Store character and accumulate CRC

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>09</th><th>08</th><th>07</th><th>06</th><th>05</th><th>04</th><th>03</th><th>02</th><th>01</th><th>00</th></tr>
<tr><td colspan="8">0</td><td colspan="3">3</td><td colspan="5">12_{16}</td></tr>
</table>

STRNTB TA,ACTION Set translation table address

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>09</th><th>08</th><th>07</th><th>06</th><th>05</th><th>04</th><th>03</th><th>02</th><th>01</th><th>00</th></tr>
<tr><td colspan="8">0</td><td colspan="3">ACTION</td><td colspan="5">$1B_{16}$</td></tr>
<tr><td colspan="16">TA</td></tr>
</table>

STRNTE ACTION,EP **Set translation table address**

<table>
<tr><th></th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>09</th><th>08</th><th>07</th><th>06</th><th>05</th><th>04</th><th>03</th><th>02</th><th>01</th><th>00</th></tr>
<tr><td></td><td colspan="8">0</td><td colspan="3">ACTION</td><td colspan="5">$1B_{16}$</td></tr>
<tr><td>EP</td><td colspan="16">TRANSLATION TABLE ADDRESS</td></tr>
</table>

TIBSWC WC,EOT,ACTION Terminate and save workcode

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>09</th><th>08</th><th>07</th><th>06</th><th>05</th><th>04</th><th>03</th><th>02</th><th>01</th><th>00</th></tr>
<tr><td>1</td><td>A5</td><td colspan="6">WC</td><td colspan="3">ACTION</td><td colspan="5">04_{16}</td></tr>
<tr><td colspan="16">0</td></tr>
</table>

A5 = EOT

TIBWL WC,WL,EOT,ACTION,EP Terminate input and build worklist

<table>
<tr><th></th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>09</th><th>08</th><th>07</th><th>06</th><th>05</th><th>04</th><th>03</th><th>02</th><th>01</th><th>00</th></tr>
<tr><td></td><td>0</td><td>A5</td><td colspan="6">WC</td><td colspan="3">ACTION</td><td colspan="5">04_{16}</td></tr>
<tr><td>EP</td><td colspan="16">WLCB ADDRESS</td></tr>
</table>

A5 = EOT

TPADDR SD,DD (SD) + (DD) → (DD)

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>09</th><th>08</th><th>07</th><th>06</th><th>05</th><th>04</th><th>03</th><th>02</th><th>01</th><th>00</th></tr>
<tr><td colspan="4">SD</td><td colspan="4">DD</td><td colspan="3">1</td><td colspan="5">10_{16}</td></tr>
</table>

MACRO | **PARAMETERS** | **PARAMETER LIST FORMAT**

TPBKUP | LV,SRC,DST | Restore text processing conditions

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>09</td><td>08</td><td>07</td><td>06</td><td>05</td><td>04</td><td>03</td><td>02</td><td>01</td><td>00</td></tr>
<tr><td>1</td><td colspan="4">0</td><td>A6</td><td colspan="2">A8</td><td colspan="3">0</td><td colspan="5">$1E_{16}$</td></tr>
</table>

A6 = LV-1 A8 = SRC + DST

TPCMPR | SD,DD | Compare file 1 registers

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>09</td><td>08</td><td>07</td><td>06</td><td>05</td><td>04</td><td>03</td><td>02</td><td>01</td><td>00</td></tr>
<tr><td colspan="4">SD</td><td colspan="4">DD</td><td colspan="3">3</td><td colspan="5">10_{16}</td></tr>
</table>

TPDECR | SD,VALUE | Decrement file 1 register

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>09</td><td>08</td><td>07</td><td>06</td><td>05</td><td>04</td><td>03</td><td>02</td><td>01</td><td>00</td></tr>
<tr><td>1</td><td colspan="3">VALUE</td><td colspan="4">SD</td><td colspan="3">0</td><td colspan="5">10_{16}</td></tr>
</table>

TPEXIT | | Exit from text processing

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>09</td><td>08</td><td>07</td><td>06</td><td>05</td><td>04</td><td>03</td><td>02</td><td>01</td><td>00</td></tr>
<tr><td colspan="8">0</td><td colspan="3">1</td><td colspan="5">$1E_{16}$</td></tr>
</table>

TPINCR | SD,VALUE | Increment file 1 register

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>09</td><td>08</td><td>07</td><td>06</td><td>05</td><td>04</td><td>03</td><td>02</td><td>01</td><td>00</td></tr>
<tr><td>0</td><td colspan="3">VALUE</td><td colspan="4">SD</td><td colspan="3">0</td><td colspan="5">10_{16}</td></tr>
</table>

TPSINSR | L,S,CHAR,I | Insert text processing character

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>09</td><td>08</td><td>07</td><td>06</td><td>05</td><td>04</td><td>03</td><td>02</td><td>01</td><td>00</td></tr>
<tr><td>L</td><td>0</td><td>0</td><td>S</td><td>0</td><td>0</td><td>1</td><td>1</td><td colspan="8">$1F_{16}$</td></tr>
<tr><td colspan="8">I</td><td colspan="8">CHAR</td></tr>
</table>

