



---

# 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**



# 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:

| <u>Publication Title</u>   | <u>Publication Number</u> |
|--|---------------------------|
| Communications Control Program<br>Version 3<br>System Programmer's Reference Manual  | 60474500                  |
| Communications Control INTERCOM<br>Version 3<br>System Programmer's Reference Manual | 60471160                  |
| Communications Control Module<br>Version 3<br>Reference Manual                       | 60470500                  |
| Macro Assembler<br>Reference Manual<br>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

|   |   |
|---|---|
| <p>1. INTRODUCTION 1-1</p> <p>Program Interface 1-1</p> <p>State Program Structure 1-1</p> <p>Manual Format 1-2</p> <p>2. INPUT STATE PROGRAMS 2-1</p> <p>Firmware Interface 2-1</p> <p>Program Control 2-1</p> <p>Data Structure for Input State Program: MLCB 2-2</p> <p>Program Organization 2-3</p> <p>Interface to the Modem State Programs 2-3</p> <p>3. TEXT PROCESSING STATE PROGRAMS 3-1</p> <p>Data Structure, TPCB 3-1</p> <p>Firmware Interface 3-2</p> <p>TPCB Initial Set-Up 3-2</p> <p>TPCB Set-Up for Restart 3-2</p> <p>TPCB Return Values 3-2</p> <p>File 1 Text Processing Registers 3-3</p> <p>Program Control 3-3</p> <p>Program Organization 3-3</p> <p>4. MODEM STATE PROGRAMS 4-1</p> <p>Firmware Interface 4-1</p> <p>Program Control 4-1</p> <p>Program Organization 4-1</p> <p>Interface to the Multiplex Level Status Handler 4-1</p> <p>Interface to the Input State Programs 4-2</p> <p>5. STATE INSTRUCTIONS 5-1</p> <p>Handling Assignable Counter 5-1</p> <p>Initialize Character Counter 5-1</p> <p>Set Character Counter 5-1</p> <p>Mask and Set Character Counter 5-1</p> <p>Set Character Counter with Mod Function 5-1</p> <p>Increment Character Counter 5-3</p> <p>Decrement Character Counter 5-3</p> <p>Compare Character Counter to a Value 5-3</p> <p>Compare Character Counter to Block Length 5-3</p> | <p>Store Character Counter in Buffer 5-3</p> <p>Character Manipulation 5-3</p> <p>Store Character 5-3</p> <p>Replace Character 5-3</p> <p>Replace and Store Character 5-4</p> <p>Add (Insert) a Character 5-4</p> <p>Expand (Repeat) Character 5-4</p> <p>Index Manipulations 5-4</p> <p>Set Modem State Index 5-4</p> <p>Set Input/Text Processing State Index 5-5</p> <p>Jump to Input/Text Processing State 5-5</p> <p>Skips 5-5</p> <p>Skip 5-6</p> <p>Skip if CRC is Equal 5-6</p> <p>Skip if State is Less Than Value 5-6</p> <p>Skip if Character is Not Equal 5-6</p> <p>Skip if Special Character Equals Current Character 5-6</p> <p>Skip if Character is Less Than Operand 5-7</p> <p>Processing CLA Status 5-7</p> <p>Test CLA Status 5-7</p> <p>Compare CLA Status 5-7</p> <p>Flag Control 5-7</p> <p>Set/Reset Translate Flag 5-7</p> <p>Set/Reset Message in Process Flag 5-7</p> <p>Operate on User Flags 5-8</p> <p>Set Flags in the Destination Buffer 5-9</p> <p>Set/Reset Parity Flag 5-9</p> <p>Worklist Handling 5-9</p> <p>Terminate Input Buffer 5-9</p> <p>Build Event Worklist 5-10</p> <p>Build CLA Status Worklist Entry 5-10</p> <p>Text Processing Macros 5-11</p> <p>Operate on File 1 Register 5-11</p> <p>Set Register Value 5-11</p> <p>Save/Restore Text Processing Conditions 5-11</p> <p>Store Character from File 1 Register 5-12</p> <p>Exit Text Processing 5-12</p> <p>Insert Text Processing Character 5-12</p> <p>Miscellaneous Macros 5-13</p> <p>Set Translation Table Address 5-13</p> <p>Reset Timer 5-13</p> <p>Backspace 5-13</p> <p>Resync a Synchronous Line 5-13</p> <p>Set CRC Value 5-14</p> <p>Allocate a New Buffer 5-14</p> <p>No Operation 5-14</p> <p>Move Field 5-14</p> <p>Store Block Length Character 5-14</p> |
|---|---|

## APPENDIXES

|    |                               |     |    |                           |     |
|----|-------------------------------|-----|----|---------------------------|-----|
| A. | SUMMARY OF STATE INSTRUCTIONS | A-1 | C. | JOB DECK STRUCTURE FOR    |     |
| B. | STATE INSTRUCTION TIMING      | B-1 |    | ASSEMBLING STATE PROGRAMS | C-1 |
|    |                               |     | D. | SAMPLE STATE PROGRAM      | D-1 |

## INDEX

## FIGURES

|     |                                 |     |     |                                      |     |
|-----|---------------------------------|-----|-----|--------------------------------------|-----|
| 1-1 | State Pointer Table Creation    | 1-1 | 4-1 | Locating a Modern State Process      | 4-1 |
| 2-1 | Locating an Input State Process | 2-1 | 5-1 | Standard Macro Parameter Definitions | 5-2 |
| 2-2 | Standard MLCB                   | 2-2 | 5-2 | CLA Status Bit Assignment            | 5-8 |
| 3-1 | Standard TPCB                   | 3-1 |     |                                      |     |



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
      ADC      mnemonic
      FMC      U ≠ NM ≠
*
      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.

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.

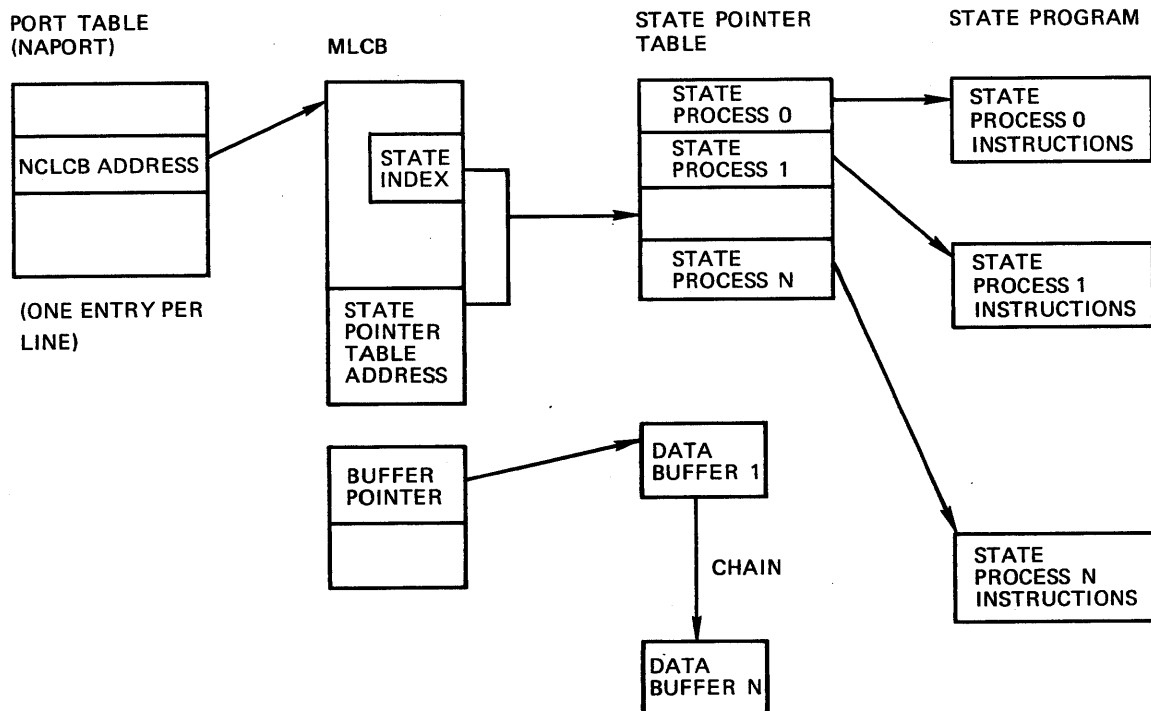


Figure 2-1. Locating an Input State Process

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

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

Flags:

