<table>
<thead>
<tr>
<th>Day 1</th>
<th>Day 2</th>
<th>Day 3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OVERVIEW</strong></td>
<td><strong>ADVANCED MCS USAGE</strong></td>
<td><strong>FNP INTERNALS</strong></td>
</tr>
<tr>
<td>Morning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>General Telecomm</td>
<td>tty_</td>
<td>Programs and</td>
</tr>
<tr>
<td>Concepts</td>
<td>TTF</td>
<td>Databases</td>
</tr>
<tr>
<td>MCS Parts</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FNP Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Afternoon</td>
<td><strong>RING 0 MCS INTERNALS</strong></td>
<td><strong>MCS METERING</strong></td>
</tr>
<tr>
<td></td>
<td>Programs and</td>
<td><strong>AND DEBUGGING</strong></td>
</tr>
<tr>
<td></td>
<td>Databases</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Multics-FNP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Dialogue</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>MCS ADMINISTRATION</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CMF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FNP Core Image</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operator Commands</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
MCS Overview

What is MCS

Multics Communications System

Software to transfer data to/from terminals or other devices connected via communications lines

Sometimes called MCM: Multics Communications Manager

Design is generalized, table-driven

Especially good at supporting diverse asynchronous terminals

In this course most of the major MCS topics are covered

Administration

CMF, TTF, FNP images

Use

All Ring 0 interfaces to MCS functions

Internals

Ring 0 and FNP

Metering and Debugging

Ring 0 and FNP

There are certain topics within MCS or related to MCS that will not be covered in detail

Details of various protocols

HASP, X.25, etc.

Video System

I/O Daemon software

FNPs other than 355/6670 families
Some communications concepts that are not necessarily specific to Multics need to be understood in order to understand MCS.

- **ASCII Character Set**
  - Multics was one of the first systems to use ASCII standard
  - **Character sizes**
    - 128 7-bit characters defined by standard
    - Usually transmitted as 8-bit characters
      - 7 data bits plus 1 parity bit
    - Stored in 9-bit bytes in Multics
      - 7 data bits plus 2 zero bits
7-bit Character

7 6 5 4 3 2 1

<table>
<thead>
<tr>
<th>D</th>
<th>D</th>
<th>D</th>
<th>D</th>
<th>D</th>
<th>D</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
</tr>
</tbody>
</table>

7-bit Character Plus Parity Bit

8 7 6 5 4 3 2 1

<table>
<thead>
<tr>
<th>P</th>
<th>D</th>
<th>D</th>
<th>D</th>
<th>D</th>
<th>D</th>
<th>D</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>r</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
</tr>
<tr>
<td>i</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

7-bit Character Stored in 9-bit Byte

9 8 7 6 5 4 3 2 1

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>D</th>
<th>D</th>
<th>D</th>
<th>D</th>
<th>D</th>
<th>D</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
</tr>
<tr>
<td></td>
<td></td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Control characters

Some ASCII codes are non-printing and have special uses

Multiple names are sometimes confusing

| X-OFF, DC3, Control-S, ^S, \023 |
| X-ON, DC1, Control-Q, ^Q, \021 |
ESC, Control-[, ^[, \033
LF, NL, Control-J, ^J, \012
CR, Control-M, ^M, \015
BS, Control-H, ^H, \010
Formfeed, NP, Control-L, ^L, \014
PAD, DEL, Rubout, \177
NUL, Control-@, ^@, Control-SP, ^SP, \000

BREAK (QUIT) is not an ASCII character

BREAK is a line condition discussed later under RS232
### ASCII Character Set Chart