TPLD | SD,DD | Move control block word to file 1 register

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>09</td><td>08</td><td>07</td><td>06</td><td>05</td><td>04</td><td>03</td><td>02</td><td>01</td><td>00</td></tr>
<tr><td colspan="4">SD</td><td colspan="4">DD</td><td colspan="3">4</td><td colspan="5">$0E_{16}$</td></tr>
</table>

TPLDL | SD,DD | Move left byte of control block word to file 1 register

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>09</td><td>08</td><td>07</td><td>06</td><td>05</td><td>04</td><td>03</td><td>02</td><td>01</td><td>00</td></tr>
<tr><td colspan="4">SD</td><td colspan="4">DD</td><td colspan="3">6</td><td colspan="5">$0E_{16}$</td></tr>
</table>

MACRO	PARAMETERS	PARAMETER LIST FORMAT
TPLDR	SD,DD	Move right byte of control block word to file 1 register

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SD				DD				5			$0E_{16}$				

TPMARK	LV	Save buffer conditions

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
0					A6	0		0			$1E_{16}$				

A6 = LV-1

TPMOVE	SD,DD	Move register to register

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SD				DD				0			$0E_{16}$				

TPRSTL	SD	Restore from left byte of file 1 register

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
0				SD				0			01_{16}				

TPRSTR	SD	Restore from right byte of file 1 register

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
1	0			SD				0			01_{16}				

TPSTL	SD,DD	Move right byte of file 1 register to left byte of control block word

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SD				DD				3			$0E_{16}$				

TPSTLC	SD,CRCA	Store left byte of file 1 register into destination buffer with CRC

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
0				SD				3			01_{16}				

TPSTLC	SD	Store left byte of file 1 register into destination buffer without CRC

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
0				SD				2			01_{16}				

MACRO	PARAMETERS	PARAMETER LIST FORMAT
TPSTR	SD,DD	Move right byte of file 1 register to right byte of control block word

15 14 13 12	11 10 09 08	07 06 05	04 03 02 01 00
SD	DD	2	$0E_{16}$

TPSTRC	SD,CRCA	Store right byte of file 1 register into destination buffer with CRC

15	14 13 12	11 10 09 08	07 06 05	04 03 02 01 00
1	0	SD	3	07_{16}

TPSTRC	SD	Store right byte of file 1 register into destination buffer without CRC

15	14 13 12	11 10 09 08	07 06 05	04 03 02 01 00
1	0	SD	2	07_{16}

TPSUBR	SD,DD	Subtract file 1 register

15 14 13 12	11 10 09 08	07 06 05	04 03 02 01 00
SD	DD	2	10_{16}

TPST	SD,DD	Move file 1 register to control block

15 14 13 12	11 10 09 08	07 06 05	04 03 02 01 00
SD	DD	1	$0E_{16}$

TSTCLA	CMASK,LABEL	Test CLA status

15	14 13 12 11 10 09 08	07 06 05	04 03 02 01 00
1	0	A7	15_{16}
CMASK			

A7 = label - *-2

TSTMXF	MFLAGS,LABEL	Test user flags

15 14 13 12 11 10	09	08	07 06 05	04 03 02 01 00
0	1	0	A7	17_{16}
MFLAGS				0

A7 = label - * -2

STATE INSTRUCTION TIMING B

Timing for input, output, and text processing is calculated by using the following tables. All timing values are expressed in microseconds.

TABLE B-1. EXECUTION TIMES FOR INPUT/TEXT PROCESSING DEPENDENT INSTRUCTIONS

Task - Per Character	Input	Text Processing
Get character	12.8	5.5
Number of instructions x 2.2	---	---
Instruction execution time(s) (See Section B.2)	---	---
Translation (select one) On 3.1 Off 1.5	---	---
CRC (select one) Yes 4.9 No 0.0	---	---
Store character	4.8	4.8
Exit	2.2	1.5
Task - Per Character	**Input**	**Text Processing**
Get and chain a destination buffer	15.0	16.0
Chain a source buffer	---	6.6
Release a buffer	11.4	11.4
Make a worklist	6.9	6.9
Start-up	---	10.1
PTTPINF interface	---	135.0

TABLE B-2. STATE INSTRUCTION EXECUTION TIMES

Macro	Execution Time	Description
ADDC	2.3 7.1	Add a character (including store)
ALNBUF	10.8	Allocate a new buffer
BKSPAC	3.9	Backspace (not over buffer boundary)
BLCNE	5.0	Skip if count not equal block length
BLDWL	16.1	Build worklist entry with given workcode
BLDWL	10.4	Build worklist entry with workcode in control block
BLK01	14.5	Build CLA status worklist
CHARLS	1.2	Skip if char < operand
CHARNE	1.4	Skip if char not equal operand
CHRCC	5.0	Mask and set char counter
CHRPT	9.4	Expand (one) character
CMPCLA	2.6	Compare CLA status
CNTNE	5.0	Skip if char count not equal
CRCEQ	2.0	Skip if CRC equal
DCC	2.9	Decrement count
ICC	2.9	Increment count
INTCC	1.8	Initialize count
INTCRC	2.8	Set CRC initial value
JUMP	4.0	Jump to state
JUMP	5.4	Update state index and jump
MJUMP	3.4	Set modem state and execute
MODCC	5.0	Set count with mod function
MSTATE	3.4	Set modem state index
MSTLS	2.3	Skip if modem state < operand
NOPR	1.5	No operation
RADDC	9.4 3.1	Expand (one) character (each additional 2 chars)
RESYNC	8.8	Resync the line
RCHAR	0.5	Replace character
RPLACE	6.7	Replace and store character