|     |   |     |  |
|-----|---|-----|--|
| F1  | = NCEOBL – end of block                           | F17 | = NCRPRT – strips parity bit                                   |
| F2  | = NCNCOCA – next output character available       | F18 | = NCSCF – suppress chain flag                                  |
| F3  | = NCLCT – last character transmitted (CDDCP)      | F19 | = NCLASTCH – LCD of source buffer reached                      |
| F4  | = NCBCREQ – buffer chaining required              | F20 | = NCEOSR – end of source buffer reached                        |
| F5  | = NCOMPRO – output message in progress            | F21 | = NCSP3 – not used   |
| F6  | = NCSP1 – not used                                | F22 | = NCUOP1   |
| F7  | = NCODDIN – ODD received                          | F23 | = NCUOP2   |
| F8  | = NCSP2 – not used                                | F24 | = NCUOP3   |
| F9  | = NCSUPCHAIN – suppress buffer chaining           | F25 | = NCUOP4   |
| F10 | = NCOBT – generate output buffer terminated (OBT) | F26 | = NCUOP5   |
| F11 | = NCBZL – reset timer                             | F27 | = NCUOP6   |
| F12 | = NCRINCH – input character in right byte         | F28 | = NCUOP7   |
| F13 | = NCCAREC – character received                    | F29 | = NCUOP8   |
| F14 | = NCRIGHTC – left/right source flag (1 = right)   | F30 | = NCETX – Delay ETX worklist generation                        |
| F15 | = NCINPRO – input message in progress             | F31 | = NCMRTO – Modem response timed out                            |
| F16 | = NCNOXL – code translation active                | F32 | = NCCARR – Line carrier type (1 = controlled;<br>0 = constant) |

} optional user flags

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.

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.

|    | 15  | 14  | 13  | 12                       | 11                              | 10  | 9   | 8   | 7                              | 6   | 5                                   | 4                       | 3 | 2 | 1 | 0 |
|----|---|-----|-----|--------------------------|---------------------------------|-----|-----|-----|--------------------------------|-----|-------------------------------------|-------------------------|---|---|---|---|
| 0  | NCLCDFCD – SOURCE BUFFER LCD/FCD                  |     |     |                          |                                 |     |     |     |                                |     |                                     |                         |   |   |   |   |
| 1  | F9  | F10 | F11 | NCTIME – MULTIPLEX TIMER |                                 |     |     |     | NCOBLCD – LCD OF OUTPUT BUFFER |     |                                     |                         |   |   |   |   |
| 2  | NCSBP – SOURCE BUFFER POINTERS                    |     |     |                          |                                 |     |     |     |                                |     |                                     |                         |   |   |   |   |
| 3  | F12   | F13 | F14 | F15                      | F16                             | F17 | F18 | F19 | F20                            | F21 | NCISTA1 – INPUT STATE PROGRAM INDEX |                         |   |   |   |   |
| 4  | NCCNTL – CHARACTER COUNT LIMIT                    |     |     |                          |                                 |     |     |     | NCCNT1 – CHARACTER COUNTER 1   |     |                                     |                         |   |   |   |   |
| 5  | NCSPTA – POINTER TO STATE PROGRAMS POINTERS TABLE |     |     |                          |                                 |     |     |     |                                |     |                                     |                         |   |   |   |   |
| 6  | NCDBP – POINTER TO STATE PROGRAMS TABLE           |     |     |                          |                                 |     |     |     |                                |     |                                     |                         |   |   |   |   |
| 7  | F22   | F23 | F24 | F25                      | F26                             | F27 | F28 | F29 | F30                            | F31 | F32                                 | NCCRCP – CRC POLYNOMIAL |   |   |   |   |
| 8  | NCSCHR – SPECIAL CHARACTER                        |     |     |                          |                                 |     |     |     | NC1BFCD – FCD OF INPUT BUFFER  |     |                                     |                         |   |   |   |   |
| 9  | NCCRCS – CRC ACCUMULATION                         |     |     |                          |                                 |     |     |     |                                |     |                                     |                         |   |   |   |   |
| 10 | NCZER1 – ZERO                                     |     |     |                          | NCCNT1 – CHARACTER COUNTER 2    |     |     |     |                                |     |                                     |                         |   |   |   |   |
| 11 | NCZER2 – ZERO                                     |     |     |                          | NCBLK1 – BLOCK LENGTH (RECORDS) |     |     |     |                                |     |                                     |                         |   |   |   |   |
| 12 | NCCXLTA – POINTER TO CODE TRANSLATE TABLE         |     |     |                          |                                 |     |     |     |                                |     |                                     |                         |   |   |   |   |
| 13 | NCFDBA – POINTER TO FIRST DESTINATION BUFFER      |     |     |                          |                                 |     |     |     |                                |     |                                     |                         |   |   |   |   |
| 14 | NCBLCNT – NUMBER OF BUFFERS ALLOCATED             |     |     |                          |                                 |     |     |     | NCSVWL – SAVED WORKLIST        |     |                                     |                         |   |   |   |   |
| 15 | RESERVED  |     |     |                          |                                 |     |     |     |                                |     |                                     |                         |   |   |   |   |
| 16 | NCDUMD  |     |     |                          |                                 |     |     |     |                                |     |                                     |                         |   |   |   |   |
| 17 | NCDUME  |     |     |                          |                                 |     |     |     |                                |     |                                     |                         |   |   |   |   |
| 18 | NCFBSA – FIRST STORAGE BUFFER ADDRESS             |     |     |                          |                                 |     |     |     |                                |     |                                     |                         |   |   |   |   |
| 19 | RESERVED FOR TIP USAGE                            |     |     |                          |                                 |     |     |     |                                |     |                                     |                         |   |   |   |   |
| 20 | RESERVED FOR TIP USAGE                            |     |     |                          |                                 |     |     |     |                                |     |                                     |                         |   |   |   |   |
| 21 | RESERVED FOR TIP USAGE                            |     |     |                          |                                 |     |     |     |                                |     |                                     |                         |   |   |   |   |
| 22 | RESERVED FOR TIP USAGE                            |     |     |                          |                                 |     |     |     |                                |     |                                     |                         |   |   |   |   |
| 23 | RESERVED FOR TIP USAGE                            |     |     |                          |                                 |     |     |     |                                |     |                                     |                         |   |   |   |   |
| 24 | RESERVED FOR TIP USAGE                            |     |     |                          |                                 |     |     |     |                                |     |                                     |                         |   |   |   |   |
| 25 | RESERVED FOR TIP USAGE                            |     |     |                          |                                 |     |     |     |                                |     |                                     |                         |   |   |   |   |
| 26 | RESERVED FOR TIP USAGE                            |     |     |                          |                                 |     |     |     |                                |     |                                     |                         |   |   |   |   |
| 27 | RESERVED FOR TIP USAGE                            |     |     |                          |                                 |     |     |     |                                |     |                                     |                         |   |   |   |   |
| 28 | RESERVED FOR TIP USAGE                            |     |     |                          |                                 |     |     |     |                                |     |                                     |                         |   |   |   |   |
| 29 | RESERVED FOR TIP USAGE                            |     |     |                          |                                 |     |     |     |                                |     |                                     |                         |   |   |   |   |
| 30 | RESERVED FOR TIP USAGE                            |     |     |                          |                                 |     |     |     |                                |     |                                     |                         |   |   |   |   |
| 31 | RESERVED FOR TIP USAGE                            |     |     |                          |                                 |     |     |     |                                |     |                                     |                         |   |   |   |   |

M-422

Figure 3-1. Standard TPCB

## FIRMWARE INTERFACE

The procedure PTPINF 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 PTPINF (TPCB) where the TPCB is contained in a data buffer. A pointer variable of type BOBUFPTR 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.
- NCFBSA 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 NCFBSA 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 NCFBSA 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:

- NCFBSA - 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 text 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.



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 accessing 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 progress.