<table>
<thead>
<tr>
<th>Bin</th>
<th>Oct</th>
<th>Dec</th>
<th>Hex</th>
<th>Def.</th>
<th>Key/Name</th>
<th>Other Definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000000</td>
<td>000</td>
<td>0</td>
<td>0</td>
<td>NUL</td>
<td>Control-@</td>
<td>Control-SP on some terminals</td>
</tr>
<tr>
<td>0000001</td>
<td>001</td>
<td>1</td>
<td>01</td>
<td>Control-A</td>
<td>SOH, Start of Header</td>
<td></td>
</tr>
<tr>
<td>0000010</td>
<td>002</td>
<td>2</td>
<td>02</td>
<td>Control-B</td>
<td>STX, Start of Text</td>
<td></td>
</tr>
<tr>
<td>0000011</td>
<td>003</td>
<td>3</td>
<td>03</td>
<td>Control-C</td>
<td>ETX, End of Text</td>
<td></td>
</tr>
<tr>
<td>0000100</td>
<td>004</td>
<td>4</td>
<td>04</td>
<td>Control-D</td>
<td>EOT, End of Transmission</td>
<td></td>
</tr>
<tr>
<td>0000101</td>
<td>005</td>
<td>5</td>
<td>05</td>
<td>Control-E</td>
<td>ENQ, Enquiry</td>
<td></td>
</tr>
<tr>
<td>0001100</td>
<td>006</td>
<td>6</td>
<td>06</td>
<td>Control-F</td>
<td>ACK, Acknowledge</td>
<td></td>
</tr>
<tr>
<td>0001111</td>
<td>007</td>
<td>7</td>
<td>07</td>
<td>BEL</td>
<td>Control-G Bell</td>
<td></td>
</tr>
<tr>
<td>0010000</td>
<td>010</td>
<td>8</td>
<td>08</td>
<td>BS</td>
<td>Control-H Backspace</td>
<td></td>
</tr>
<tr>
<td>0010100</td>
<td>011</td>
<td>9</td>
<td>09</td>
<td>HT</td>
<td>Control-I Horizontal Tab</td>
<td></td>
</tr>
<tr>
<td>0010110</td>
<td>012</td>
<td>10</td>
<td>0A</td>
<td>NL</td>
<td>Control-J Newline, Line Feed, LF</td>
<td></td>
</tr>
<tr>
<td>0010111</td>
<td>013</td>
<td>11</td>
<td>0B</td>
<td>VT</td>
<td>Control-K Vertical Tab</td>
<td></td>
</tr>
<tr>
<td>0011000</td>
<td>014</td>
<td>12</td>
<td>0C</td>
<td>NP</td>
<td>Control-L New Page, Form Feed, FF</td>
<td></td>
</tr>
<tr>
<td>0011010</td>
<td>015</td>
<td>13</td>
<td>0D</td>
<td>CR</td>
<td>Control-M Carriage Return</td>
<td></td>
</tr>
<tr>
<td>0011100</td>
<td>016</td>
<td>14</td>
<td>0E</td>
<td>RRS</td>
<td>Control-N Red Ribbon Shift, SO, Shift Out</td>
<td></td>
</tr>
<tr>
<td>0011111</td>
<td>017</td>
<td>15</td>
<td>0F</td>
<td>BRS</td>
<td>Control-O Black Ribbon Shift, SI, Shift In</td>
<td></td>
</tr>
<tr>
<td>0010000</td>
<td>020</td>
<td>16</td>
<td>10</td>
<td>Control-P</td>
<td>DLE, Data Link Escape</td>
<td></td>
</tr>
<tr>
<td>0010010</td>
<td>021</td>
<td>17</td>
<td>11</td>
<td>Control-Q</td>
<td>DC1, X-ON</td>
<td></td>
</tr>
<tr>
<td>0010101</td>
<td>022</td>
<td>18</td>
<td>12</td>
<td>Control-R</td>
<td>DC2</td>
<td></td>
</tr>
<tr>
<td>0010111</td>
<td>023</td>
<td>19</td>
<td>13</td>
<td>Control-S</td>
<td>DC3, X-OFF</td>
<td></td>
</tr>
<tr>
<td>0010100</td>
<td>024</td>
<td>20</td>
<td>14</td>
<td>Control-T</td>
<td>DC4</td>
<td></td>
</tr>
<tr>
<td>0010101</td>
<td>025</td>
<td>21</td>
<td>15</td>
<td>Control-U</td>
<td>NAK, Negative Acknowledge</td>
<td></td>
</tr>
<tr>
<td>0010110</td>
<td>026</td>
<td>22</td>
<td>16</td>
<td>Control-V</td>
<td>SYN, Synchronous Idle</td>
<td></td>
</tr>
<tr>
<td>0011001</td>
<td>027</td>
<td>23</td>
<td>17</td>
<td>Control-W</td>
<td>ETB, End of Transmission Block</td>
<td></td>
</tr>
<tr>
<td>0011000</td>
<td>030</td>
<td>24</td>
<td>18</td>
<td>Control-X</td>
<td>CAN, Cancel</td>
<td></td>
</tr>
<tr>
<td>0011001</td>
<td>031</td>
<td>25</td>
<td>19</td>
<td>Control-Y</td>
<td>EM, End of Medium</td>
<td></td>
</tr>
<tr>
<td>0011010</td>
<td>032</td>
<td>26</td>
<td>1A</td>
<td>Control-Z</td>
<td>SUB</td>
<td></td>
</tr>
<tr>
<td>0011011</td>
<td>033</td>
<td>27</td>
<td>1B</td>
<td>ESC, Control-[</td>
<td>Escape, Alt-mode</td>
<td></td>
</tr>
<tr>
<td>0011100</td>
<td>034</td>
<td>28</td>
<td>1C</td>
<td>Control-\</td>
<td>FS, File Separator</td>
<td></td>
</tr>
<tr>
<td>0011110</td>
<td>035</td>
<td>29</td>
<td>1D</td>
<td>Control-\</td>
<td>GS, Group Separator</td>
<td></td>
</tr>
<tr>
<td>0011111</td>
<td>036</td>
<td>30</td>
<td>1E</td>
<td>Control-\</td>
<td>RS, Record Separator</td>
<td></td>
</tr>
<tr>
<td>0011111</td>
<td>037</td>
<td>31</td>
<td>1F</td>
<td>Control-\</td>
<td>US, Unit Separator</td>
<td></td>
</tr>
<tr>
<td>0100000</td>
<td>040</td>
<td>32</td>
<td>20</td>
<td>SP</td>
<td>blank Space</td>
<td></td>
</tr>
<tr>
<td>0100001</td>
<td>041</td>
<td>33</td>
<td>21</td>
<td>!</td>
<td>exclamation point</td>
<td></td>
</tr>
<tr>
<td>0100010</td>
<td>042</td>
<td>34</td>
<td>22</td>
<td>&quot;</td>
<td>double quote</td>
<td></td>
</tr>
<tr>
<td>0100011</td>
<td>043</td>
<td>35</td>
<td>23</td>
<td>#</td>
<td>number sign</td>
<td></td>
</tr>
<tr>
<td>0100100</td>
<td>044</td>
<td>36</td>
<td>24</td>
<td>$</td>
<td>dollar sign</td>
<td></td>
</tr>
<tr>
<td>0100101</td>
<td>045</td>
<td>37</td>
<td>25</td>
<td>%</td>
<td>percent</td>
<td></td>
</tr>
<tr>
<td>0100110</td>
<td>046</td>
<td>38</td>
<td>26</td>
<td>&amp;</td>
<td>ampersand</td>
<td></td>
</tr>
<tr>
<td>0100111</td>
<td>047</td>
<td>39</td>
<td>27</td>
<td>’</td>
<td>acute accent</td>
<td></td>
</tr>
<tr>
<td>0101000</td>
<td>050</td>
<td>40</td>
<td>28</td>
<td>(</td>
<td>left parenthesis</td>
<td></td>
</tr>
<tr>
<td>0101010</td>
<td>052</td>
<td>42</td>
<td>2A</td>
<td>*</td>
<td>asterisk</td>
<td></td>
</tr>
<tr>
<td>0101011</td>
<td>053</td>
<td>43</td>
<td>2B</td>
<td>+</td>
<td>plus</td>
<td></td>
</tr>
<tr>
<td>0101100</td>
<td>054</td>
<td>44</td>
<td>2C</td>
<td>,</td>
<td>comma</td>
<td></td>
</tr>
<tr>
<td>0101111</td>
<td>055</td>
<td>45</td>
<td>2D</td>
<td>-</td>
<td>minus</td>
<td></td>
</tr>
<tr>
<td>0101110</td>
<td>056</td>
<td>46</td>
<td>2E</td>
<td>.</td>
<td>period</td>
<td></td>
</tr>
<tr>
<td>0101111</td>
<td>057</td>
<td>47</td>
<td>2F</td>
<td>/</td>
<td>right slash</td>
<td></td>
</tr>
</tbody>
</table>
0110000 060 48 30 0 0
0110001 061 49 31 1 1
0110010 062 50 32 2 2
0110011 063 51 33 3 3
0110100 064 52 34 4 4
0110101 065 53 35 5 5
0110110 066 54 36 6 6
0110111 067 55 37 7 7
0111000 070 56 38 8 8
0111001 071 57 39 9 9
0111010 072 58 3A : colon
0111011 073 59 3B ; semicolon
0111100 074 60 3C < less than
0111101 075 61 3D = equals
0111110 076 62 3E > greater than
0111111 077 63 3F ? question mark
1000000 100 64 40 @ commercial at
1000001 101 65 41 A Capital A
1000010 102 66 42 B Capital B
1000011 103 67 43 C Capital C
1000100 104 68 44 D Capital D
1000101 105 69 45 E Capital E
1000110 106 70 46 F Capital F
1000111 107 71 47 G Capital G
1001000 110 72 48 H Capital H
1001001 111 73 49 I Capital I
1001010 112 74 4A J Capital J
1001011 113 75 4B K Capital K
1001100 114 76 4C L Capital L
1001101 115 77 4D M Capital M
1001110 116 78 4E N Capital N
1001111 117 79 4F O Capital O
1010000 120 80 50 P Capital P
1010001 121 81 51 Q Capital Q
1010010 122 82 52 R Capital R
1010011 123 83 53 S Capital S
1010100 124 84 54 T Capital T
1010101 125 85 55 U Capital U
1010110 126 86 56 V Capital V
1010111 127 87 57 W Capital W
1011000 130 88 58 X Capital X
1011001 131 89 59 Y Capital Y
1011010 132 90 5A Z Capital Z
1011011 133 91 5B [ left bracket
1011100 134 92 5C \ back slash
1011101 135 93 5D ] right bracket
1011110 136 94 5E ` circumflex
1011111 137 95 5F _ underline
1100000 140 96 60 `\ grave accent
1100001 141 97 61 a small a
1100010 142 98 62 b small b
1100011 143 99 63 c small c
1100100 144 100 64 d small d
1100101 145 101 65 e small e
<table>
<thead>
<tr>
<th>ASCII Code</th>
<th>Decimal</th>
<th>Binary</th>
<th>Hexadecimal</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>1100110</td>
<td>146</td>
<td>102</td>
<td>66</td>
<td>f</td>
</tr>
<tr>
<td>1100111</td>
<td>147</td>
<td>103</td>
<td>67</td>
<td>g</td>
</tr>
<tr>
<td>1101000</td>
<td>150</td>
<td>104</td>
<td>68</td>
<td>h</td>
</tr>
<tr>
<td>1101001</td>
<td>151</td>
<td>105</td>
<td>69</td>
<td>i</td>
</tr>
<tr>
<td>1101010</td>
<td>152</td>
<td>106</td>
<td>6A</td>
<td>j</td>
</tr>
<tr>
<td>1101011</td>
<td>153</td>
<td>107</td>
<td>6B</td>
<td>k</td>
</tr>
<tr>
<td>1101100</td>
<td>154</td>
<td>108</td>
<td>6C</td>
<td>l</td>
</tr>
<tr>
<td>1101101</td>
<td>155</td>
<td>109</td>
<td>6D</td>
<td>m</td>
</tr>
<tr>
<td>1101110</td>
<td>156</td>
<td>110</td>
<td>6E</td>
<td>n</td>
</tr>
<tr>
<td>1101111</td>
<td>157</td>
<td>111</td>
<td>6F</td>
<td>o</td>
</tr>
<tr>
<td>1110000</td>
<td>160</td>
<td>112</td>
<td>70</td>
<td>p</td>
</tr>
<tr>
<td>1110001</td>
<td>161</td>
<td>113</td>
<td>71</td>
<td>q</td>
</tr>
<tr>
<td>1110010</td>
<td>162</td>
<td>114</td>
<td>72</td>
<td>r</td>
</tr>
<tr>
<td>1110011</td>
<td>163</td>
<td>115</td>
<td>73</td>
<td>s</td>
</tr>
<tr>
<td>1110100</td>
<td>164</td>
<td>116</td>
<td>74</td>
<td>t</td>
</tr>
<tr>
<td>1110101</td>
<td>165</td>
<td>117</td>
<td>75</td>
<td>u</td>
</tr>
<tr>
<td>1110110</td>
<td>166</td>
<td>118</td>
<td>76</td>
<td>v</td>
</tr>
<tr>
<td>1110111</td>
<td>167</td>
<td>119</td>
<td>77</td>
<td>w</td>
</tr>
<tr>
<td>1111000</td>
<td>170</td>
<td>120</td>
<td>78</td>
<td>x</td>
</tr>
<tr>
<td>1111001</td>
<td>171</td>
<td>121</td>
<td>79</td>
<td>y</td>
</tr>
<tr>
<td>1111010</td>
<td>172</td>
<td>122</td>
<td>7A</td>
<td>z</td>
</tr>
<tr>
<td>1111011</td>
<td>173</td>
<td>123</td>
<td>7B</td>
<td>(</td>
</tr>
<tr>
<td>1111100</td>
<td>174</td>
<td>124</td>
<td>7C</td>
<td></td>
</tr>
<tr>
<td>1111101</td>
<td>175</td>
<td>125</td>
<td>7D</td>
<td>)</td>
</tr>
<tr>
<td>1111110</td>
<td>176</td>
<td>126</td>
<td>7E</td>
<td>-</td>
</tr>
<tr>
<td>1111111</td>
<td>177</td>
<td>127</td>
<td>7F</td>
<td>PAD</td>
</tr>
</tbody>
</table>

**Standard Communications Concepts**
Baud Rate

Discrete signal events transmitted per second

Usually but not always equal to bits per second (bps)

New 2400 bps modems are 1200 baud

Line protocol

Parameters of a line that user can’t change

Parameters that user can change are called terminal protocol

A line can have several levels of protocols that:

Synchronise data transmission and reception

Ensure correct and complete reception of data by flow control and acknowledgment of received data.

Multiplex data for different devices on single line
Communications Protocols Supported by Multics

SERIAL BINARY

ASYNCHRONOUS

SYNCHRONOUS

KERMIT XMODEM BISYNC HDLC G115

3270 HASP 2780 3780 X.25 RCI VIP POLLED

/ Synchronisation of Clocks and Data /
/ Reliability of data (flow control & acknowledge) /
/ Multiplexing / Multiplexing

ASYNC -
SYNC - higher volume data
BISYNC - IBM standard
G115 - Honeywell BISync
Asynchronous Protocols

Data comes in bit by bit as changing signal on a wire with one voltage to represent logic 0 and another to represent logic 1.