TABLE B-2. STATE INSTRUCTION EXECUTION TIMES (Contd)

Macro	Execution Time	Description
RSTIME	3.4	Reset timer
RSTINP	2.5	Reset input in progress flag
RSTMXF	3.9	Reset user flags
RSTPAR	2.5	Reset parity flag
RSTRAN	1.9	Reset translate flag
RTRN	4.0	Jump to current state process
SBLC	1.4	Store block length in character counter 1
SETCC	5.0	Set count
SETFLG	3.4	Set flags in buffer
SETINP	2.5	Set input in progress flaq
SETMXF	3.9	Set user flags
SETPAR	2.5	Set parity flag
SETRAN	1.9	Set translation flag
SKIP	1.5	Skip forward
SKIPB	1.5	Skip backward
SPCHEQ	1.8	Skip if special char = char
STATE	4.0	Set next state
STATLS	2.3	Skip if state operand
STORC	3.2	Store count
STORE	1.4	Store character
STRNTB	2.0	Set translation table address
STRNTE	--	Set translation table address
TIBSWC	10.4	Terminate input and save workcode
TIBWL	16.1	Terminate input and build worklist
TPADDR	5.2	(SD) + (DD) (DD)

TABLE B-2. STATE INSTRUCTION EXECUTION TIMES (Contd)

Macro	Execution Time	Description
TPBKUP	9.4	Restore TP conditions
TPCMPR	5.2	Compare file 1 registers
TPDECR	5.2	Decrement file 1 register
TPEXIT	2.8	Exit text processing
TPINCR	5.2	Increment file 1 register
TPINSR	--	Insert text processing character
TPLD	4.4	Move control block word to file 1 register
TPLDL	4.4	Move left byte of control block word to file 1 register
TPLDR	4.4	Move right byte of control block word to file 1 register
TPMARK	6.3	Save buffer conditions
TPMOVE	4.4	Move register to register
TPRSTL	2.3	Restore from left byte of file 1 register
TPRSTR	2.3	Restore from right byte of file 1 register
TPSTL	4.4	Move right byte of file 1 reqister to left byte of control block word
TPSTLC	2.3	Store left byte of file 1 register into test buffer
TPSTR	4.4	Move right byte of file 1 register to right byte of control block word
TPSTRC	2.3	Store right byte of file 1 register into test buffer
TPSUBR	5.2	Subtract file 1 register
TPST	4.4	Move file 1 register to control block
TSTCLA	2.6	Test CLA status
TSTMXF	3.9	Test user flags

JOB DECK STRUCTURE FOR ASSEMBLING STATE PROGRAMS

(To be supplied later)

SAMPLE STATE PROGRAM D

This sample is the input state program (first pass) for the HASP TIP. Since there is no code or format conversion in this first pass state processing, this comparatively simple state program is only concerned with moving data from the circular input buffer (CIB) to the input source buffer, and then notifying the TIP that the data is ready for upline text processing.

This appendix has the following subsections:

- Equates
- Input state program pointers table (HSINST)
- Input state processes making up the input state program

```
                     *******************************************
                     *                                         *
                     *           HASP STATE PROGRAMS AND       *
                     *              TRANSLATION TABLES         *
                     *                  ASSEMBLIES             *
                     *                                         *
                     *******************************************

                           NAM     HSR4IPS



                 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                 *                                                                   *
                 *        MUX SUBSYSTEM EQUATES                                      *
                 *                                                                   *
                 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                 *
0004                      EQU     MXETX($4)     ETX FLAG FOR CLA STATUS HANDLER
0002                      EQU     MXMRTO(2)     RESPONS TIMEOUT
0001                      EQU     MXCARR($1)    CONTROLLED CARRIER FLAG
0000                      EQU     MSTCHK(0)
0001                      EQU     MSTERR(1)
0002                      EQU     MSTLNI(2)
0003                      EQU     MSTENB(3)
0004                      EQU     MSTIDL(4)
0005                      EQU     MSTOUT(5)
0006                      EQU     MSTINP(6)
                 *
                 * * *   MUX FLAGS
                 *
0400                      EQU     NCUOP1($400) BIT 15
0200                      EQU     NCUOP2($200) BIT 14
0100                      EQU     NCUOP3($100) BIT 13
0080                      EQU     NCUOP4($080) BIT 12
0040                      EQU     NCUOP5($040) BIT 11
0020                      EQU     NCUOP6($020) BIT 10
0010                      EQU     NCUOP7($010) BIT  9
0008                      EQU     NCUOP8($008) BIT  8
0004                      EQU     NCUOP9($004) BIT  7 (TEXT PROCESSING ONLY)
0002                      EQU     NCUOPA($002) BIT  6 (TEXT PROCESSING ONLY)
0001                      EQU     NCUOPB($001) BIT  5 (TEXT PROCESSING ONLY)



                 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                 *                                                                   *
                 *    WORK CODES                                                     *
                 *                                                                   *
                 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                 *
0003                      EQU     MMBUTCH(3)    MUX BUFFER THRESHOLD
0021                      EQU     A0WK1($21)
0022                      EQU     A0WK2(A0WK1+1)
0023                      EQU     A0WK3(A0WK2+1)
0024                      EQU     A0WK4(A0WK3+1)
0025                      EQU     A0WK5(A0WK4+1)
0026                      EQU     A0WK6(A0WK5+1)
0027                      EQU     A0WK7(A0WK6+1)
0028                      EQU     A0WK8(A0WK7+1)
0029                      EQU     A0WK9(A0WK8+1)
002A                      EQU     A0WK10(A0WK9+1)
002B                      EQU     A0WK11(A0WK10+1)
002C                      EQU     A0WK12(A0WK11+1)
002D                      EQU     A0WK13(A0WK12+1)
002E                      EQU     A0WK14(A0WK13+1)
002F                      EQU     A0WK15(A0WK14+1)
0030                      EQU     A0WK16(A0WK15+1)
0031                      EQU     A0WK17(A0WK16+1)
0032                      EQU     A0WK18(A0WK17+1)
0033                      EQU     A0WK19(A0WK18+1)
0034                      EQU     A0WK20(A0WK19+1)
0035                      EQU     A0WK21(A0WK20+1)
0036                      EQU     A0WK22(A0WK21+1)
```

```
                 ****************************************
                 *                                      *
                 * HASP REL4 CONSTANT EQUATES           *
                 *                                      *
                 ****************************************

0001                  EQU     HCSOH($01)     *    BSC OUTER PROTOCOL CHARACTERS
0002                  EQU     HCSTX($02)
0010                  EQU     HCDLE($10)
0026                  EQU     HCETB($26)
002D                  EQU     HCENQ($2D)
0032                  EQU     HCSYN($32)
003D                  EQU     HCNAK($3D)
0070                  EQU     HCACK($70)

0000                  EQU     HCZERO($0)     *     CHARACTER 0
00F0                  EQU     HCCONTROL($F0)       CONTROL RCB
00C1                  EQU     HCSIGNON($C1)        SIGNON SRCB

0014                  EQU     HWKWLNO($14) *    HASP WORKLIST NUMBER
0021                  EQU     HWKENQ(A0WK1)     ENQ RECEIVED WORKCODE
0022                  EQU     HWKERR(HWKENQ+1)  ERR RECEIVED WORKCODE
0023                  EQU     HWKACK(HWKENQ+2)  ACK RECEIVED WORKCODE
0024                  EQU     HWKNAK(HWKENQ+3)  NAK RECEIVED WORKCODE
0025                  EQU     HWKMSG(HWKENQ+4)  MSG RECEIVED WORKCODE
0026                  EQU     HWKBTH(HWKENQ+5)  BUFFER THRESHOLD WORKCODE

0001                  EQU     HFNEW($01)
0002                  EQU     HFXPT($02)

00C0                  EQU     HNONCMP($C0) *    NON COMPRESSED DATA SCB
00A0                  EQU     HCMPNBLKS($A0)    COMPRESSED NON BLANKS SCB

003F                  EQU     HMNCMSK($3F) *    NON-COMPRESSED-DATA SCE MASK
0010                  EQU     HMXPT(16)    *    TRANSPARENT DATA MASK
001F                  EQU     HMCBMSK($1F) *    COMPRESSED BLANKS MASK
001F                  EQU     HMCNBMSK($1F)     COMPRESSED NON-BLANKS MASK
00FF                  EQU     HMCHRMSK($FF)     CHARACTER MASK
```

```
                                      *********************************************************************
                                      *********************************************************************
                                      *                                                                   *
                                      *         HASP INPUT STATE PROGRAMS (1ST PASS) POINTER TABLE        *
                                      *                                                                   *
                                      *********************************************************************
                                      *********************************************************************
                                      HSINST     MAC    NM
                                                 EQU    HS≠NM≠(*-HINSPT)
                                                 ADC    H≠NM≠
           POINTER                               EMC
                                      *
                         0000 P                  ENT    HINSPT
                         0000 P                  EQU    HINSPT(*)
P0000        0018 P                              HSINST CLASTAT  0
P0001        0019 P                              HSINST DCDNOT   1    STANDARD DEFINITIONS FOR
P0002        0028 P                              HSINST OVERUN   2    INPUT STATE PROGRAMS
P0003        0029 P                              HSINST BUTHR    3
P0004        002E P                              HSINST INIT