## 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.

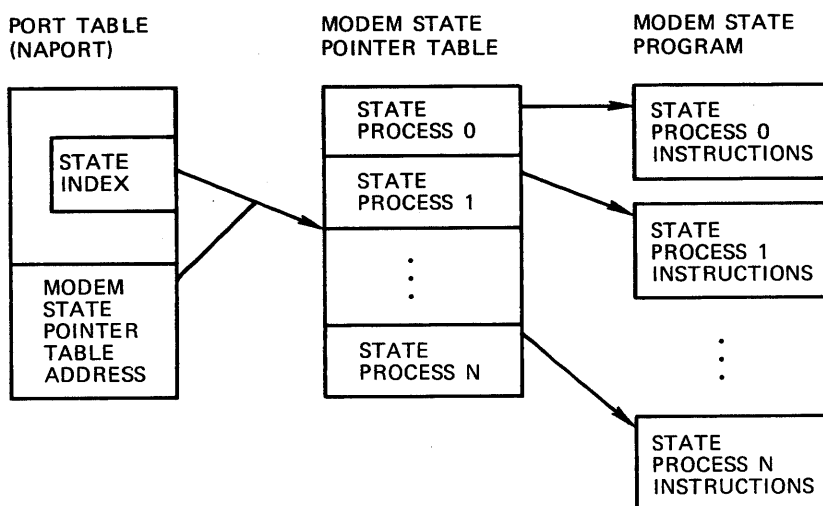


Figure 4-1. Locating a Modem State Process

## 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.

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 used 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 modulo 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.   |                                |   |
|        | <u>Symbolic Name</u>   | <u>Value</u>                   | <u>Description</u>                          |
|        | 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  | <u>Symbolic Name</u>   | <u>Value</u>                   | <u>Description</u>                          |
|        | Not specified  | 0                              | Error                                       |
|        | -  | 1                              | Count 1                                     |
|        | -  | 2                              | Count 2                                     |
| CRCA   | <u>Symbolic Name</u>   | <u>Value</u>                   | <u>Description</u>                          |
|        | 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.  |                                |   |
|        | <u>Symbolic Name</u>   | <u>Value</u>                   | <u>Description</u>                          |
|        | Not specified  | 0                              | File 1 register (first)                     |
|        | -  | 0-15                           | File 1 register (first through 16th)        |
| EOT    | <u>Symbolic Name</u>   | <u>Value</u>                   | <u>Description</u>                          |
|        | 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.   |                                |   |
|        | <u>Symbolic Name</u>   | <u>Value</u>                   | <u>Description</u>                          |
|        | 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.  |                                |   |
|        | <u>Symbolic Name</u>   | <u>Value<br/>(hexadecimal)</u> | <u>Description</u>                          |
|        | Not specified  | 0                              | Default                                     |
|        | -  | 0                              | Use saved workcode                          |
|        | -  | 1-7F                           | Use given workcode                          |
|        |  |                                | } Multiplex or OPS-level                    |
| 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  $\neq$  value: skip

### Macro Call

CNTNE     COUNT,CV,LABEL

Use specified character count (COUNT).

Labeled instruction is within  $\pm 8$  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  $\neq$  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:

| <u>Index</u>          | <u>Location</u>        | <u>Field</u> |
|-----------------------|------------------------|--------------|
| Modem                 | Port table<br>(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.

| <u>Symbolic Name</u> | <u>Value (hexadecimal)</u> | <u>Description</u> |
|----------------------|----------------------------|--------------------|
| 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.

| <u>Symbolic Name</u> | <u>Value (hexadecimal)</u> | <u>Description</u>                  |
|----------------------|----------------------------|-------------------------------------|
| 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.

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

### RTN

| <u>Symbolic Name</u> | <u>Value (hexadecimal)</u> | <u>Description</u>        |
|----------------------|----------------------------|---------------------------|
| 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  $\pm 255$  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

| <u>Symbolic Name</u> | <u>Value (hexadecimal)</u> | <u>Description</u> |
|----------------------|----------------------------|--------------------|
| 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  $\geq$  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.

| <u>Symbolic Name</u> | <u>Value (hexadecimal)</u> | <u>Description</u>                     |
|----------------------|----------------------------|--|
| 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  $\neq$  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  $\neq$  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.

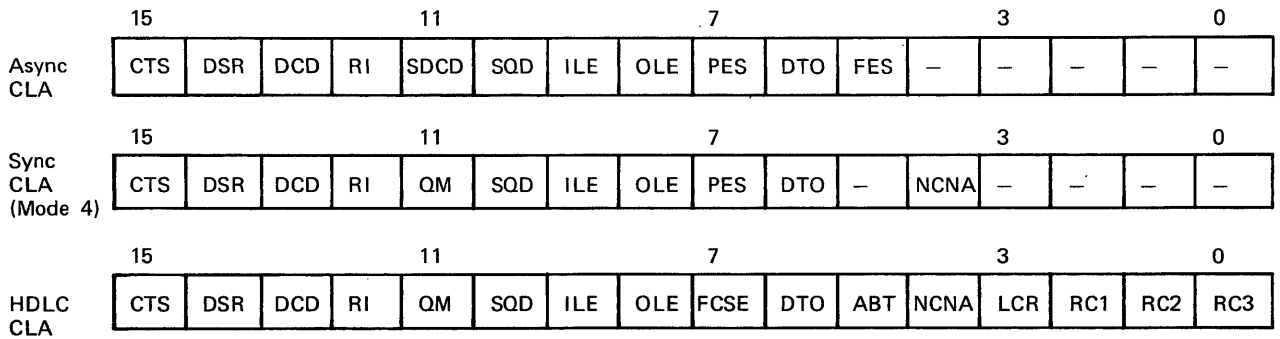
## Macro Calls

SETINP ACTION

Sets the flag.

RSTINP ACTION

Resets the flag.



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 } - Reason codes  
RC3 }
- 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.

| <u>Symbolic Name</u> | <u>Value (hexadecimal)</u> | <u>Description</u> |
|----------------------|----------------------------|--------------------|
| 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.

| <u>Symbolic Name</u> | <u>Value (hexadecimal)</u> | <u>Description</u> |
|----------------------|----------------------------|--------------------|
| Not specified        | 0                          | Default            |
| ---                  | 2-7E                       | Flag bits          |

**BUFF** Selects flag word to operate upon.

| <u>Symbolic Name</u> | <u>Value (hexadecimal)</u> | <u>Description</u> |
|----------------------|----------------------------|--------------------|
| 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

Terminates 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

/BYWSIZE = 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.

| <u>Symbolic Name</u> | <u>Value (hexadecimal)</u> | <u>Description</u>               |
|----------------------|----------------------------|----------------------------------|
| 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.

| <u>Symbolic Name</u> | <u>Value (hexadecimal)</u> | <u>Description</u> |
|----------------------|----------------------------|--------------------|
| Not specified        | 0                          | Default to level 1 |
| LEVEL1               | 0                          | Level 1            |
| LEVEL2               | 1                          | Level 2            |

SRC Specifies the source buffer.

| <u>Symbolic Name</u> | <u>Value (hexadecimal)</u> | <u>Description</u> |
|----------------------|----------------------------|--------------------|
| Not specified        | 0                          | Default - null     |
| SRC                  | 1                          | Source buffer      |

DST Specifies the destination buffer.

| <u>Symbolic Name</u> | <u>Value (hexadecimal)</u> | <u>Description</u> |
|----------------------|----------------------------|--------------------|
| 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       |                      |   |
|   |                  | <u>Symbolic Name</u> | <u>Value (hexadecimal)</u>                                  |
|   |                  |                      | <u>Description</u>  |
|   |                  | 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 |                      |   |

| <u>Symbolic Name</u> | <u>Value (hexadecimal)</u> | <u>Description</u>   |
|----------------------|----------------------------|--|
| Not specified        | 0                          | Default<br>Insert character supplied with this instruction |
| CURNT                | 1                          | Insert current source character                            |
| other                | other                      | Illegal. Causes error message:<br>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

| <u>Symbolic Name</u> | <u>Value (hexadecimal)</u> | <u>Description</u>   |
|----------------------|----------------------------|--|
| Not specified        | 0-7F <sub>16</sub>         | Determines position of character to be inserted relative to the mark |
| ---                  | other                      | Illegal.<br>Causes error message:<br>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.

| <u>Symbolic Name</u> | <u>Value (hexadecimal)</u> | <u>Description</u>     |
|----------------------|----------------------------|------------------------|
| 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.

| <u>Symbolic Name</u> | <u>Value (hexadecimal)</u> | <u>Description</u> |
|----------------------|----------------------------|--------------------|
| 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<sub>16</sub>. 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

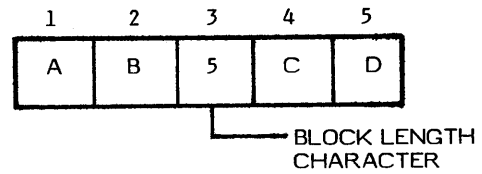
**Parameters**

ADJ Specifies an adjustment to the start of the block.

| <u>Symbolic Name</u> | <u>Value (hexadecimal)</u> | <u>Description</u> |
|----------------------|----------------------------|--------------------|
| Not specified        | 0                          | Default            |
| ---                  | 0-FF                       | Adjustment         |

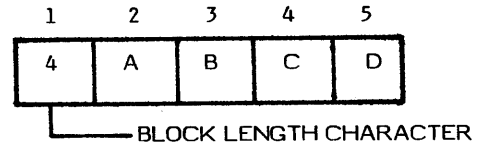
**Usage**

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).