Two things need to be synchronized between the sender and the receiver:

- When to sample each bit (synchronising clocks)
- Which bit is the first in data (first in each character)

Asynchronous protocols resynchronize both for each character:

- Start bit used to signal beginning of first bit of character
  - Transition from idle state (logic 1) to start bit (logic 0)
- One or two stop bits used to put line in idle state at end of each character
- 10 bits used to send 1 character, so BPS = 10 CPS, efficiency 70%
Asynchronous Protocol

Problem: Synchronisation of clocks and of data start

"6" = 066 octal = 0110110 binary = 00110110 with even parity

At 300 baud, 1/300 sends between bits.

Bits sent to send ASCII "6"

----> Message Flow ---->

<table>
<thead>
<tr>
<th>P</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Sampling needed every 1/300 second to read received bits.
Asynchronous Protocol

Bits sent to send ASCII "6"

----> Message Flow ---->

Idle | Stop | P | D6 | D5 | D4 | D3 | D2 | D1 | D0 | Start | Idle
0 0 1 1 0 1 1 0

---

See Transition
Asynchronous throughput

7 data bits 3 overhead bits

Maximum throughput: 70%
In Multics, maximum speed for asynchronous lines is 19,200 bps.

Protocol used by typical asynchronous terminals is called ASCII. Could be used by non-ASCII, e.g. EBCDIC, terminals.

Multics used to support other asynchronous protocols:
- 2741
- 1050
- ARDS

Standard asynchronous connections provide no guarantee of reliability.

The Kermit and XMODEM protocols sit on top of the asynchronous protocol.

They provide flow control, error detection, acknowledgement, for the purposes of reliable file transfer.

**Synchronous Protocols**

Again, need to synchronize clocks and data.

Clocks are synchronized by extra signal.

Data synchronization is per-block.

Two or more SYN characters are sent to indicate start of block.

Typical synchronous protocol adds information at beginning and end of each block:
- Block length, checksum, etc.

Typical synchronous protocol requires acknowledgement of each block from receiver.

Throughput depends on details of protocol, but is higher than asynchronous for medium to high volumes of data.
Typical Synchronous Protocol

| S | S | H...H | D | D.....D | D | T...T |   |
| Y | Y | D...D | A | A.....A | A | R...R |   |
| N | N | R...R | T | T.....T | T | A...A |   |

**Typical Values:**
- 2 Sync bytes
- 2 Header bytes
- 2 Trailer bytes
- 100 data bytes
- 8 Overhead bytes

**Throughput:** $\frac{100}{108} = 93\%$
In Multics, maximum speed for synchronous lines is 72 KB.

Multics supports a variety of synchronous protocols:

- BSC
- HASP
- 2780/3780
- 3270
- G115
- VIP
- Polled VIP
- HDLC
- X.25

Multiplexer

- There are many ways to have several logical connections over a single physical connection.
- This is known as a multiplexed line.
- For some types of multiplexed lines MCS can perform the work of multiplexing/demultiplexing the logical connections.
- The idea of multiplexing is generalized in MCS to include an arbitrary number of levels of multiplexing.
- MCS considers the FNP to be a multiplexer because it has many channels connected to it and multiplexes all of that information over a single physical connection (the DIA).

Modem

- MODulator/DEModulator
- Allows data to be transmitted over phone lines.
- Many different standards for modems.
- Modems may be full or half duplex.
- Not to be confused with echoplex.

RS232/V24
Defines connection between
Data Terminal Equipment (DTE): Terminal or computer
Data Circuit-terminating Equipment (DCE): Modems
Connection is by 25 pin connectors and cables
Diagram shows the pins used by Multics in asynchronous connections
Synchronous connection also uses pin 24 for timing (clock signal)
Used for all Multics protocols except HDLC

Multiplexers
Time-division
Frequency-division
Statistical-division

If you use a protocol Multics understand: x25, you don't need the 2nd Multiplexer.
### RS-232/V24

<table>
<thead>
<tr>
<th>DTE</th>
<th>RS232</th>
<th>DCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terminal</td>
<td></td>
<td>Modem</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Modem</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Computer</td>
</tr>
</tbody>
</table>

#### CCITT Circuit Pin Table

<table>
<thead>
<tr>
<th>Pin</th>
<th>No.</th>
<th>Name/Nom</th>
<th>Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>101</td>
<td>GND/TP</td>
<td>Ground</td>
<td>Terre de protection</td>
</tr>
<tr>
<td>7</td>
<td>102</td>
<td>GND/TS</td>
<td>Ground</td>
<td>Terre de signalisation</td>
</tr>
<tr>
<td>2</td>
<td>103</td>
<td>TX/ED</td>
<td>Transmission</td>
<td>Emission de donnees</td>
</tr>
<tr>
<td>3</td>
<td>104</td>
<td>RX/RD</td>
<td>Reception</td>
<td>Reception des donnees</td>
</tr>
<tr>
<td>5</td>
<td>105</td>
<td>RTS/DPE</td>
<td>Request to send</td>
<td>Demande pour emettre</td>
</tr>
<tr>
<td>6</td>
<td>106</td>
<td>CTS/PAE</td>
<td>Clear to send</td>
<td>Pret a emettre</td>
</tr>
<tr>
<td>8</td>
<td>107</td>
<td>DSR/PDP</td>
<td>Dataset ready</td>
<td>Poste de donne pret</td>
</tr>
<tr>
<td>10</td>
<td>109</td>
<td>CD/DP</td>
<td>Carrier Detect</td>
<td>Detection de la porteuse</td>
</tr>
<tr>
<td>20</td>
<td>108</td>
<td>DTR/CPD</td>
<td>Data terminal ready</td>
<td>Connectez le poste de donnees</td>
</tr>
</tbody>
</table>

#### Male Connector

```
1 2 3 4 5 6 7 8 9 10 11 12 13
14 15 16 17 18 19 20 21 22 23 24 25
```

#### Female Connector

```
13 12 11 10 9 8 7 6 5 4 3 2 1
25 24 23 22 21 20 19 18 17 16 15 14
```
Modem Dialup Sequence

<table>
<thead>
<tr>
<th></th>
<th>TTY DTE</th>
<th>Modem DCE</th>
<th>Modem DCE</th>
<th>FNP DTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>g</td>
<td>(TX) 2</td>
<td>---------</td>
<td>2 &lt;--------</td>
<td>(TX) f</td>
</tr>
<tr>
<td>f</td>
<td>(RX) 3</td>
<td>---------</td>
<td>3 --------</td>
<td>(RX) g</td>
</tr>
<tr>
<td>c</td>
<td>(RTS) 4</td>
<td>---------</td>
<td>4 &lt;--------</td>
<td>(RTS) b</td>
</tr>
<tr>
<td>e</td>
<td>(CTS) 5</td>
<td>---------</td>
<td>5 --------</td>
<td>(CTS) e</td>
</tr>
<tr>
<td>d</td>
<td>(DSR) 6</td>
<td>---------</td>
<td>6 --------</td>
<td>(DSR) a</td>
</tr>
<tr>
<td>e</td>
<td>(CD) 8</td>
<td>---------</td>
<td>8 --------</td>
<td>(CD) e</td>
</tr>
<tr>
<td>c</td>
<td>(DTR) 20</td>
<td>---------</td>
<td>20 --------</td>
<td>(DTR) b</td>
</tr>
</tbody>
</table>

To listen to a line, the FNP raises RTS (pin 4) and DTR (pin 20) and then waits for CTS (pin 5), DSR (pin 6) and CD (pin 8) to go high.

The FNP detects a hangup if DSR (pin 6) or CD (pin 8) drop for more than one second. If CTS (pin 5) drops, the FNP suspends output.

Scenario:

a) Modem on FNP is powered on. 
   *raises DSR*

b) FNP boots and listens to the line. 
   *raises RTS DSR*

c) The terminal is turned on. 
   *raises DSR ATR*

d) The terminal's modem is turned on. 
   *raises DSR*

e) The telephone call is made and the modems are connected. 
   *Raise (CTS CD)*

f) Multics sends the login banner.

g) User types login line.