P0005        0036 P                              HSINST DAT0
P0006        0041 P                              HSINST SOH
P0007        004B P                              HSINST DLE0
P0008        0057 P                              HSINST BCB
P0009        005D P                              HSINST LFCS
P000A        0066 P                              HSINST RFCS
P000B        006B P                              HSINST 1RCB
P000C        0076 P                              HSINST CONTROL
P000D        007F P                              HSINST SRCB
P000E        0084 P                              HSINST SCB
P000F        0095 P                              HSINST DATA
P0010        009E P                              HSINST DLE
P0011        00A2 P                              HSINST SIGNON
P0012        00A6 P                              HSINST ETB
P0013        00AD P                              HSINST 1CRC
P0014        00AE P                              HSINST 2CRC
P0015        00B3 P                              HSINST ERROR
P0016        00B6 P                              HSINST TERM
P0017        00C2 P                              HSINST IDLE
```

```
                        ************************************************************************
                        *                                                                      *
                        *         HSCLASTAT - CLA STATUS HANDLER                               *
                        *                                                                      *
                        ************************************************************************
                        ************************************************************************
P0018     0020          HCLASTAT  NOPR   EXIT          IGNORE STATUS
P0019                   ************************************************************************
                        ************************************************************************
                        *                                                                      *
                        *         HSDCDNCT - DATA-CARRIER-DETECT DROPPED                       *
                        *                                                                      *
                        ************************************************************************
                        ************************************************************************
P0019     0237          HDCDNOT   TSTMXF MXCARR,HDCD1 *   SKIP IF     CONTROLLED CARRIER
P001A     0020
P001B     013F                    RESYNC EXIT         *   RESYNC CLA AND EXIT
P001C     0237          HDCD1     TSTMXF MXETX,HDCD2  *   SKIP IF WORKLIST WANTED
P001D     0080
P001E     013F                    RESYNC EXIT         *   RESYNC CLA AND EXIT
P001F     842B          HDCD2     MSTLS  MSTIDL,HDCD3 DOUBLE CHECK THAT MODEM STATE IS IDLE
P0020     852B                    MSTLS  MSTIDL+1,HDCD4
P0021     013F          HDCD3     RESYNC EXIT         *   MODEM STATE NOT IDLE
P0022     0117          HDCD4     RSTMXF MXETX        *   CLEAR WL ENTRY NEEDED FLAG
P0023     0080
P0024     001A                    RSTIME 0            *   STOP TIMER
P0025     8003                    BLDWL  ,,,HWCRK2    *   BUILD WL ENTRY
P0026     0000
P0027     013F                    RESYNC EXIT         *   RESYNC CLA AND EXIT
P0028                   ************************************************************************
                        ************************************************************************
                        *                                                                      *
                        *         HSOVERUN - TOO MANY BUFFERS                                  *
                        *                                                                      *
                        ************************************************************************
                        ************************************************************************
P0028     5508          HOVERUN   JUMP   HSERRCR,RTN  GOTO STATE ERROR REMEMBER CUR STATE
P0029                   ************************************************************************
                        ************************************************************************
                        *                                                                      *
                        *         HSBUTHR - BUFFER-THRESHOLD REACHED IN SYSTEM                 *
                        *                                                                      *
                        ************************************************************************
                        ************************************************************************
P0029     0304          HBUTHR    TIBWL  MMBUTCH      *   TELL MUX SS TO RELEASE BUFFERS
P002A     0000
P002B     A604                    TIBSWC HWKBTH       *   MAKE BUFFER THRESHOLD WLE
P002C     0000
P002D     9608                    JUMP   HSTERM       *   TERMINATE INPUT
P002E                   ************************************************************************
                        ************************************************************************
                        *                                                                      *
                        *         HSINIT - INITIAL INPUT STATE                                 *
                        *                                                                      *
                        ************************************************************************
                        ************************************************************************
P002E     32CC          HINIT     CHARNE HCSYN,HINIT1 LOCK FOR SYN CHAR
P002F     0117                    RSTMXF HMXPT        RESET MUX XPT FLAG
P0030     0200
P0031     0117                    RSTMXF MXETX        *   CLEAR ETX FLAG
P0032     0080
P0033     C619                    MSTATE MSTINP       *   SET MODEM STATE INPUT
P0034     C528                    STATE  HSDAT0,EXIT  IT IS - SWITCH TO DATA ARRIVING
P0035     013F          HINIT1    RESYNC EXIT         IT ISNT - RESYNC CLA
P0036                   ************************************************************************
                        ************************************************************************
                        *                                                                      *
                        *         HSDAT0 - DATA ARRIVING                                       *
                        *                                                                      *
                        ************************************************************************
                        ************************************************************************
P0036     322C          HDAT0     CHARNE HCSYN,HDAT01 SYN CHAR
P0037     0020                    NOPR   EXIT         YES - IGNORE
P0038     012C          HDAT01    CHARNE HCSOH,HDAT02 SOH
P0039     0628                    STATE  HSSOH,EXIT   YES
P003A     1C2C          HDAT02    CHARNE HCDLE,HDAT03 DLE
P003B     0728                    STATE  HSDLE0,EXIT