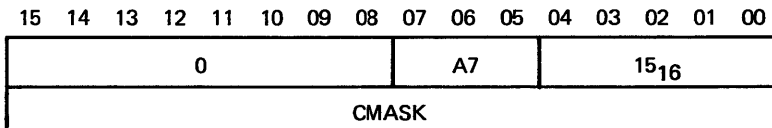
File 1 registers are numbered 1 to 16; they are indexed 0 to 15.

| MACRO  | PARAMETERS      | PARAMETER LIST FORMAT  |    |    |    |    |    |        |    |                  |                  |    |    |    |    |    |    |    |      |    |   |  |  |  |  |    |        |  |                  |                  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|--|-----------------|--|----|----|----|----|----|--------|----|------------------|------------------|----|----|----|----|----|----|----|------|----|---|--|--|--|--|----|--------|--|------------------|------------------|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|
| ADDC   | CHAR,ACTION     | Add a character  |    |    |    |    |    |        |    |                  |                  |    |    |    |    |    |    |    |      |    |   |  |  |  |  |    |        |  |                  |                  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |                 | <table border="1"> <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<sub>16</sub></td> </tr> </table>  | 15 | 14 | 13 | 12 | 11 | 10     | 09 | 08               | 07               | 06 | 05 | 04 | 03 | 02 | 01 | 00 | CHAR |    |   |  |  |  |  |    | ACTION |  |                  | 11 <sub>16</sub> |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 15   | 14              | 13   | 12 | 11 | 10 | 09 | 08 | 07     | 06 | 05               | 04               | 03 | 02 | 01 | 00 |    |    |    |      |    |   |  |  |  |  |    |        |  |                  |                  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| CHAR   |                 |  |    |    |    |    |    | ACTION |    |                  | 11 <sub>16</sub> |    |    |    |    |    |    |    |      |    |   |  |  |  |  |    |        |  |                  |                  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| ALNBUF   | FCD,ACTION      | Allocate a new buffer  |    |    |    |    |    |        |    |                  |                  |    |    |    |    |    |    |    |      |    |   |  |  |  |  |    |        |  |                  |                  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |                 | <table border="1"> <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<sub>16</sub></td> </tr> </table>   | 15 | 14 | 13 | 12 | 11 | 10     | 09 | 08               | 07               | 06 | 05 | 04 | 03 | 02 | 01 | 00 | FCD  |    |   |  |  |  |  |    | ACTION |  |                  | 18 <sub>16</sub> |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 15   | 14              | 13   | 12 | 11 | 10 | 09 | 08 | 07     | 06 | 05               | 04               | 03 | 02 | 01 | 00 |    |    |    |      |    |   |  |  |  |  |    |        |  |                  |                  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| FCD  |                 |  |    |    |    |    |    | ACTION |    |                  | 18 <sub>16</sub> |    |    |    |    |    |    |    |      |    |   |  |  |  |  |    |        |  |                  |                  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| BKSPAC   |                 | Backspace  |    |    |    |    |    |        |    |                  |                  |    |    |    |    |    |    |    |      |    |   |  |  |  |  |    |        |  |                  |                  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |                 | <table border="1"> <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<sub>16</sub></td> </tr> </table>  | 15 | 14 | 13 | 12 | 11 | 10     | 09 | 08               | 07               | 06 | 05 | 04 | 03 | 02 | 01 | 00 | 0    |    |   |  |  |  |  |    |        |  |                  | 1D <sub>16</sub> |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 15   | 14              | 13   | 12 | 11 | 10 | 09 | 08 | 07     | 06 | 05               | 04               | 03 | 02 | 01 | 00 |    |    |    |      |    |   |  |  |  |  |    |        |  |                  |                  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0  |                 |  |    |    |    |    |    |        |    |                  | 1D <sub>16</sub> |    |    |    |    |    |    |    |      |    |   |  |  |  |  |    |        |  |                  |                  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| BLCNE  | COUNT,LABEL     | Skip if counter value unequal to block length  |    |    |    |    |    |        |    |                  |                  |    |    |    |    |    |    |    |      |    |   |  |  |  |  |    |        |  |                  |                  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |                 | <table border="1"> <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="5">0</td><td colspan="3">A7</td><td colspan="6">1C<sub>16</sub></td> </tr> <tr> <td colspan="16">0</td> </tr> </table> <p>A1 = count -1                      A7 = label - *-2<br/>                     Macro takes the form BLC1NE or BLC2NE where A1 = 0 or 1</p> | 15 | 14 | 13 | 12 | 11 | 10     | 09 | 08               | 07               | 06 | 05 | 04 | 03 | 02 | 01 | 00 | A1   | 1  | 0 |  |  |  |  | A7 |        |  | 1C <sub>16</sub> |                  |  |  |  |  | 0  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 15   | 14              | 13   | 12 | 11 | 10 | 09 | 08 | 07     | 06 | 05               | 04               | 03 | 02 | 01 | 00 |    |    |    |      |    |   |  |  |  |  |    |        |  |                  |                  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| A1   | 1               | 0  |    |    |    |    | A7 |        |    | 1C <sub>16</sub> |                  |    |    |    |    |    |    |    |      |    |   |  |  |  |  |    |        |  |                  |                  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0  |                 |  |    |    |    |    |    |        |    |                  |                  |    |    |    |    |    |    |    |      |    |   |  |  |  |  |    |        |  |                  |                  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| BLDWL  | WC,WL,ACTION,EP | Build worklist entry with given workcode   |    |    |    |    |    |        |    |                  |                  |    |    |    |    |    |    |    |      |    |   |  |  |  |  |    |        |  |                  |                  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |                 | <table border="1"> <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="7">WC</td><td colspan="3">ACTION</td><td colspan="5">03<sub>16</sub></td> </tr> <tr> <td colspan="16">EP    WLCB ADDRESS</td> </tr> </table> <p>WL is ignored but is present in macro call</p>                                      | 15 | 14 | 13 | 12 | 11 | 10     | 09 | 08               | 07               | 06 | 05 | 04 | 03 | 02 | 01 | 00 | 0    | WC |   |  |  |  |  |    | ACTION |  |                  | 03 <sub>16</sub> |  |  |  |  | EP    WLCB ADDRESS |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 15   | 14              | 13   | 12 | 11 | 10 | 09 | 08 | 07     | 06 | 05               | 04               | 03 | 02 | 01 | 00 |    |    |    |      |    |   |  |  |  |  |    |        |  |                  |                  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0  | WC              |  |    |    |    |    |    | ACTION |    |                  | 03 <sub>16</sub> |    |    |    |    |    |    |    |      |    |   |  |  |  |  |    |        |  |                  |                  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| EP    WLCB ADDRESS |                 |  |    |    |    |    |    |        |    |                  |                  |    |    |    |    |    |    |    |      |    |   |  |  |  |  |    |        |  |                  |                  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |



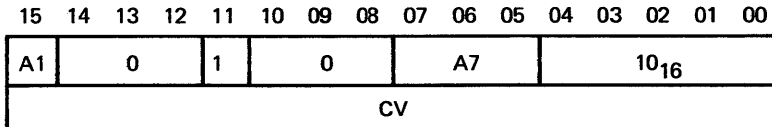
**MACRO                    PARAMETERS                    PARAMETER LIST FORMAT**

**CMPCLA                    CMASK,LABEL                    Compare CLA status**



A7 = label - \*-2

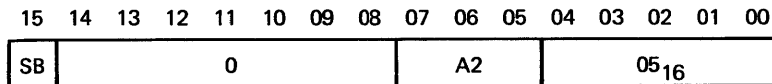
**CNTNE                    COUNT,CV,LABEL                    Skip if character counter does not equal CV**



A1 = count -1    A7 = label - \*-2

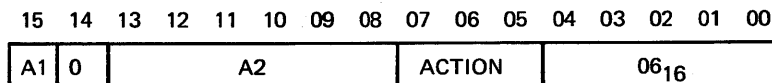
Macro also takes the form CNT1NE CV,LABEL and CNT2NE CV, LABEL where A1 = 0 or 1

**CRCEQ                    SB,LABEL                    Skip if CRC equal**



A2 = label - \*-1

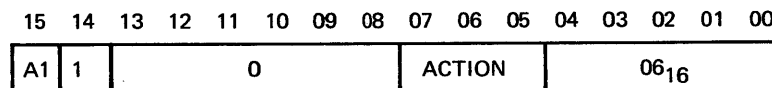
**DCC                    COUNT,LABEL,ACTION                    Decrement count**



A1 = count -1    A2 = label - \*-1

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

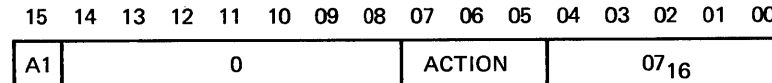
**ICC                    COUNT,ACTION                    Increment count**