*TX, RX only changes during connection*

*RTS = half duplex, transmit mode or receive mode*

*DSR = connection established*

*CTS = flow control, tied to DTR on terminal/pinout side.*
<table>
<thead>
<tr>
<th>Break/Interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Out-of-band signal</td>
</tr>
<tr>
<td>Logic 0 on pin 2 for 100 to 600 msec</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Line Driver</th>
</tr>
</thead>
<tbody>
<tr>
<td>Also known as short-haul modems or point-to-point modems</td>
</tr>
<tr>
<td>Inexpensive replacement for modems for point-to-point connections</td>
</tr>
<tr>
<td>No real standards</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Direct Connect</th>
</tr>
</thead>
<tbody>
<tr>
<td>For very short distances, a direct RS232 connection is possible</td>
</tr>
<tr>
<td>Very inexpensive replacement for modems</td>
</tr>
<tr>
<td>Theoretically limited to about 50'</td>
</tr>
</tbody>
</table>

| Practically can be extended much further |

| Known as hardwired connection, null modem, modem bypass |
| Each DTE must be made to think it is connected to a DCE |
6-WIRE DIRECT CONNECT

<table>
<thead>
<tr>
<th>TTY DTE</th>
<th>FNP DTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>(PG)</td>
<td>1</td>
</tr>
<tr>
<td>(SG)</td>
<td>7</td>
</tr>
<tr>
<td>(TX)</td>
<td>2</td>
</tr>
<tr>
<td>(RX)</td>
<td>3</td>
</tr>
<tr>
<td>(RTS)</td>
<td>4</td>
</tr>
<tr>
<td>(CTS)</td>
<td>5</td>
</tr>
<tr>
<td>(DSR)</td>
<td>6</td>
</tr>
<tr>
<td>(CD)</td>
<td>8</td>
</tr>
<tr>
<td>(DTR)</td>
<td>20</td>
</tr>
</tbody>
</table>

5-WIRE DIRECT CONNECT

<table>
<thead>
<tr>
<th>TTY DTE</th>
<th>FNP DTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>(PG)</td>
<td>1</td>
</tr>
<tr>
<td>(SG)</td>
<td>7</td>
</tr>
<tr>
<td>(TX)</td>
<td>2</td>
</tr>
<tr>
<td>(RX)</td>
<td>3</td>
</tr>
<tr>
<td>(RTS)</td>
<td>4</td>
</tr>
<tr>
<td>(CTS)</td>
<td>5</td>
</tr>
<tr>
<td>(DSR)</td>
<td>6</td>
</tr>
<tr>
<td>(CD)</td>
<td>8</td>
</tr>
<tr>
<td>(DTR)</td>
<td>20</td>
</tr>
</tbody>
</table>
Direct-Connect Dialup Sequence

TTY       FNP
DTE       DTE

d (TX) 2 --------/-------- 2 (TX) d

c (RX) 3 --------/-------- 3 (RX) c

b (RTS) 4 --| |-- 4 (RTS) a

b (CTS) 5 --| |-- 5 (CTS) a

a (DSR) 6 --|-- 6 (DSR) b

a (CD) 8 --------/-------- 8 (CD) b

b (DTR) 20 --------/-------- 20 (DTR) a

Scenario:
a) FNP boots and listens to the line.
b) The terminal is turned on.
c) Multics sends the login banner.
d) User types login line.

4-WIRE DIRECT CONNECT

TTY       FNP
DTE       DTE

(PG) 1 1

(SG) 7 ------------------ 7

(TX) 2 --------/-------- 2

(RX) 3 --------/-------- 3

(RTS) 4 --| |-- 4

(CTS) 5 --| |-- 5

(DSR) 6 --| |-- 6

(CD) 8 --| |-- 8

(DTR) 20 --------/ 20

Standard Communications Concepts
Line Monitor

Monitor RS232 signals and display transmitted and received data

Breakout Box

Monitor RS232 signals

Also known as a Blue Box.

Line Monitors, Blue Boxes

<table>
<thead>
<tr>
<th>DTE</th>
<th>RS232</th>
<th>DCE</th>
<th>Phone lines</th>
<th>DCE</th>
<th>RS232</th>
<th>DTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terminal</td>
<td>+ + +</td>
<td>Modem</td>
<td>Modem</td>
<td>+ + +</td>
<td>Computer</td>
<td></td>
</tr>
<tr>
<td>Blue Box/Line Mon.</td>
<td></td>
<td></td>
<td>Blue Box/Line Mon.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

Standard Communications Concepts

MCS Overview
Standard Multics Concepts

Ring 0 Supervisor

When a process requires supervisor services, it executes supervisor programs.

Three ways to enter supervisor code:

Call

Explicit request for supervisor services

For example, create a segment, make a segment known, write data to a terminal, read input from a terminal, etc.

Fault

Faults are caused by conditions within the CPU

Implicit request for supervisor services

For example, page faults, segment faults, linkage faults

Interrupt

Interrupts are caused by conditions outside the CPU

Perform a service for another process

For example, handle completion of a disk read, interrupt from FNP, etc.

Initializer/Answering Service

The Initializer process has a number of tasks to perform

Listen to login lines

Execute operator commands

Load FNPs, other multiplexers

Etc.

Block/Wakeup

Mechanism used to wait for events of unknown duration

Terminal I/O, tape I/O

Process is in user ring while blocked
When event occurs, some other process sends a wakeup to blocked process

Compare with Wait/Notify, to wait for events of short duration

Disk I/O, system locks

Process is in ring 0 while waiting

When event occurs, some other process notifies waiting process by changing its Traffic Control state
MCS Parts

<table>
<thead>
<tr>
<th>FNP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ring 0</td>
</tr>
<tr>
<td>Ring 4</td>
</tr>
</tbody>
</table>
MCS Parts

I/O Modules, e.g. tty_
Block/Wakeup
PL/I: >sll, >sss

Answering Service
FNP loading, dumping
PL/I: >tools

Data conversion, translation
Manage wired buffer space TTY-BUF
Communicate with FNP
Handle FNP interrupts
De-multiplexing (on other systems done on FNP, it should be PL/I & ALM: Hardcore (MST)

Transfer of data over DIA
Buffer space management
Echoing
Padding sending nulls for delays, done in Ring 0 on Multics
Flow control for Async I/O and xoff
Tell CS state of channels
Send terminal input to CS
Send output from CS to terminal
Modem control (RS-232, e.g.)
Interval timers
Format, interpret synchronous msgs
MAP355: loaded from Multics into FNP
Assembly language
The Ring 0 and FNP parts of Mes are covered in this course.

The MCS functions made available to the user by Ring 0 and the FNP will be explained using the tty_ I/O module as an example.

Following diagram shows the relationships of the various I/O modules for terminal I/O:

tty_ uses the same hcs_ interface as other I/O modules, and therefore makes a good example.

I/O Module Dependencies

```
+----- hasp_workstation_ <----------------+ hcs_<-----
|                                          |
+----- hasp_host_ <---------------------+           |
|                                          |
+----- ibm3270_                           |
|                                          |
+----- ibm2780_/3780_ <-------remote_punch_|
|                                          |
+----- bisync_ <----remote_teleprinter_+---|
|                                          |
+----- g115_<-------------------------------+ |
|                                          |
+----- tty_<-----------------------------+ |
|                                          |
+----- tc_io_<-----kermit                  |
|                                          |
+----- emacs_<-----xmodem_io_<--------------|
```

Mes Parts

MCS Overview
a FNP Hardware Description

| Up to 8 FNPs can be configured |

| Names |
| FNP |
| Front-End Network Processor |
| Front End |
| Fuh-Nup |
| Datanet |
| DN355 |
| 355 |
| 6670 |
| 6678 |
| 18x |
| FEP |
| MFX |

| Model Numbers |
| 355 |
| The original Multics FNP |
| Has given its name to many of the MCS programs |
| Limited to 32K memory |
| No longer supported as of MR11 |

| 6632 |
| A newer version of the 355 |
| Limited to 32K memory |
| No longer supported as of MR11 |

| 6670 |
| Level-6 based, emulates 355 |
Up to 256K memory with extended memory addressing

Two models

- 6678 with cache memory
- 6651 without cache

Architecture

- Has same basic architecture as Multics: IOM, SCU+MEM, CPU
- Peripherals connected to IOM
- DIA
- HSLAs

FNP Architecture

FNP IOM Channels:

0 FNP Console
1 FNP Reader
2 FNP printer
3
4 DIA
5
6 HSLA 0
7 HSLA 1
8 HSLA 2
9-14 LSLAs
15 Clock
Data and Instruction Formats

- 18-bit words
- I/O commands also know about 36-bit words
### STORE REFERENCE INSTRUCTION FORMAT

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>8</td>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>I</td>
<td>TAG</td>
<td>Opcode</td>
<td>Delta</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>6</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### INDEX REGISTER (Xn) -- UNPAGED ADDRESS

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>TAG</td>
<td>Address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### INDEX REGISTER (Xn) -- PAGED ADDRESS

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>3</td>
<td>9</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>TAG</td>
<td>Page No.</td>
<td>Offset</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
ACCUMULATOR REGISTER (A)

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>7</td>
</tr>
<tr>
<td>A</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
</tr>
</tbody>
</table>

QUOTIENT REGISTER (Q)

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>7</td>
</tr>
<tr>
<td>Q</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
</tr>
</tbody>
</table>

ACCUMULATOR-QUOTIENT REGISTER (AQ)

<table>
<thead>
<tr>
<th>0</th>
<th>1 1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>7 8</td>
</tr>
<tr>
<td></td>
<td>5</td>
</tr>
<tr>
<td>Even Word (A)</td>
<td>Odd Word (Q)</td>
</tr>
<tr>
<td>18</td>
<td>18</td>
</tr>
</tbody>
</table>
Extended Memory Addressing

So-called paging is used to address beyond 32K

Difficult to program, so only certain types of data are stored in extended memory

No programs in extended memory
FNP Extended Memory Addressing

Memory

+----------+ 000000-000377
|           | Page 0
+----------+
+----------+ 000400-000777
|           | Page 1
+----------+
+----------+ 001000-001377
|           | Page 2
+----------+

Page Table

4000 | A=0+
+-----+
4001 | A=0+
+-----+
4002 | A=0+
+-----+
    | .    
    | .    
    | .    
4175 | A=0+
+-----+
4176 | A=1+
+-----+
4177 | A=1+
+-----+

+----------+ 076400-076777
|           | Page 175
+----------+
+----------+ 077000-077377
|           | Page 176
+----------+
+----------+ 077400-077777
|           | Page 177, end of 32K
+----------+
+----------+ 100000-100377
|           | Page 200, start extended mem
+----------+
+----------+ 100400-100777
|           | Page 201
+----------+
+----------+ 177000-177377
|           | Page 376
+----------+
+----------+ 177400-177777
|           | Page 377, end of 64K
+----------+