P003C     3D6C          HDAT03    CHARNE HCNAK,HDAT05 NAK
P003D     A404                    TIBSWC HWKNAK       *   YES- NAK WLE TO TIP
P003E     0000
P003F     9608                    JUMP   HSTERM       *   TERMINATE INPUT
P0040     8408          HDAT05    JUMP   HSINIT       *   ALLOW LINE TO RESYNC
P0041                   ************************************************************************
                        ************************************************************************
                        *                                                                      *
                        *         HSSOH - SOH RECEIVED                                         *
                        *                                                                      *
                        ************************************************************************
                        ************************************************************************
P0041     322C          HSOH      CHARNE HCSYN,HSOH1  SYN
P0042     0020                    NOPR   EXIT         YES - IGNORE
P0043     2D6C          HSOH1     CHARNE HCENQ,HSOH2  ENQ
P0044     A104                    TIBSWC HWKENQ       *   YES- ENQ WLE TO TIP
P0045     0000
P0046     9608                    JUMP   HSTERM       *   TERMINATE INPUT
P0047     024C          HSOH2     CHARNE HCSTX,HSOH3  STX
P0048     021F                    INTCRC ZCRC         *    INITIALIZE CRC ACCUM
P0049     0888                    STATE  HSBCB,CRCEXIT
P004A     8408          HSOH3     JUMP   HSINIT       *   ALLOW LINE TO RESYNC
```

```
P004B                          *******************************************************************
                               *******************************************************************
                               *                                                                 *
                               *          HSDLE0 - DLE RECEIVED                                  *
                               *                                                                 *
                               *******************************************************************
                               *******************************************************************
P004B     322C                 HDLE0      CHARNE HCSYN,HDLE01 SYN
P004C     0528                            STATE  HSDAT0,EXIT  YES - IGNORE
P004D     706C                 HDLE01     CHARNE HCACK,HDLE02 ACK
P004E     A304                            TIBSWC HWKACK       *   YES- ACK WLE TO TIP
P004F     0000

P0050     9608                            JUMP   HSTERM       *   TERMINATE INPUT
P0051     028C                 HDLE02     CHARNE HCSTX,HDLE03 STX
P0052     0017                            SETMXF HMXPT        SET MUX XPT FLAG
P0053     0200

P0054     021F                            INTCRC ZCRC         *    INITIALIZE CRC ACCUM
P0055     0828                            STATE  HSBCB,EXIT
P0056     8408                 HDLE03     JUMP   HSINIT       *   ALLOW LINE TO RESYNC
P0057                          *******************************************************************
                               *******************************************************************
                               *                                                                 *
                               *          HSBCB - PROCESS BCB                                    *
                               *                                                                 *
                               *******************************************************************
                               *******************************************************************
                    0057 P     HBCB       EQU    HBCB(*)
P0057     322C                            CHARNE HCSYN,HBCB1
P0058     0020                            NOPR   EXIT         IGNORE
P0059     102C                 HBCB1      CHARNE HCDLE,HBCB2  DLE
P005A     0020                            NOPR   EXIT         IGNORE
P005B     0011                 HBCB2      ADDC   HCZERO       ADD DUMMY FOR RIGHT-CHAR-ALLIGNMENT
P005C     0968                            STATE  HSLFCS,CRCSTOREX  STORE BCB,CRC AND EXIT
P005D                          *******************************************************************
                               *******************************************************************
                               *                                                                 *
                               *          HSLFCS - PROCESS LEFT FCS                              *
                               *                                                                 *
                               *******************************************************************
                               *******************************************************************
P005D     322C                 HLFCS      CHARNE HCSYN,HLFCS1 SYN
P005E     0020                            NOPR   EXIT         IGNORE
P005F     102C                 HLFCS1     CHARNE HCDLE,HLFCS2 DLE
P0060     0020                            NOPR   EXIT         IGNORE
P0061     0237                 HLFCS2     TSTMXF HMXPT,HLFCS3 SKIP IF XPT-FLAG SET
P0062     0200

P0063     0220                            SKIP   HLFCS4
P0064     0513                 HLFCS3     SETFLG HFXPT,CURN   SET XPT-FLAG IN FIRST-BUFFER
P0065     0A68                 HLFCS4     STATE  HSRFCS,CRCSTOREX STORE LFCS,CRC AND EXIT
P0066                          *******************************************************************
                               *******************************************************************
                               *                                                                 *
                               *          HSRFCS - PROCESS RIGHT FCS                             *
                               *                                                                 *
                               *******************************************************************
                               *******************************************************************
P0066     322C                 HRFCS      CHARNE HCSYN,HRFCS1 SYN
P0067     0020                            NOPR   EXIT         IGNORE
P0068     102C                 HRFCS1     CHARNE HCDLE,HRFCS2 DLE
P0069     0020                            NOPR   EXIT         IGNORE
P006A     0868                 HRFCS2     STATE  HS1RCB,CRCSTOREX  STORE RFCS,CRC AND EXIT
P006B                          *******************************************************************
                               *******************************************************************
                               *                                                                 *
                               *          HS1RCB - PROCESS FIRST / NEXT RCB                      *
                               *                                                                 *
                               *******************************************************************
                               *******************************************************************
P006B     322C                 H1RCB      CHARNE HCSYN,H1RCB1 SYN