A1 = count -1

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

**INTCC                    COUNT,ACTION                    Initialize count**



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

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

|    |   |   |        |                  |
|----|---|---|--------|------------------|
| A3 | 0 | 2 | ACTION | 1F <sub>16</sub> |
|----|---|---|--------|------------------|

A3 = ICRC

JUMP                      STATE,RTN                      Jump to state

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

|   |   |       |   |                  |
|---|---|-------|---|------------------|
| 0 | 1 | STATE | 0 | 08 <sub>16</sub> |
|---|---|-------|---|------------------|

JUMP                      STATE                      Update state index and jump

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

|   |   |       |   |                  |
|---|---|-------|---|------------------|
| 1 | 0 | STATE | 0 | 08 <sub>16</sub> |
|---|---|-------|---|------------------|

MJUMP                      STATE                      Set modem state and execute

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

|   |   |       |   |                  |
|---|---|-------|---|------------------|
| 1 | 0 | STATE | 0 | 19 <sub>16</sub> |
|---|---|-------|---|------------------|

MODCC                      COUNT,CV                      Set count with modulus function

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

|    |   |  |  |                  |
|----|---|--|--|------------------|
| A1 | 0 |  |  | 1C <sub>16</sub> |
| CV |   |  |  |                  |

A1 = count -1

MSTATE                      STATE,ACTION                      Set modem state index

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

|   |  |       |        |                  |
|---|--|-------|--------|------------------|
| 0 |  | STATE | ACTION | 19 <sub>16</sub> |
|---|--|-------|--------|------------------|

MSTLS                      STATE,LABEL                      Skip if modem state < operand

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

|   |       |  |    |                  |
|---|-------|--|----|------------------|
| 1 | STATE |  | A2 | 0B <sub>16</sub> |
|---|-------|--|----|------------------|

A2 = label - \*-1

NOPR                      ACTION                      No operation (execute ACTION only)

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

|   |  |  |        |                  |
|---|--|--|--------|------------------|
| 0 |  |  | ACTION | 00 <sub>16</sub> |
|---|--|--|--------|------------------|

MACRO

PARAMETERS

PARAMETER LIST FORMAT

RADDC

CHAR

Expand (add) current character

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

|      |   |                  |
|------|---|------------------|
| CHAR | 6 | 11 <sub>16</sub> |
|------|---|------------------|

RESYNC

ACTION

Resync the line

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

|   |   |        |                  |
|---|---|--------|------------------|
| 0 | 1 | ACTION | 1F <sub>16</sub> |
|---|---|--------|------------------|

RCHAR

CHAR,ACTION

Replace character

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

|      |        |                  |
|------|--------|------------------|
| CHAR | ACTION | 02 <sub>16</sub> |
|------|--------|------------------|

RPLACE

CHAR,CRCA

Replace and store character with CRC

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

|      |   |                  |
|------|---|------------------|
| CHAR | 0 | 02 <sub>16</sub> |
| 0    | 3 | 12 <sub>16</sub> |

RPLACE

CHAR

Replace and store character without CRC

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

|      |   |                  |
|------|---|------------------|
| CHAR | 0 | 02 <sub>16</sub> |
| 0    | 2 | 12 <sub>16</sub> |

RSTIME

TIME,ACTION

Reset timer

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

|      |        |                  |
|------|--------|------------------|
| TIME | ACTION | 1A <sub>16</sub> |
|------|--------|------------------|

RSTINP

ACTION

Reset input in progress flag

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

|   |        |                  |
|---|--------|------------------|
| 0 | ACTION | 1F <sub>16</sub> |
|---|--------|------------------|

RSTMXF

MFLAGS,ACTION

Reset user flags

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

|        |   |        |                  |
|--------|---|--------|------------------|
| 0      | 1 | ACTION | 17 <sub>16</sub> |
| 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 <sub>16</sub> |
|---|---|--------|------------------|

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 <sub>16</sub> |
|---|---|---|---|--------|------------------|

SKIP

LABEL

Skip forward

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

|    |   |                  |
|----|---|------------------|
| A9 | 1 | 00 <sub>16</sub> |
|----|---|------------------|

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 <sub>16</sub> |
|----|---|------------------|

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 <sub>16</sub> |
|----|--------|------------------|

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 <sub>16</sub> |
|---|-------|--------|------------------|

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 <sub>16</sub> |
|---|-------|----|------------------|

A2 = label - \*-1



MACRO                      PARAMETERS                      PARAMETER LIST FORMAT

STORC                      COUNT,ACTION                      Store count

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

|    |   |        |                  |
|----|---|--------|------------------|
| A1 | 0 | ACTION | 14 <sub>16</sub> |
|----|---|--------|------------------|

A1 = count -1  
Also STORC1 ACTION and STORC2 ACTION

STORE                      Store character without CRC

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

|   |   |                  |
|---|---|------------------|
| 0 | 2 | 12 <sub>16</sub> |
|---|---|------------------|

STORE                      CRCA                      Store character and accumulate CRC

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

|   |   |                  |
|---|---|------------------|
| 0 | 3 | 12 <sub>16</sub> |
|---|---|------------------|

STRNTB                      TA,ACTION                      Set translation table address

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

|    |        |                  |
|----|--------|------------------|
| 0  | ACTION | 1B <sub>16</sub> |
| TA |        |                  |

STRNTE                      ACTION,EP                      Set translation table address

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

|    |                           |                  |
|----|---------------------------|------------------|
| 0  | ACTION                    | 1B <sub>16</sub> |
| EP | TRANSLATION TABLE ADDRESS |                  |

TIBSWC                      WC,EOT,ACTION                      Terminate and save workcode

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

|   |    |    |        |                  |
|---|----|----|--------|------------------|
| 1 | A5 | WC | ACTION | 04 <sub>16</sub> |
| 0 |    |    |        |                  |

A5 = EOT

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

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

|    |              |    |        |                  |
|----|--------------|----|--------|------------------|
| 0  | A5           | WC | ACTION | 04 <sub>16</sub> |
| EP | WLCB ADDRESS |    |        |                  |

A5 = EOT

TPADDR                      SD,DD                      (SD) + (DD) → (DD)

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

|    |    |   |                  |
|----|----|---|------------------|
| SD | DD | 1 | 10 <sub>16</sub> |
|----|----|---|------------------|

MACRO                      PARAMETERS                      PARAMETER LIST FORMAT

TPBKUP                      LV,SRC,DST                      Restore text processing conditions

|  |    |    |    |    |    |    |    |    |    |    |    |                  |    |    |    |    |
|--|----|----|----|----|----|----|----|----|----|----|----|------------------|----|----|----|----|
|  | 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04               | 03 | 02 | 01 | 00 |
|  | 1  | 0  |    |    |    | A6 | A8 | 0  |    |    |    | 1E <sub>16</sub> |    |    |    |    |

A6 = LV-1                      A8 = SRC + DST

TPCMR                      SD,DD                      Compare file 1 registers

|  |    |    |    |    |    |    |    |    |    |    |    |    |                  |    |    |    |
|--|----|----|----|----|----|----|----|----|----|----|----|----|------------------|----|----|----|
|  | 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03               | 02 | 01 | 00 |
|  | SD |    |    |    | DD |    |    |    | 3  |    |    |    | 10 <sub>16</sub> |    |    |    |

TPDECR                      SD,VALUE                      Decrement file 1 register

|  |    |       |    |    |    |    |    |    |    |    |    |    |    |                  |    |    |  |
|--|----|-------|----|----|----|----|----|----|----|----|----|----|----|------------------|----|----|--|
|  | 15 | 14    | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02               | 01 | 00 |  |
|  | 1  | VALUE |    |    |    | SD |    |    |    | 0  |    |    |    | 10 <sub>16</sub> |    |    |  |

TPEXIT                      Exit from text processing

|  |    |    |    |    |    |    |    |    |    |    |    |    |                  |    |    |    |
|--|----|----|----|----|----|----|----|----|----|----|----|----|------------------|----|----|----|
|  | 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03               | 02 | 01 | 00 |
|  | 0  |    |    |    |    |    |    |    | 1  |    |    |    | 1E <sub>16</sub> |    |    |    |

TPINCR                      SD,VALUE                      Increment file 1 register

|  |    |       |    |    |    |    |    |    |    |    |    |    |    |                  |    |    |  |
|--|----|-------|----|----|----|----|----|----|----|----|----|----|----|------------------|----|----|--|
|  | 15 | 14    | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02               | 01 | 00 |  |
|  | 0  | VALUE |    |    |    | SD |    |    |    | 0  |    |    |    | 10 <sub>16</sub> |    |    |  |