FNP Hardware Description

MCS Overview
FNP Address Calculation Example 1
Reference To Address In Low-Order Memory Using Non-Paged Addressing

\[ c(x2) = 076700; \quad c(x3) = 077240; \quad c(475) = 004000; \quad c(4175) = 000000; \quad c(4176) = 133040 \]

Instruction: lda 1,2

<table>
<thead>
<tr>
<th>I</th>
<th>Tag</th>
<th>Opcode</th>
<th>Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10</td>
<td>000111</td>
<td>000000001</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>07</td>
<td>001</td>
</tr>
</tbody>
</table>

Final Address

| 0 | 7 | 6 | 7 | 0 | 1 |

X-register 2

<table>
<thead>
<tr>
<th>Tag</th>
<th>15-bit address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>76700</td>
</tr>
</tbody>
</table>
FNP Address Calculation Example 1
Reference To Address In Low-Order Memory Using Paged Addressing

c(x2)=076700; c(x3)=077240; c(475)=004000; c(4175)=000000; c(4176)=133040

++-----------------------------------+
| Instruction: lda 1, 2              |
++-----------------------------------+
| 2 | 0 | 7 | 0 | 0 | 1 |
| 010 | 000 | 111 | 000 | 000 | 001 |
++-----------------------------------+
| Tag | Opcode | Delta |
| 0 | 10 | 000111 | 000000001 |
| 0 | 2 | 07 | 001 |
++-----------------------------------+
|         |         |
| \    / | \    / |
| V     V |
++-----------------------------------+
| Final Address |
| 0 | 7 | 6 | 7 | 0 | 1 |
++-----------------------------------+
| X-register 2 |
++-----------------------------------+
| 0 | 7 | 6 | 7 | 0 | 0 |
| 000 | 111 | 110 | 111 | 000 | 000 |
++-----------------------------------+
| Tag | 15-bit address |
| 0 | 76700 |
++-----------------------------------+
| Tag | Page No. | Offset |
| 000 | 111 110 1 | 11 000 000 |
| 0 | 175 | 300 |
++-----------------------------------+
| \    / |
| V     V |
++-----------------------------------+
| Page table entry @ 4175 |
| [c(475) + 175 -> 4176] |
++-----------------------------------+
| 0 | 0 | 0 | 0 | 0 | 0 |
| 000 | 000 | 000 | 000 | 000 | 000 |
++-----------------------------------+
| Page addr mod 256| | Act |
| 000 000 000 0 | 00 | 0 | 00000 |
++-----------------------------------+
FNP Address Calculation Example 3
Reference To Address In Extended Memory

c (x2)=076700; c(x3)=077240; c(475)=004000; c(4175)=000000; c(4176)=133040

--- Instruction: lda 1,3 ---

<table>
<thead>
<tr>
<th>3</th>
<th>0</th>
<th>7</th>
<th>0</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>011</td>
<td>000</td>
<td>111</td>
<td>000</td>
<td>000</td>
<td>001</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tag</th>
<th>Opcode</th>
<th>Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>11</td>
<td>000111</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>07</td>
</tr>
</tbody>
</table>

--- Final Address ---

| 1 | 3 | 3 | 2 | 4 | 1 |

--- X-register 3 ---

<table>
<thead>
<tr>
<th>0</th>
<th>7</th>
<th>7</th>
<th>2</th>
<th>4</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>111</td>
<td>111</td>
<td>010</td>
<td>100</td>
<td>000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tag</th>
<th>15-bit address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>77240</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tag</th>
<th>Page No.</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>111 111 0</td>
<td>10 100 000</td>
</tr>
<tr>
<td>0</td>
<td>176</td>
<td>240</td>
</tr>
</tbody>
</table>

--- Page table entry @ 4176 [c(475) + 176 -> 4176] ---

<table>
<thead>
<tr>
<th>1</th>
<th>3</th>
<th>3</th>
<th>0</th>
<th>4</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>011</td>
<td>011</td>
<td>000</td>
<td>100</td>
<td>000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Page addr mod 256</th>
<th>Act</th>
</tr>
</thead>
<tbody>
<tr>
<td>001 011 011 0</td>
<td>00</td>
</tr>
</tbody>
</table>

--- FNP Hardware Description ---

MCS Overview
HSLA

High-Speed Line Adapter

Emulated in 6670s

All communications lines are connected to HSLAs

Maximum of 32 channels per HSLA

Maximum of 3 HSLAs per FNP -> maximum of 96 channels per FNP

HSLAs do much of the work of running a channel, freeing up the FNP's CPU

Set and detect RS232 signals

Interrupt FNP when a signal changes

Read incoming characters into a buffer

Interrupt FNP when action is required

Buffer full

BREAK condition

Interesting character (such as CR, LF, EOT) read

Character Control Table (CCT) tells HSLA what characters are interesting

Does not do echoing

Send buffer of characters to terminal

Interrupt FNP when finished

Old FNPs had HSLAs and LSLAs (Low-Speed Line Adapters)

Channel names

One component for each level of multiplexing

FNP. H/L 99

b.h101

FNP . H/L 99 . SUB

a.h006.prtl

Mother/Daughter Boards
HSLAs are emulated by Mother and Daughter Boards

Communications lines are connected to daughter boards

Two lines are connected to each daughter board

There are different types of daughter boards for different types of lines

Asynchronous

Autocall channels

BSC

G115

HDL C \textit{connections} only 1 channel per board

HDL C daughter boards only have one line connected instead of two

Daughter boards are properly known as CLAs

Communications Line Adapter

Daughter boards are mounted on mother boards

Four daughter boards can be mounted on one mother board

Maximum combined speed of lines on mother board is 72KB

An FNP can have up to 16 mother boards

12 mother boards 4 daughter boards 2 lines \Rightarrow 96 lines per FNP

4 mother boards are the equivalent of an HSLA (32 lines)

Mother boards are properly known as HMLCs

High-speed multi-line controller

Highest priority on motherboard is possibly folklore, not conclusive.
Mother/Daughter Boards

- HMLC = Mother Board
- CLA = Daughter Board

<table>
<thead>
<tr>
<th>CLA</th>
<th>CLA</th>
<th>CLA</th>
<th>CLA</th>
</tr>
</thead>
<tbody>
<tr>
<td>a.h231 a.h230</td>
<td>a.h229 a.h228</td>
<td>a.h227 a.h226</td>
<td>a.h225 a.h224</td>
</tr>
</tbody>
</table>

- HMLC

<table>
<thead>
<tr>
<th>CLA</th>
<th>CLA</th>
<th>CLA</th>
<th>CLA</th>
</tr>
</thead>
<tbody>
<tr>
<td>a.h107 a.h106</td>
<td>a.h105 a.h104</td>
<td>a.h103 a.h102</td>
<td>a.h101 a.h100</td>
</tr>
</tbody>
</table>

- HMLC

<table>
<thead>
<tr>
<th>CLA</th>
<th>CLA</th>
<th>CLA</th>
<th>CLA</th>
</tr>
</thead>
<tbody>
<tr>
<td>a.h031 a.h030</td>
<td>a.h029 a.h028</td>
<td>a.h027 a.h026</td>
<td>a.h025 a.h024</td>
</tr>
</tbody>
</table>

- HMLC

<table>
<thead>
<tr>
<th>CLA</th>
<th>CLA</th>
<th>CLA</th>
<th>CLA</th>
</tr>
</thead>
<tbody>
<tr>
<td>a.h023 a.h022</td>
<td>a.h021 a.h020</td>
<td>a.h019 a.h018</td>
<td>a.h017 a.h016</td>
</tr>
</tbody>
</table>

- HMLC

<table>
<thead>
<tr>
<th>CLA</th>
<th>CLA</th>
<th>CLA</th>
<th>CLA</th>
</tr>
</thead>
<tbody>
<tr>
<td>a.h015 a.h014</td>
<td>a.h013 a.h012</td>
<td>a.h011 a.h010</td>
<td>a.h009 a.h008</td>
</tr>
</tbody>
</table>

- HMLC

<table>
<thead>
<tr>
<th>CLA</th>
<th>CLA</th>
<th>CLA</th>
<th>CLA</th>
</tr>
</thead>
<tbody>
<tr>
<td>a.h007 a.h006</td>
<td>a.h005 a.h004</td>
<td>a.h003 a.h002</td>
<td>a.h001 a.h000</td>
</tr>
</tbody>
</table>

- HMLC

FNP Hardware Description

MCS Overview
Direct Interface Adapter
Connects Multics IOM to FNP IOM
200KBit/sec smart controller

DIA can:
    Send interrupts in both directions
    Transfer data between memories of FNP and Multics at request of either

MCS does not use all these possibilities, i.e. FNP initiates all data transfers

Can have 2 DIAs per FNP
    But Multics will view each DIA as a separate FNP
MCS Manuals

| AN85: Communication System SDN
| MCS Internals, both Ring 0 and FNP
| Theoretically unavailable: last update MR7.0
| CC75: Multics Administrators' Manual--Communications
| AG93: Multics Subroutines and Input/Output Modules
| AG91: Multics Programmer's Reference Guide
  | TTF described in Appendix B
  | Input conversion rules explained in Chapter 3
| AN87: Hardware and Software Formats PLM
  | Theoretically obsolete and unavailable
  | Chapter 6 has FNP data formats
| DD01: DN355/6600 Macro Assembler Program