P006C     0020                            NOPR   EXIT         IGNORE
P006D     102C                 H1RCB1     CHARNE HCDLE,H1RCB2 DLE
P006E     0020                            NOPR   EXIT         IGNORE
P006F     002C                 H1RCB2     CHARNE HCZERO,H1RCB5    NO (MORE) RECORDS
P0070     1288                            STATE  HSETB,CRCEXIT    DONE, LOOK FOR ETB
P0071     262C                 H1RCB5     CHARNE HCETB,H1RCB3 ETB WITHOUT ZERO RCB
P0072     1388                            STATE  HS1CRC,CRCEXIT   YES GO PROCESS CRC NOW
P0073     F02C                 H1RCB3     CHARNE HCCONTROL,H1RCB4 NO - CONTROL RECORD
P0074     0C88                            STATE  HSCONTROL,CRCEXIT  PROCESS CONTROL SRCB
P0075     0D68                 H1RCB4     STATE  HSSRCB,CRCSTOREX  NO - GET SRCB
P0076                          *******************************************************************
                               *******************************************************************
                               *                                                                 *
                               *          HSCONTROL - CONTROL RCB RECEIVED,LOOK AT SRCB          *
                               *                                                                 *
                               *******************************************************************
                               *******************************************************************
P0076     322C                 HCONTROL   CHARNE HCSYN,HCON1  SYN
P0077     0020                            NOPR   EXIT         IGNORE
P0078     102C                 HCON1      CHARNE HCDLE,HCON2  DLE
P0079     0020                            NOPR   EXIT         IGNORE
P007A     C16C                 HCON2      CHARNE HCSIGNON,HCON3   SIGNON
P007B     A01C                            SETCC2 HC80         YES - SET 80 CHAR LENGTH
P007C     0050

P007D     1188                            STATE  HSSIGNON,CRCEXIT PROCESS THE SIGNON + THROW AWAY SRCB
P007E     0E68                 HCON3      STATE  HSSCB,CRCSTOREX  NO - PROCESS NORMALLY
```

```
P007F                     ***********************************************************************
                          ***********************************************************************
                          *                                                                     *
                          *          HSSRCB - PROCESS SRCBS                                     *
                          *                                                                     *
                          ***********************************************************************
                          ***********************************************************************
P007F     322C            HSRCB      CHARNE HCSYN,HSRCB1 SYN
P0080     0020                       NOPR   EXIT         IGNORE
P0081     102C            HSRCB1     CHARNE HCDLE,HSRCB2 DLE
P0082     0020                       NOPR   EXIT         IGNORE
P0083     0E68            HSRCB2     STATE  HSSCB,CRCSTOREX      CRC STORE AND EXIT
P0084                     ***********************************************************************
                          ***********************************************************************
                          *                                                                     *
                          *          HSSCB - PROCESS SCBS                                       *
                          *                                                                     *
                          ***********************************************************************
                          ***********************************************************************
P0084     322C            HSCB       CHARNE HCSYN,HSCB1  SYN
P0085     0020                       NOPR   EXIT         IGNORE
P0086     102C            HSCB1      CHARNE HCDLE,HSCB1A DLE
P0087     0020                       NOPR   EXIT         IGNORE
P0088     262C            HSCB1A     CHARNE HCETB,HSCB2  ETB
P0089     1388                       STATE  HS1CRC,CRCEXIT PROCESS CRC
P008A     002C            HSCB2      CHARNE HCZERO,HSCB3 EOR
P008B     0868                       STATE  HS1RCB,CRCSTOREX YES - GET NEXT RCB
P008C     C06A            HSCB3      CHARLS HNONCMP,HSCB4    NON - COMPRESSED
P008D     901C                       CHRCC2 HMNCMSK      SET COUNT TO NUM OF NON COMPRESSED
P008E     003F
P008F     0F68                       STATE  HSDATA,CRCSTOREX SET DATA STATE CRC, STORE AND EXIT
P0090     A06A            HSCB4      CHARLS HCMPNBLKS,HSCB5  COMPRESSED NON BLANK
P0091     A01C                       SETCC2 HCONE        SET COUNT TO ONE
P0092     0001
P0093     0F68                       STATE  HSDATA,CRCSTOREX SET DATA STATE CRC ,STCRE AND EXIT
P0094     0060            HSCB5      NOPR   CRCSTOREX    COMPRESSED BLANKS - STORE SCB,CRC,EX
P0095                     ***********************************************************************
                          ***********************************************************************
                          *                                                                     *
                          *          HSDATA - PROCESS CHARACTERS AFTER SCB                      *
                          *                                                                     *
                          ***********************************************************************
                          ***********************************************************************
P0095     32AC            HDATA      CHARNE HCSYN,HDATA3 IS CHAR A SYN
P0096     0237                       TSTMXF HMXPT,HDATA1 YES - XPT WORKSTATION
P0097     0200
P0098     0020                       NOPR   EXIT         NO - IGNORE
P0099     8066            HDATA1     CCC2   HDATA2,CRCSTOREX   YES SO PROCESS IT
P009A     0E68            HDATA2     STATE  HSSCB,CRCSTOREX   UNTIL DONE
P009B     102C            HDATA3     CHARNE HCDLE,HDATA4 DLE
P009C     1028                       STATE  HSDLE,EXIT   YES - PROCESS IT
P009D     0400            HDATA4     SKIPB  HDATA1       NOT DLE - PROCESS CHARACTER
                          ***********************************************************************
                          ***********************************************************************
                          *                                                                     *
                          *          HSDLE - PROCESS CHAR AFTER DLE                             *
                          *                                                                     *
                          ***********************************************************************
                          ***********************************************************************
P009E     322C            HDLE       CHARNE HCSYN,HDLE1  SYN
P009F     0F28                       STATE  HSDATA,EXIT  IGNORE
P00A0     0F08            HDLE1      STATE  HSDATA       OTHERWISE SET STAE BACK TO DATA
P00A1     0800                       SKIPB  HDATA1       AND PROCESS THIS CHARACTER
                          ***********************************************************************