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

|  |    |    |    |    |    |    |    |    |                  |    |    |    |    |    |    |    |
|--|----|----|----|----|----|----|----|----|------------------|----|----|----|----|----|----|----|
|  | 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07               | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
|  | L  | 0  | 0  | S  | 0  | 0  | 1  | 1  | 1F <sub>16</sub> |    |    |    |    |    |    |    |
|  | I  |    |    |    |    |    |    |    | CHAR             |    |    |    |    |    |    |    |

TPLD                      SD,DD                      Move 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 |    |    |    | 4  |    |    |    | 0E <sub>16</sub> |    |    |    |

TPLDL                      SD,DD                      Move left 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 |    |    |    | 6  |    |    |    | 0E <sub>16</sub> |    |    |    |

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 <sub>16</sub> |
|----|----|---|------------------|

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 <sub>16</sub> |
|---|----|---|---|------------------|

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 <sub>16</sub> |
|----|----|---|------------------|

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 <sub>16</sub> |
|---|----|---|------------------|

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 <sub>16</sub> |
|---|---|----|---|------------------|

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 <sub>16</sub> |
|----|----|---|------------------|

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 <sub>16</sub> |
|---|----|---|------------------|

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 <sub>16</sub> |
|---|----|---|------------------|

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 <sub>16</sub> |
|----|----|---|------------------|

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 <sub>16</sub> |
|---|---|----|---|------------------|

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 <sub>16</sub> |
|---|---|----|---|------------------|

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 <sub>16</sub> |
|----|----|---|------------------|

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 <sub>16</sub> |
|----|----|---|------------------|

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 <sub>16</sub> |
| 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 <sub>16</sub> |
| 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)<br>(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<br>7.1     | Add a character (including store)    |
| ALNBUF | 10.8           | Allocate a new buffer                |
| BKSPAC | 3.9            | Backspace (not over buffer boundary) |

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

| Macro  | Execution Time | Description   |
|--------|----------------|---|
| 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<br>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 flag                |
| 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 register 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**

---

**C**

(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 MXMRT0(2) RESPON 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($80) BIT 12
0040 EQU NCUOP5($40) BIT 11
0020 EQU NCUOP6($20) BIT 10
0010 EQU NCUOP7($10) BIT 9
0008 EQU NCUOP8($08) BIT 8
0004 EQU NCUOP9($04) BIT 7 (TEXT PROCESSING ONLY)
0002 EQU NCUOPA($02) BIT 6 (TEXT PROCESSING ONLY)
0001 EQU NCUOPB($01) BIT 5 (TEXT PROCESSING ONLY)

```

```

*****
*
*   WORK CODES
*
*****

```

```

0003 EQU MMBUTCH(3) MUX BUFFER THRESHOLD
0021 EQU A0MK1($21)
0022 EQU A0MK2(A0MK1+1)
0023 EQU A0MK3(A0MK2+1)
0024 EQU A0MK4(A0MK3+1)
0025 EQU A0MK5(A0MK4+1)
0026 EQU A0MK6(A0MK5+1)
0027 EQU A0MK7(A0MK6+1)
0028 EQU A0MK8(A0MK7+1)
0029 EQU A0MK9(A0MK8+1)
002A EQU A0MK10(A0MK9+1)
002B EQU A0MK11(A0MK10+1)
002C EQU A0MK12(A0MK11+1)
002D EQU A0MK13(A0MK12+1)
002E EQU A0MK14(A0MK13+1)
002F EQU A0MK15(A0MK14+1)
0030 EQU A0MK16(A0MK15+1)
0031 EQU A0MK17(A0MK16+1)
0032 EQU A0MK18(A0MK17+1)
0033 EQU A0MK19(A0MK18+1)
0034 EQU A0MK20(A0MK19+1)
0035 EQU A0MK21(A0MK20+1)
0036 EQU A0MK22(A0MK21+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      HWKLN0($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 WCRKCODE

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

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

003F      EQU      HMCMASK($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
*
*****
*****

```

| POINTER | MAC | NH               | ENT    | HINSPT    |   |
|---------|-----|------------------|--------|-----------|---|
|         | EQU | HS#NH#(*-HINSPT) | EQU    | HINSPT(*) |   |
|         | ADC | H#NH#            | HSINST | CLASTAT   | 0 |
|         | EMC |                  | HSINST | DCDNOT    | 1 |
|         |     |                  | HSINST | OVERUN    | 2 |
|         |     |                  | HSINST | BUTHR     | 3 |
|         |     |                  | HSINST | INIT      |   |
|         |     |                  | HSINST | DATO      |   |
|         |     |                  | HSINST | SOH       |   |
|         |     |                  | HSINST | OLED      |   |
|         |     |                  | HSINST | BCB       |   |
|         |     |                  | HSINST | LFCS      |   |
|         |     |                  | HSINST | RFCS      |   |
|         |     |                  | HSINST | 1RCB      |   |
|         |     |                  | HSINST | CONTROL   |   |
|         |     |                  | HSINST | SRCE      |   |
|         |     |                  | HSINST | SCB       |   |
|         |     |                  | HSINST | DATA      |   |
|         |     |                  | HSINST | DLE       |   |
|         |     |                  | HSINST | SIGNON    |   |
|         |     |                  | HSINST | ETB       |   |
|         |     |                  | HSINST | 1CRC      |   |
|         |     |                  | HSINST | 2CRC      |   |
|         |     |                  | HSINST | ERROR     |   |
|         |     |                  | HSINST | TERM      |   |
|         |     |                  | HSINST | IOLE      |   |

STANDARD DEFINITIONS FOR  
INPUT STATE PROGRAMS

```

*****
*
*          HSCLASTAT - CLA STATUS HANDLER
*
*****
P0018      0020      HCLASTAT  NOPR  EXIT          IGNORE STATUS
P0019
*
*          HSDCCDNOT - DATA-CARRIER-DETECT DROPPED
*
*****
P0019      0237      HSDCCDNOT  TSTMXF  MXCARR,HDCD1 *   SKIP IF      CONTROLLED CARRIER
P001A      0020
P001B      013F
P001C      0237      HDCD1       TSTMXF  MXETX,HDCD2 *   SKIP IF WORKLIST WANTED
P001D      0080
P001E      013F
P001F      0428      HDCD2       MSTLS  MSTIOL,HDCD3 DOUBLE CHECK THAT MODEM STATE IS IDLE
P0020      0528      MSTLS  MSTIOL+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
P0025      0003
P0026      0000
P0027      013F
P0028
*
*          RESYNC EXIT *   RESYNC CLA AND EXIT
*
*****
*
*          HSOVERUN - TOO MANY BUFFERS
*
*****
P0028      5508      HSOVERUN  JUMP  HSERRCR,RTN GOTO STATE EPPOR REMEMBER CUR STATE
P0029
*
*          HSBUTHR - BUFFER-THRESHOLD REACHED IN SYSTEM
*
*****
P0029      0304      HSBUTHR   TIBWL  HMBUTCH *   TELL MUX SS TO RELEASE BUFFERS
P002A      0000
P002B      A604
P002C      0000
P002D      9608
P002E
*
*          JUMP  HSTERM *   TERMINATE INPUT
*
*****
*
*          HSINIT - INITIAL INPUT STATE
*
*****
P002E      320C      HSINIT    CHARNE HCSYN,HINIT1 LOCK FOR SYN CHAR
P002F      0117      RSTMXF  HMXPT   RESET MUX XPT FLAG
P0030      0200
P0031      0117
P0032      0080
P0033      0619
P0034      0528
P0035      013F      HINIT1    RESYNC EXIT IT ISNT - RESYNC CLA
P0036
*
*          HSDAT0 - DATA ARRIVING
*
*****
P0036      322C      HSDAT0   CHARNE HCSYN,HDAT01 SYN CHAR
P0037      0020      NOPR  EXIT   YES - IGNORE
P0038      012C      HDAT01   CHARNE HCSOH,HDAT02 SYN
P0039      0628      STATE  HSSOH,EXIT YES
P003A      102C      HDAT02   CHARNE HCDLE,HDAT03 DLE
P003B      0728      STATE  HSCLE0,EXIT
P003C      306C      HDAT03   CHARNE HCNAK,HDAT05 NAK
P003D      A404      TIBSWC  HMKNAK *   YES- NAK WLE TO TIP
P003E      0000
P003F      9608
P0040      0408      HDAT05   JUMP  HSTERM *   TERMINATE INPUT
P0041
*
*          JUMP  HSINIT *   ALLOW LINE TO RESYNC
*
*****
*
*          HSSCH - SCH RECEIVED
*
*****
P0041      322C      HSSCH    CHARNE HCSYN,HSOH1 SYN
P0042      0020      NOPR  EXIT   YES - IGNORE
P0043      2D6C      HSOH1    CHARNE HCE0,HSOH2 ENG
P0044      A104      TIBSWC  HMKENG *   YES- ENG WLE TO TIP
P0045      0000
P0046      9608
P0047      024C      HSOH2    JUMP  HSTERM *   TERMINATE INPUT
P0048      021F      CHARNE  HCSTX,HSOH3 STX
P0049      0888      INTCRC  ZCRC *   INITIALIZE CRC ACCUM
P004A      0408      STATE  HSECB,CRCEXIT
*
*          JUMP  HSINIT *   ALLOW LINE TO RESYNC
*
*****

```

P0048