FNS/1
CC92.
CMF/CDT

Channel Definition Table
CDT describes all configured FNPs, multiplexers, lines
Created from ASCII source file Channel Master File
An Answering Service Database
Stored in >scl>cdt
Used by Initializer to manage lines, logins, etc.
Used to initialize Ring 0 databases at Multics bootload
Ring 0 Databases are not stored permanently
Used to initialize FNP databases at FNP bootload
Spare channel count: 10;
FNP required up time: 5;

Sample CMF

Check_acs: all;
FNP: a;
type: DN6670;
lsa: 0; no lslas, observe to keep it &
hsa: 3;
memory: 64;
image: >sldd>mcs>info>fnp_a;
service: active;

name: a.c000; comment: "COLTS executive channel";
baud: 9600; line_type: COLTS; terminal_type: none;
service: slave;

name: a.h000; comment: "console pupitre";
baud: 1200; line_type: ASCII; terminal_type: ROSY;
service: mc;
attributes: hardwired,dont_read_answerback;

name: a.h001; comment: "console datanet";
baud: 1200; line_type: ASCII; terminal_type: ROSY;
service: login;
attributes: hardwired,dont_read_answerback;

name: a.h002; comment: "AJ510 dans salle de consoles 125";
baud: 1200; line_type: ASCII; terminal_type: AJ510;
service: login;
attributes: hardwired,dont_read_answerback;

name: a.h003; comment: "AJ860 dans salle de consoles 121";
baud: 1200; line_type: ASCII; terminal_type: AJ860;
service: login;
attributes: hardwired,dont_read_answerback;

/** a.h006 is X.25 Sync HDLC daughter. This board steals a.h007 as well. */

name: a.h006;
comment: "X.25 canal principal 113001300";
service: multiplexer;
multiplexer_type: x25;
baud: 9600;
terminal_type: X25_TRANSPAC;
line_type: X25LAP;

name: a.h006.d01-a.h006.d02;
service: autocall;
generic_destination: "transpac";
comment: "X.25 dial_out sub-channel";

name: a.h006.001-a.h006.013;
service: login;
comment: "X.25 login sub-channel";
terminal_type: ascii_crt;
baud: 1200;
attributes: dont_read_answerback;

name: a.hOOS; comment: "connexion stations HASP";
baud: 4800; line_type: BSC; terminal_type: HASP_HOST;
multiplexer_type: hasp;
service: multiplexer;

name: a.hOOS.opr;
  service: slave;
  line_type: BSC;
  terminal_type: HASP_HOST;

name: a.hOOS.rdr1;
  service: slave;
  line_type: BSC;
  terminal_type: HASP_HOST;

name: a.hOOS.prl1;
  service: slave;
  line_type: BSC;
  terminal_type: HASP_HOST;

name: a.hOOS.pun1;
  service: slave;
  line_type: BSC;
  terminal_type: HASP_HOST;

name: a.hOlO; comment: "Questar dans le bureau Adjemian-Weber";
baud: 4800; line_type: ASCII; terminal_type: VIP7205;
service: login; attributes: hardwired,dont_read_answerback;
CMF delivered in >udd>sa>a

FNP_required_up_time:

  | Global keyword
  | 2 crashes in this time => no reload
  | Applies to lower-level multiplexers as well

Spare_channel_count:

  | Global keyword
  | Number of extra entries in ring 0 databases
  | Adding CDT entries for which there is no room in ring 0 databases causes problems

Other global keywords

  | Define default values for omitted local keywords

FNP:

  | One FNP statement for each configured FNP
  | Followed by information about the FNP
  | type:
    | dn6670  Generic for All
  | memory:
    | Memory size in Kwords
  | hsla:
    | Number of HSLAs: can always say 3
    | Must have one line declared on HSLA 0 before using HSLA 1
    | Likewise for HSLA 1 and 2
  | image:
    | Pathname of image to load in FNP
  | service:
    | active or inactive
name:

One name statement for each configured channel
Followed by information about the channel
Can have a range of channel names, e.g. a.h006.001-a.h006.016

comment:

Comment stored in CDT
Not like /* comments */ which are ignored in CMF
Important to use comments to document
Also important to have well-organized CMF

baud:

Up to 19.2KB for async, 72KB for sync
110, 133, 140, 300, 600, 1200, 1800, 2400, 4800, 7200, 9600, 19200, 40800, 50000, or 72000

line_type:

<table>
<thead>
<tr>
<th>Asyn</th>
<th>B:Syn</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line protocol: ASCII, G115, BSC, VIP, POLLED_VIP, or X25LAP</td>
<td></td>
</tr>
<tr>
<td>Cannot be changed by user</td>
<td></td>
</tr>
</tbody>
</table>

service:

login

| Loaned to process for duration of process |
| Initializer owns all lines |

slave

| Loaned to existing process, returned when finished or at end of process |
| Requested by sending wakeup to Initializer (using dial_manager_) |
| Example: Used by I/O daemons to attach printers, readers, etc. |
| You can: io attach ?? ?? h.h !|
| Like slave, but for the Initializer’s own use |
Used for operator terminals

autocall

Like slave, but Initializer makes requested phone or network connection before loaning it

Example: dial_out, dial_out

inactive

Not listened to when FNP boots

Useful for holding a spot for a line that is not yet physically installed

multiplexer

Line is a multiplexer channel

multiplexer_type statement must be used as well

multiplexer_type:

ibm3270, vip7760, hasp, x25, or sty

dataset_type:

Used mostly for half-duplex modems

Require special handling of RTS, CTS, etc.

terminal_type:

Can be changed dynamically by user

For multiplexer, TTF entry gives multiplexer-specific parameters in the additional_info field

generic_destination:

Valid for autocall and slave lines

Allows users to attach line without knowing channel names

Useful to group together channels, allow changing channel assignments without affecting users

charge:

Specifies a surcharge for using the channel

For login lines this is in addition to connect charges
Must correspond to device type in installation parms

check_acs:

ACS Segments in >scl>rcp

E.g. >scl>rcp>a.h001.acs

Keywords specify when access checking is to be done

login
slave
priv_attach
dial_in
dial_out
all
Default is priv_attach, dial_out

attributes:

hardwired
Eliminates use of the login_time parameter in installation_parms

set_modes
Modes are set according to default terminal type at dialup
Only attribute that is on by default

dont_read_answerback
System does not send ^E (ENQ) at dialup to request terminal's answerback

check_answerback
See answerback statement

audit
See access_class statement

none
^audit,^check_answerback,^dont_read_answerback,^hardwired
Default is
set_modes, ^audit, ^check_answerback, ^dont_read_answerback, ^hardwired

| answerback: |
| Specifies expected answerback for terminal |
| If check_answerback attribute is set, any other answerback will be refused connection |

| access_class: |
| Specifies the AIM classes of users allowed to login on the line |
| Enforced if the audit attribute is set |

| initial_command: |
| Preaccess command (e.g. login, modes, ttp, etc.) to be executed at each dialup |

| cv_cmf |
| Converts CMF into CDT |
| Usually CMF.cmf -> CMF.cdt |
| Resulting segment has same format as >scl>cdt but with no dynamic information |

| install |
| Initializer is the maintainer of CDT |
| Sends a request to Initializer |
| Initializer merges dynamic info from existing CDT with info from new CDT |
| Requires access to >scl>admin_acs>cdt.install.acs |

| Adding, deleting, changing channels |
| Many CMF changes do not take affect immediately |
| Some require FNP reboot to take affect |
| Tables 5-1 and 5-2 in CC75 explain when changes take effect |
| Most important are adding lines and changing speed of lines |
| Require FNP reboot |
Two commands for displaying information from CDT

- `display_cdt`
  - Gives detailed information on CDT entries

- `tty_lines`
  - Gives brief information on CDT entries

- `reset_cdt_meters` resets `n_dialups`, `n_logins`, `dialed_up_time`
display_cdt

CDTE at 515|15360

| in_use:       | 3 (dialed)     |
| name:         | a.h114         |
| comment:      | DKU7102 sur sous canal mpx trt de Paris2 |
| charge_type:  | 0 (none)       |
| service_type: | 1 (login)      |
| current_service_type: | 1 (login) |
| dim:          | 1 (tty)        |
| line_type:    | 1 (ASCII)      |
| terminal_type:| DKU7102        |
| baud_rate:    | 1200           |
| fnp_no:       | 1 (a)          |
| flags.attributes: | hardwired,dont_read_answerback,check_acs; |
| event:        | 000470001164407777000107 |
| tra_vec:      | 3 (wait_login_line) |
| count:        | 1              |
| twx:          | 46             |
| state:        | 5 (dialed up)  |
| current_terminal_type: | AJ510 |
| cur_line_type:| 1 (ASCII)      |
| tty_id_code:  | none           |
| process:      | 7777711       |
| next_channel: | 0              |
| n_dialups:    | 393            |
| n_logins:     | 348            |
| dialed_up_time: | 1083 hrs 45 mins 11 secs. |
| dialup_time:  | 02/20/84 1727.4 hfh Mon |
| recent_wakeup_count: | 1 |
| recent_wakeup_time: | 02/20/84 1807.9 hfh Mon |
Values for tra_vec in display_cdt output and WP column in tty_lines output