                          ***********************************************************************
                          *                                                                     *
                          *          HSSIGNON - PROCESS SIGNON-CARD                             *
                          *                                                                     *
                          ***********************************************************************
                          ***********************************************************************
P00A2     262C            HSIGNON    CHARNE HCETB,HSIGN2 *    CHECK FOR EARLY ETB
P00A3     1388                       STATE  HS1CRC,CRCEXIT    LOOK FOR CRC
P00A4     8086            HSIGN2     CCC2   HSIGN1,CRCEXIT    ACCUM CRC, DISCARD DATA
P00A5     0E88            HSIGN1     STATE  HSSCB,CRCEXIT    UNTIL DONE ALL 80
P00A6                     ***********************************************************************
                          ***********************************************************************
                          *                                                                     *
                          *          HSETB - PROCESS ETB                                        *
                          *                                                                     *
                          ***********************************************************************
                          ***********************************************************************
P00A6     322C            HETB       CHARNE HCSYN,HETB1  SYN
P00A7     0020                       NOPR   EXIT         IGNORE
P00A8     102C            HETB1      CHARNE HCDLE,HETB2  DLE
P00A9     0020                       NOPR   EXIT         IGNORE
P00AA     262C            HETB2      CHARNE HCETB,HETB3  ETB
P00AB     1388                       STATE  HS1CRC,CRCEXIT    PROCESS
P00AC     5508            HETB3      JUMP   HSERROR,RTN  GOTO STATE ERROR REMEMBER CUR STATE
P00AD                     ***********************************************************************
                          ***********************************************************************
                          *                                                                     *
                          *          HS1CRC - PROCESS LEFT CRC                                  *
                          *                                                                     *
                          ***********************************************************************
                          ***********************************************************************
P00AD     1488            H1CRC      STATE  HS2CRC,CRCEXIT    SET FOR RIGHT CRC ,CRC AND EXIT
```

```
P00AE                 ****************************************************************
                      ****************************************************************
                      *
                      *         HS2CRC - PROCESS RIGHT CRC
                      *
                      ****************************************************************
                      ****************************************************************
P00AE     0025        H2CRC     CRCEQ  B8,H2CRC1    CRC EQUAL
P00AF     5508                  JUMP   HSERROR,RTN  NO, ERROR
P00B0     A504        H2CRC1    TIBSWC HWKMSG       *   YES- WLE TO TIP
P00B1     0000
P00B2     9608                  JUMP   HSTERM       *   TERMINATE INPUT
P00B3                 ****************************************************************
                      ****************************************************************
                      *                                                              *
                      *         HSERROR - ERROR IN DATA MESSAGE                      *
                      *                                                              *
                      ****************************************************************
                      ****************************************************************
P00B3     A204        HERROR    TIBSWC HWKERR       *   GIVE TIP AN ERROR WLE
P00B4     0000
P00B5     9608                  JUMP   HSTERM       *   TERMINATE INPUT
P00B6                 ****************************************************************
                      ****************************************************************
                      *                                                              *
                      *         HSTERM - TERMINATE INPUT                             *
                      *                                                              *
                      ****************************************************************
                      ****************************************************************
P00B6     0419        HTERM     MSTATE MSTIDLE      *   SET MODEM STATE TO IDLE
P00B7     02D7                  TSTMXF MXCARR,HTERM1    SKIP IF CONTROLLED CARRIER
P00B8     0020
P00B9     001A                  RSTIME 0            *   TURN OFF TIMER
P00BA     0117                  RSTMXF MXETX        *   RESET ETX FLAG
P00BB     0080
P00BC     8003                  BLOWL  ,,,HWORK1    *   MAKE WLE W/ SAVED WORKCODE
P00BD     8000
P00BE     9708                  JUMP   HSIDLE       *   WAIT AT IDLE
P00BF     0017        HTERM1    SETMXF MXETX        *   SET ETX FLAG
P00C0     0080
P00C1     9708                  JUMP   HSIDLE       *   WAIT AT IDLE
P00C2                 ****************************************************************
                      ****************************************************************
                      *                                                              *
                      *         HSIDLE - ALL DONE,IGNORE ANY ARRIVING DATA           *
                      *                                                              *
                      ****************************************************************
                      ****************************************************************
P00C2     013F        HIDLE     RESYNC EXIT             RESYNC CLA
```

INDEX

COMMENT SHEET

MANUAL TITLE: State Programming Language Reference Manual

PUBLICATION NO.: 60472200 **REVISION:** C

NAME:____

COMPANY:____

STREET ADDRESS:____

CITY:____________________ **STATE:**__________ **ZIP CODE:**__________

This form is not intended to be used as an order blank. Control Data Corporation welcomes your evaluation of this manual. Please indicate any errors, suggested additions or deletions, or general comments below (please include page number references).

CUT ALONG LINE

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

FOLD ON DOTTED LINES AND TAPE

TAPE TAPE

FOLD FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 8241 MINNEAPOLIS, MINN.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION
Publications and Graphics Division
P. O. Box 4380-P
Anaheim, California 92803

CUT ALONG LINE

FOLD FOLD

CORPORATE HEADQUARTERS, P.O. BOX 0, MINNEAPOLIS, MINN. 55440
SALES OFFICES AND SERVICE CENTERS IN MAJOR CITIES THROUGHOUT THE WORLD

LITHO IN U.S.A.

CONTROL DATA CORPORATION