```

*****
*****
*
*      HSDLE0 - DLE RECEIVED
*
*****
P0048    322C    HOLE0    CHARNE HCSYN,HOLE01 SYN
P004C    0520    STATE HSDAT0,EXIT YES - IGNORE
P004D    786C    HOLE01    CHARNE HCKACK,HOLE02 ACK
P004E    A384    TIBSWC HMKACK * YES- ACK WLE TO TIP
P004F    0800
P0050    9608    JUMP HSTERN * TERMINATE INPUT
P0051    020C    HOLE02    CHARNE HCSTX,HOLE03 STX
P0052    0017    SETMXF HMXPT SET MUX XPT FLAG
P0053    0200
P0054    021F    INTCRC ZCRC * INITIALIZE CRC ACCUM
P0055    0028    STATE HSBCE,EXIT
P0056    0408    HOLE03    JUMP HSNIT * ALLOW LINE TO RESYNC
P0057

```

```

*****
*
*      HSBCE - PROCESS BCB
*
*****

```

0057 P

P0057 322C  
P0058 0020  
P0059 102C  
P005A 0020  
P005B 0011  
P005C 0968  
P005D

```

*****
*
*      HBCB EQU HBCB(*)
*
P0057    322C    HBCB EQU HBCB(*)
P0058    0020    CHARNE HCSYN,HBCB1
P0059    102C    NOPR EXIT IGNORE
P005A    0020    HBCB1 CHARNE HCDLE,HBCB2 DLE
P005B    0011    NOPR EXIT IGNORE
P005C    0968    HBCB2 ADDC HZZERO ADD DUMMY FOR RIGHT-CHAR-ALIGNMENT
P005D    STATE HSLFCS,CRCSTOREX STORE BCB,CRC AND EXIT

```

```

*****
*
*      HSLFCS - PROCESS LEFT FCS
*
*****

```

P005D 322C  
P005E 0020  
P005F 102C  
P0060 0020  
P0061 0237  
P0062 0200  
P0063 0220  
P0064 0513  
P0065 0A68  
P0066

```

*****
*
*      HLFCS CHARNE HCSYN,HLFCS1 SYN
*
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  
P0067 0020  
P0068 102C  
P0069 0020  
P006A 0868  
P006B

```

*****
*
*      HRFCS CHARNE HCSYN,HRFCS1 SYN
*
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 HSRFCS,CRCSTOREX STORE RFCS,CRC AND EXIT
P006B

```

```

*****
*
*      HSRFCS - PROCESS FIRST / NEXT RCB
*
*****

```

P006B 322C  
P006C 0020  
P006D 102C  
P006E 0020  
P006F 002C  
P0070 1288  
P0071 262C  
P0072 1388  
P0073 F02C  
P0074 0C88  
P0075 0D68  
P0076

```

*****
*
*      H1RCB CHARNE HCSYN,H1RCB1 SYN
*
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 HZERO,H1RCB5 NO (MORE) RECORDS
P0070    1288    STATE HSETB,CRCXIT DONE, LOOK FOR ETB
P0071    262C    H1RCB5 CHARNE HCEB,H1RCB3 ETB WITHOUT ZERO RCB
P0072    1388    STATE HSRFCS,CRCXIT YES GO PROCESS CFC NOW
P0073    F02C    H1RCB3 CHARNE HCCONTROL,H1RCB4 NO - CONTROL RECORD
P0074    0C88    STATE HSCONTROL,CRCXIT PROCESS CONTROL SRCB
P0075    0D68    H1RCB4 STATE HSSRCB,CRCSTOREX NO - GET SRCB
P0076

```

```

*****
*
*      HSCONTROL - CONTROL RCB RECEIVED,LOOK AT SRCB
*
*****

```

P0076 322C  
P0077 0020  
P0078 102C  
P0079 0020  
P007A C16C  
P007B AC1C  
P007C 0050  
P007D 1188  
P007E 0E68

```

*****
*
*      HCCNTROL CHARNE HCSYN,HCON1 SYN
*
P0076    322C    HCCNTROL CHARNE HCSYN,HCON1 SYN
P0077    0020    NCPP EXIT IGNORE
P0078    102C    HCON1 CHARNE HCDLE,HCON2 DLE
P0079    0020    NOPR EXIT IGNORE
P007A    C16C    HCON2 CHARNE HCSIGNON,HCON3 SIGNON
P007B    AC1C    SETCC2 HC40 YES - SET 80 CHAR LENGTH
P007C    0050
P007D    1188    HCON3 STATE HSSIGNON,CRCXIT PROCESS THE SIGNON + THROW AWAY SRCB
P007E    0E68    STATE HSSCB,CRCSTOREX NO - PROCESS NORMALLY

```

P007F

```

*****
*
*           HSSRCB - PROCESS SRCBS
*
*****

```

P007F 322C  
P0080 0020  
P0081 102C  
P0082 0020  
P0083 0E68  
P0084

```

+SRCB CHARNE HCSYN,HSRCB1 SYN
NOPR EXIT IGNORE
+SRCB1 CHARNE HCDLE,HSRCB2 DLE
NOPR EXIT IGNORE
+SRCB2 STATE HSSCB,CRSTOREX CRC STORE AND EXIT
*****

```

```

*
*           HSSCB - PROCESS SCBS
*
*****

```

P0084 322C  
P0085 0020  
P0086 102C  
P0087 0020  
P0088 262C  
P0089 1388  
P008A 002C  
P008B 0868  
P008C 066A  
P008D 901C  
P008E 003F  
P008F 0F68  
P0090 A06A  
P0091 A01C  
P0092 0001  
P0093 0F68  
P0094 0060  
P0095

```

+SCB CHARNE HCSYN,HSCB1 SYN
NOPR EXIT IGNORE
+SCB1 CHARNE HCDLE,HSCB1A DLE
NOPR EXIT IGNORE
+SCB1A CHARNE HCETB,HSCB2 ETB
STATE HS1CRC,CRCEXIT PROCESS CRC
+SCB2 CHARNE HZERO,HSCB3 EOR
STATE HS1RCB,CRSTOREX YES - GET NEXT RCB
+SCB3 CHARLS HNONCMP,HSCB4 NON - COMPRESSED
CHRCC2 HMNCSK SET COUNT TO NUM OF NON COMPRESSED
+SCB4 STATE HSDATA,CRSTOREX SET DATA STATE CRC, STORE AND EXIT
CHARLS HCMPNBLKS,HSCB5 COMPRESSED NON BLANK
SETCC2 HCOONE SET COUNT TO ONE
+SCB5 STATE HSDATA,CRSTOREX SET DATA STATE CPC, STORE AND EXIT
NOPR CRSTOREX COMPRESSED BLANKS - STORE SCB,CRC,EX
*****

```

```

*
*           HSDATA - PROCESS CHARACTERS AFTER SCB
*
*****

```

P0095 32AC  
P0096 0237  
P0097 0200  
P0098 0020  
P0099 0866  
P009A 0E68  
P009B 102C  
P009C 102A  
P009D 0400

```

+DATA CHARNE HCSYN,HDATA3 IS CHAR A SYN
TSTMXF HMXPT,HDATA1 YES - XPT WORKSTATION
NOPR EXIT NO - IGNORE
+DATA1 CCC2 HDATA2,CRSTOREX YES SO PROCESS IT
+DATA2 STATE HSSCB,CRSTOREX UNTIL DONE
+DATA3 CHARNE HCDLE,HDATA4 DLE
STATE HSDLE,EXIT YES - PROCESS IT
+DATA4 SKIPB HCATA1 NOT DLE - PROCESS CHARACTER
*****

```

```

*
*           HSDLE - PROCESS CHAR AFTER DLE
*
*****

```

P009E 322C  
P009F 0F28  
P00A0 0F08  
P00A1 0800

```

+DLE CHARNE HCSYN,HOLE1 SYN
STATE HSDATA,EXIT IGNORE
+DLE1 STATE HSDATA OTHERWISE SET STAE BACK TO DATA
AND PROCESS THIS CHARACTER
SKIPB HCATA1
*****