1 wait_dialup  Channel waiting for dialup.
2 wait_answerback  WRU sent, waiting for reply
3 wait_login_line  Greeting typed, wait for login command.
4 wait_login_args  Want rest of login line
5 wait_old_password  "-cpw" was specified. Wait for old password.
6 wait_password  Waiting for password. (If "-cpw", repeat of new one.)
7 wait_new_password  "-cpw" was specified. Wait for new password
8 wait_logout_sig  Channel is hooked up. Wait for logout.
9 wait_logout  A logout has been requested. Wait for process to die
10 wait_logout_hold  As above but don't hang up when it dies.
11 wait_detach  As above but ignore channel afterwards.
12 wait_new_proc  As above but make new process and continue.
13 wait_remove  As above but completely expunge channel.
14 wait_fin_priv_attach  When channel dials up, connect it to user
15 wait_dial_release  Waiting for master process to release.
16 wait_dial_out  Waiting for auto call to complete
17 wait_hangup  Wait for the hangup event to occur for a channel
18 wait_slave_request  Ignore line until someone asks
19 wait_greeting_msg  Print greeting message and wait for login line
20 wait_delete_channel  Channel deleted - mark cdte after process is destroyed
21 wait_connect_request  logged in; awaiting request re disconnected processes
22 wait_tandd_hangup  when channel hangs up, proceed with t & d attachment
23 wait_fin_tandd_attach  when channel dials up, finish t & d attachment
24 wait_discard_wakeups  disregard all wakeups on channel
25 wait_before_hangup  allow output to print before hanging up

Values for state in display_cdt output and S column in tty_lines output

-1 masked  Terminal channel is there, but masked by MCS
1 hung  Terminal channel is there, but dead.
2 known  Channel being "listened" to, awaiting dialup.
5 dialed  Channel is dialed up. This is normal state.

Values for in_use in display_cdt output and A column in tty_lines output

0 free  Entry is empty.
1 hung up  Entry is usable but tty is hung up.
2 listening  Entry is waiting for phone call.
3 dialed  Entry is connected but login not complete.
4 logged in  Entry is logged in but no process.
5 logged in & proc  Entry has a valid process.
6 dialing  Entry (auto_call line) is dialing
7 dialed out  Entry (auto_call line) is in use
tty_lines

Attached lines = 132 (size = 136) at 02/20/84 1817.0

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>No. S WP A Baud User</th>
</tr>
</thead>
<tbody>
<tr>
<td>a.c000</td>
<td>(NU)</td>
<td>0 5 18 2 9600 COLTS executive channel</td>
</tr>
<tr>
<td>a.h000</td>
<td>(NU)</td>
<td>0 5 0 1 1200 console pupitre</td>
</tr>
<tr>
<td>a.h001</td>
<td>ROSY</td>
<td>125 5 3 3 1200 console datanet</td>
</tr>
<tr>
<td>a.h002</td>
<td>AJ510</td>
<td>171 5 6 3 1200 AJ510 dans salle de consoles 125</td>
</tr>
<tr>
<td>a.h003</td>
<td>AJ860</td>
<td>155 5 17 1 1200 AJ860 dans salle de consoles 121</td>
</tr>
<tr>
<td>a.h006</td>
<td>(NU)</td>
<td>0 0 0 1 1200 X.25 major channel</td>
</tr>
<tr>
<td>a.h006.d01</td>
<td></td>
<td>1649 1 18 1 300 X.25 dial_out sub-channel</td>
</tr>
<tr>
<td>a.h006.d02</td>
<td></td>
<td>603 1 18 1 300 X.25 dial_out sub-channel</td>
</tr>
<tr>
<td>a.h006.d03</td>
<td></td>
<td>228 1 18 1 300 X.25 dial_out sub-channel</td>
</tr>
<tr>
<td>a.h006.d04</td>
<td></td>
<td>89 1 18 1 300 X.25 dial_out sub-channel</td>
</tr>
<tr>
<td>a.h006.d05</td>
<td></td>
<td>38 1 18 1 300 X.25 dial_out sub-channel</td>
</tr>
<tr>
<td>a.h006.d01</td>
<td>ASCII_CAPS</td>
<td>2788 5 8 5 1200 Desgoutte CNIP2 (none) X.25 login sub-channel</td>
</tr>
<tr>
<td>a.h006.d02</td>
<td>MINITEL</td>
<td>1790 2 1 2 1200 X.25 login sub-channel</td>
</tr>
<tr>
<td>a.h006.d03</td>
<td>ASCII_CRT</td>
<td>1441 2 1 2 1200 X.25 login sub-channel</td>
</tr>
<tr>
<td>a.h006.d04</td>
<td>ASCII_CRT</td>
<td>114 2 1 2 1200 X.25 login sub-channel</td>
</tr>
<tr>
<td>a.h006.d05</td>
<td>ASCII_CRT</td>
<td>910 2 1 2 1200 X.25 login sub-channel</td>
</tr>
<tr>
<td>a.h006.d06</td>
<td>ASCII_CRT</td>
<td>719 2 1 2 1200 X.25 login sub-channel</td>
</tr>
<tr>
<td>a.h006.d07</td>
<td>ASCII_CRT</td>
<td>587 2 1 2 1200 X.25 login sub-channel</td>
</tr>
</tbody>
</table>

**tty_lines**

**display_cdt**

**field**

**Name**

**Type**

**No.**

**S**

**WP**

**A**

**Baud**

**User**

**n_dialups**

**state**

**tra_vec**

**in_use**

**baud**

**comment**
An FNP image is the software to load into the FNP. It consists of 'object decks' of individual programs bound together by the bind_fnp command.

Two FNP images are delivered in system libraries:
- >unb>mcs
- >unb>site_mcs

When the FNP is booted (during Multics bootload or by operator command or after FNP crash) this software is sent to the FNP from Multics.

Before loading, configuration information from the CDT is patched in.

An FNP's image must contain all software necessary to run the types of lines configured on the FNP.

An image should not have unneeded software:
- This would waste FNP memory that could be used for I/O buffers.

Choosing the necessary modules is the main work in creating an image:
- All software must fit in <32K

The site_mcs and mcs images contain software for several types of lines.

Sites create their own FNP images tailored to their FNP line configurations.

Often have a different image for each FNP:
- Image is stored in a segment whose pathname is specified in CDT.
<table>
<thead>
<tr>
<th>Module Name</th>
<th>Needed if:</th>
<th>Which should happen:</th>
</tr>
</thead>
<tbody>
<tr>
<td>dia_man</td>
<td>always</td>
<td>always</td>
</tr>
<tr>
<td>interpreter</td>
<td>always</td>
<td>always</td>
</tr>
<tr>
<td>scheduler</td>
<td>always</td>
<td>always</td>
</tr>
<tr>
<td>utilities</td>
<td>always</td>
<td>always</td>
</tr>
<tr>
<td>init</td>
<td>always</td>
<td>always</td>
</tr>
<tr>
<td>control_tables</td>
<td>line_type: ascii;</td>
<td>usually slows up for speed;</td>
</tr>
<tr>
<td>hsla_man</td>
<td>hsla: &gt;0;</td>
<td>usually</td>
</tr>
<tr>
<td>trace</td>
<td>module: trace;</td>
<td>usually</td>
</tr>
<tr>
<td>meters</td>
<td>meter: yes;</td>
<td>usually</td>
</tr>
<tr>
<td>mclt</td>
<td>line_type: COLTS</td>
<td>usually keeps for happy</td>
</tr>
<tr>
<td>gl15_tables</td>
<td>line_type: gl15</td>
<td>sometimes (Level-6 remotes)</td>
</tr>
<tr>
<td>bsc_tables</td>
<td>line_type: bsc;</td>
<td>sometimes (HASP, IBM remotes)</td>
</tr>
<tr>
<td>x25_tables</td>
<td>line_type: x251ap;</td>
<td>sometimes (real Multics x.25)</td>
</tr>
<tr>
<td>hasp_tables</td>
<td>multiplexer_type: hasp</td>
<td>sometimes (HASP)</td>
</tr>
<tr>
<td>vip_tables</td>
<td>line_type: vip</td>
<td>sometimes (Sync VIPs)</td>
</tr>
<tr>
<td>polled_vip_tables</td>
<td>line_type: polled_vip</td>
<td>sometimes (Polled sync VIPs)</td>
</tr>
<tr>
<td>ibm3270_tables</td>
<td>line_type: bsc;</td>
<td>sometimes (3270 controller)</td>
</tr>
<tr>
<td>acu_tables</td>
<td>service: autocal;</td>
<td>sometimes (non-x.25 dialout)</td>
</tr>
<tr>
<td>autobaud_tables</td>
<td>baud: auto;</td>
<td>sometimes</td>
</tr>
<tr>
<td>ic_sampler</td>
<td>debug_fnp ic_sample</td>
<td>rare</td>
</tr>
<tr>
<td>breakpoint_man</td>
<td>debug_fnp breakpoints</td>
<td>rare</td>
</tr>
<tr>
<td>console_man</td>
<td>console: yes;</td>
<td>rare</td>
</tr>
</tbody>
</table>

Multiplexing can be done via remote.

or

A process can run controller for 3270's.
The bind_fnp command uses a control file with .bind_fnp suffix to select software modules to put in FNP image.
/* bindfile for MCS, Multics Communications System */

version: 6.5d;

lsla: 0;
hsla: 3;
memory: 64; (correspond to cdT, otherwise lower assumed.
console: no;
printer: no;
meter: yes;

/* module load list - init module must be last */

order: scheduler,
       interpreter,
       control_tables,
       dia_man,
       melt,
       hsla_man,
       utilities,
       trace,
       bsc_tables,
       hasp_tables,
       x25_tables,
       meters,
       init;