```

```

*
*           HSSIGNON - PROCESS SIGNON-CARD
*
*****

```

P00A2 262C  
P00A3 1388  
P00A4 0886  
P00A5 0E88  
P00A6

```

+SIGNON CHARNE HCETB,HSIGN2 * CHECK FOR EARLY ETB
STATE HS1CRC,CRCEXIT LOOK FOR CRC
+SIGN2 CCC2 HSIGN1,CRCEXIT ACCUM CRC, DISCARD DATA
+SIGN1 STATE HSSCB,CRCEXIT UNTIL DONE ALL 80
*****

```

```

*
*           HSETB - PROCESS ETB
*
*****

```

P00A6 322C  
P00A7 0020  
P00A8 102C  
P00A9 0020  
P00AA 262C  
P00AB 1388  
P00AC 5588  
P00AD

```

+ETB CHARNE HCSYN,HETB1 SYN
NOPR EXIT IGNORE
+ETB1 CHARNE HCDLE,HETB2 DLE
NOPR EXIT IGNORE
+ETB2 CHARNE HCETB,HETB3 ETB
STATE HS1CRC,CRCEXIT PROCESS
+ETB3 JUMP HSEERROR,RTN GOTO STATE ERROR REHFMFBR CUR STATE
*****

```

```

*
*           HS1CRC - PROCESS LEFT CRC
*
*****

```

P00AD 1488

```

+1CRC STATE HS2CRC,CRCEXIT SET FOR RIGHT CRC ,CRC AND EXIT
*****

```

P00AE

```

*****
*
*      HS2CRC - PROCESS RIGHT CRC
*
*****

```

P00AE 0025  
P00AF 5500  
P00B0 A504  
P00B1 0000  
P00B2 9608  
P00B3

```

*****
H2CRC  CRCEQ  BE,H2CRC1  CRC EQUAL
        JUMP  HSEFFCR,RTN  NO, ERROR
H2CRC1  TIBSMC  HMKMSG      * YES- WLE TO TIP
        JUMP  HSTERM      * TERMINATE INPUT
*****

```

```

*****
*
*      HSERROP - ERROR IN DATA MESSAGE
*
*****

```

P00B3 A204  
P00B4 0000  
P00B5 9608  
P00B6

```

*****
HERRCR  TIBSMC  HMKERR      * GIVE TIP AN ERROR WLE
        JUMP  HSTERM      * TERMINATE INPUT
*****

```

```

*****
*
*      HSTERM - TERMINATE INPUT
*
*****

```

P00B6 0419  
P00B7 02D7  
P00B9 0020  
P00B9 001A  
P00BA 0117  
P00BB 0000  
P00BC 8003  
P00BD 0000  
P00BE 9708  
P00BF 0017  
P00C0 0000  
P00C1 9708  
P00C2

```

*****
HTERM   MSTATE  MSTIDLE  * SET MODEM STATE TO IDLE
        TSTMXF  MXCARF,HTERM1  SKIP IF CONTROLLED CARRIER
*****

```

```

*****
RSTIME  0          * TURN OFF TIMER
RSTMXF  MXETX     * RESET ETX FLAG
*****

```

```

*****
BLOWL  ,,HWORK1  * MAKE WLE W/ SAVED WORKCODE
*****

```

```

*****
HTERM1  JUMP  HSIDLE  * WAIT AT IDLE
        SETMXF  MXETX  * SET ETX FLAG
*****

```

```

*****
HTERM1  JUMP  HSIDLE  * WAIT AT IDLE
*****

```

```

*****
*
*      HSIDLE - ALL DONE, IGNORE ANY ARRIVING DATA
*
*****

```

P00C2 013F

```

*****
HIDLE   RESYNC  EXIT      RESYNC CLA
*****

```

# INDEX

- ACTION 5-2
- ADDC 5-4
- ALNBUF 5-14
- Assignable counters 5-1
  
- Backspace 5-13
- BKSPAC 5-13
- BLCNE 5-3
- BLDWL 5-10
- BLK01 5-10
- Buffer allocation 5-13
- Buffer flags 5-9
  
- CHAR 5-2
- Character
  - add 5-4
  - block length 5-14
  - counters 5-1, 5-14
  - current 5-7
  - expand 5-4
  - repeat 5-4
  - replace 5-4
  - skips 5-6, 5-7
  - special 5-7
  - store 5-4, 5-12, 5-14
- CHARNE 5-6
- Circular input buffer (CIB) 2-1, 4-1
- CLA 5-7, 5-10
  - status bits 5-8
- CMPCLA 5-7
- CNTNE 5-3
- Command driver 2-2
- Communications line adapter (CLA) 5-7, 5-10
- Control
  - input state program 2-1
  - modem state program 4-1
  - text processing program 3-3
- COUNT 5-2
- Counters, character 5-1, 5-3, 5-4
- CRC 5-4, 5-6, 5-11, 5-13
- CRCA 5-2
- CRCEQ 5-6
- CV 5-2
  
- Data structures
  - input state program 2-2
  - text processing program 3-1
- DCC 5-3
- DD 5-2
  
- EOT 5-2
- EP 5-2
  
- Flags 5-7
- File 1 registers 3-2, 5-11, 5-12
  
- Index 5-5
  - input state 5-5
  - modem state 4-1, 5-5
  - text processing state 5-5
- Input data processor (IDP) 1-1, 2-1
- Input states 2-1, 4-2, 5-5, 5-6
- Instructions 1-1, 5-1
- INTCRC 5-14
  
- Interfaces
  - firmware to input states 2-1
  - firmware to modem states 4-1
  - firmware to text processing 3-2
  - input state to modem state 2-3, 4-2
  - input state to text processing 2-3
  - modem state to multiplex subsystem 4-2
  - program 1-1
  
- JUMP 5-5
- Jumps 5-5
  
- LABEL 5-2
  
- MJUMP 5-4
- MLCB 2-1, 2-2, 5-5, 5-10
- MODCC 5-1
- Modem states 2-3, 4-1, 5-4
- Move
  - field 5-13
  - file 1 register 5-11, 5-12
- MSTATE 5-4
- MSTLS 5-6
- Multiplex subsystem 1-1, 4-2
  
- No-Op 5-13
- NOPR 5-14
  
- OPS-level TIP 1-1, 2-3, 3-1, 3-2, 3-3, 5-4, 5-9, 5-10
- Output data processor (ODP) 1-1
  
- Parity flag 5-9
- Port table 2-1, 4-1, 5-5
- Process flag 5-7
- Programs
  - control 2-1, 3-3, 4-1
  - input state 2-1, 4-2
  - interfaces 1-1; also see Interfaces
  - modem 2-3, 4-1
  - organization 2-3, 3-3, 4-1
  - state 1-1
  - structure 1-1
  - text processing 3-1
- PTTPINF 3-2
  
- RADDC 5-4
- RCHAR 5-4
- RESYNC 5-13
- RPLACE 5-4
- RSTIME 5-13
- RSTINP 5-7
- RSTMXF 5-8
- RSTPAR 5-9
- RSTRAN 5-7
- RTRN 5-5
  
- SBLC 5-14
- SD 5-2
- SETFLG 5-9
- SETINP 5-7
- SETMXF 5-8
- SETPAR 5-9
- SETRAN 5-7
- SKIP, SKIPB 5-6



Skips 5-5, 5-6, 5-7

SPCHEQ 5-7

STATE 5-5

State

input 2-1, 4-2

modem 2-3, 4-1

pointer table 1-2, 2-1, 4-1

process 1-1, 2-1

program 1-1, 2-1, 4-1

reserved (0, 1, 2 and 3) 2-3, 4-2

text processing 3-1

Status, CLA 5-7

Status handler, multiplex 4-1

STORC, STORC1, STORC2 5-3

STORE 5-4

STRNTB 5-13

STRNTE 5-13

Synchronous line 5-13

Text processing 3-1, 5-11, 5-12

TIBSWC 5-9

TIBWL 5-9

Timer 5-12

TPADDR 5-11

TPBKUP 5-11

TPCB 3-1

TPDECR 5-11

TPEXIT 5-12

TPINCR 5-11

TPINSR 5-12

TPLD, TPLDL, TPLDR 5-14

TPMARK 5-11

TPMOVE 5-14

TPST, TPSTL, TPSTR 5-14

TPRSTL 5-12

TPRSTR 5-12

TPSTLC 5-12

TPSTRC 5-12

Translate flag 5-7

Translation table 5-12

TSTCLA 5-7

TSTMXF 5-8

User flags 5-8

VALUE 5-2

WC 5-2

WL 5-2

Worklists 5-10

# 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

AA3419 REV. 4/79 PRINTED IN U.S.A.

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