/* entry to init from bootload */

entry: istart;

/* table size specifications */

module: hsla_man;
   type: hsla;
   size: 97;
module: trace;
   type: trace;
   mask: 317777; /* trace enable mask */
   size: 2048;

   /* mask bits: 0 scheduler
               1 dia_man
               2 interpreter
               3 utilities
               4 lsla_man
               5 hsla_man */

end;
LDD Structure

```
+--------------------------+
| ldd                      |
+--------------------------+
| mcs                      |
+--------------------------+
| source                   |
| *.s.archive              |
+--------------------------+
| object                   |
| *.archive                |
+--------------------------+
| info                     |
| 355_macros               |
| macros.map355            |
| macros_asm               |
| site_mcs.bind_fnp        |
| site_mcs.list            |
| mcs.bind_fnp             |
| mcs.list                 |
```
SLDD Structure

+--------------+
| sldd         |
+--------------+
|               |
| mcs          |
+--------------+
|               |
+--------------+
| source       |
+--------------+
| *.s.ud.archive |
+--------------+
| object       |
+--------------+
| *.objdk      |
+--------------+
| info         |
| fnp_a.bind_fnp |
| fnp_a.list   |
| fnp_a.search |
| fnp_b.bind_fnp |
| fnp_b.list   |
| fnp_b.search |
| 355_macros   |
| macros.map355 |
map355

The assembler for MAP355, the assembler language of the FNP

Actually calls GCOS emulator to call assembler

GCOS software is necessary

toto.map355 -> toto.objdk

-list control argument

-macro_file control argument

Default is -macro_file >ldd>mcs>info>355_macros

macros

Like include files for PL/I

gcos >ldd>mcs>info>macros_asm -truncat (-list -lower_case).

macros.map355 -> 355_macros
- Attributes in SAT/PDT
  - dialok : slave disconnected
  - save_on_disconnect
  - disconnect_ok
* Config deck

<table>
<thead>
<tr>
<th>PRPH FNP A A 18. ON</th>
</tr>
</thead>
<tbody>
<tr>
<td>PARM TTYB 10240.</td>
</tr>
</tbody>
</table>

| Default = 6K        |

MCS Administration
installation_parms

| cwe_count
| Initializer hangs up login line if more than cwe_count wakeups in cwe_time seconds
| Default 10

| cwe_time
| Default 3 seconds

| login_time
| Initializer hangs up login line if no login in login_time seconds
| Typical value 180 seconds
| Not used for lines marked attributes: hardwired

| device_prices, device_names
| Used in conjunction with charge_type: in CDT
Operator commands

load_mpx

Start FNP or lower-level multiplexer

Starting FNP means loading it with image

Then load lower-level multiplexers

And start listening to all non-multiplexed lines that are not service: inactive

Starting lower-level multiplexer is similar to listening to a non-multiplexed lines

Accept connection, and when connection is made initiate protocol

Done automatically during Multics bootload (during Answering Service startup) and after multiplexer crash

Done manually with load_mpx to change some line attrs, add lines

Any lines already in use must be hung up before reloading multiplexer

Use -force to force hangups

dump_mpx

Stops multiplexer completely

If it is an FNP, a dump is taken

stop_mpx

Stops further dialups from being accepted by multiplexer

Used when an FNP with logged in users must be reloaded

stop_mpx

send_message or warn or bump fnp

load_mpx

Prevents automatic loading if done before Answering Service startup

start_mpx

Undoes the effect of stop_mpx

bump fnp tag
| Allows bumping of users on a single FNP

| detach
| Makes a line inactive, bumping any user connected on the line

| remove
| Like detach, but uses a more brutal method of terminating user's process

| attach
| Makes a line active, undoing the effect of detach or remove
| Can also recover masked channels

| accept
| Accepts the line as an operator terminal
| Line must be service: mc, or can type 'dial system' on a login terminal

| drop
| Undoes the affect of an accept
Terminal Characteristics

Many terminal characteristics can be modified:

- Character set/code
- Character sequences to move carriage/cursor
- Use of tabs
- Padding
- Terminal initialization (set tabs, etc.)
- Line/page length
- Character echoing modes

These characteristics can all be modified several different ways:

- TTF
- \texttt{stty}
- \texttt{io\_call control, io\_call modes}
- \texttt{iox\_control, iox\_modes}

We will look at how to modify them in per terminal-type in TTF and dynamically using \texttt{iox_}

\texttt{stty} and \texttt{io\_call} are just command-level interfaces to \texttt{iox_}
TTF/TTT

| ASCII source Terminal Type File is used to produce Terminal Type Table

| Has three major types of information

| Characteristics of different terminal types for use by Ring 0 and FNP

| Characteristics of different terminal types for use by the Video System

| Definition of some preaccess commands

| Definition of terminal answerbacks

| System TTT is in >scl>ttt

| cv_ttf

| Converts TTF to TTT

| Usually TTF.ttf -> TTF.ttt

| install

| The install command is used to install new TTT in >scl>ttt

| Unlike CMF, there is no dynamic information

| Process calling the install command does the installation itself

| >scl>ttt is protected by ring brackets of 1,5,5

| Access required to >scl>admin_acs>ttt.install.acs

| Can use private TTTs

| set_ttt_path

| Specifies the TTT to use in process

| Preaccess command and answerbacks not meaningful in private TTT, since they are only handled by Initializer

| display_ttt

| Used to display some or all of a TTT

| TTF Syntax

| TTF is a typical source file for a table with statements identified by keywords
Many statements take characters or lists of characters as arguments. Such characters may be specified in several different ways:

- Single unquoted character, e.g. X
- Single quoted character, e.g. "X" or ";"
- 1 to 3 digit octal number, e.g. 033
- Control character name, e.g. ESC
- Control characters in form ^A

Repetition of single characters or groups of characters is possible:

- Uses (N) for repetition factor
- < ... > to group characters
Terminal Type Definition

Input_conversion: standard_input_conv;

/***** Typical ASCII teleprinter terminal *****/
terminal_type: ASCII;
   modes: default,1179,pl,can_type=overstrike,hndlquit,
       fulldpex,echoplex,crecho,lfecho,^tabs,tabecho;
   bauds: 110 133 150 300 600 1200 ...;
   vert_nl_delays: 0 1 1 5 9 18 ...;
   horz_nl_delays: 0.00 0.012 0.025 0.019 0.060 0.120 ...;
   const_tab_delays: 0 0 0 0 2 3 ...;
   var_tab_delays: 0 0.180 0.250 0.250 0.500 1.000 ...;
   backspace_delays: 0 0 0 0 1 2 ...;
   vt_ff_delays: 9 24 29 59 100 200 ...;
   output_conversion: ascii_output_conv;
   special: ascii_special;
   line_types: ASCII, VIP, POLLED_VIP;
   old_type: 11;

/***** ASCII teleprinter terminal (upper-case only) *****/
terminal_type: ASCII_CAPS like ASCII;
   modes: default,1179,pl,can_type=replace,hndlquit,capo,
       fulldpex,echoplex,crecho,lfecho,^tabs,tabecho;
   special: ascii_caps_special;
   input_translation: ascii_caps_input_trans;
   old_type: 7;

/***** Typical ASCII crt terminal *****/
terminal_type: ASCII_CRT like ASCII;
   modes: default,1179,p123,can_type=replace,hndlquit,scroll,
       fulldpex,echoplex,crecho,lfecho,^tabs,tabecho;
   bauds: 110 133 150 300 600 1200 ...;

/***** ASCII CRT terminal (upper-case only) *****/
terminal_type: ASCII_CRT_CAPS like ASCII_CAPS;
   modes: default,1179,p123,can_type=replace,hndlquit,scroll,capo,
       fulldpex,echoplex,crecho,lfecho,^tabs,tabecho;
   bauds: 110 133 150 300 600 1200 ...;

/***** TeleVideo, Inc. Model TVI-912 *****/
terminal_type: TVI912 like ASCII_CRT;
   modes: default,1179,p123,can_type=replace,hndlquit,scroll,
       fulldpex,echoplex,crecho,lfecho,^tabs,tabecho;
   initial_string: ESC 3 CR ESC 1 (7) <(10) <SP> ESC 1> CR;
   special: tvi912_tvi920_special;
TTY Modes and Control Orders

| Look at many functions of Ring Zero and FNP by looking at user interface |
| The tty I/O module and asynchronous lines are used as examples |
| Much processing is the same for non-multiplexed synchronous lines, and for subchannels of multiplexers |
| Comparison with other line types is made whenever possible |

Terminal Initialization

| Initial string specified in TTT |
| Sent to terminal at dialup time |
| Generally used to set tabs |
| Can be resent using send_initial_string control order |
| Only useful for asynchronous terminals |

Input Conversion, Translation, etc.

| All terminal input goes through four steps of conversion and translation |
| Translate to ASCII |
| Put in canonical form |
| Process erase and kill characters |
| Process input escape sequences |
| All of these steps are performed in ring 0 |
| All of these steps are exactly the same for all line types: asynchronous lines, non-multiplexed synchronous lines, multiplexer subchannels |

Translation \( \text{ibm} \text{iso} \text{asc} \) \text{smcp keys}\), but does not change what is echoed to screen.

| Input is translated into ASCII |
| Similar to PL/I translate builtin |
| Does not change size of input string |
| Does not change what is echoed to terminal |
| Echoing is done in FNP, translation is done later in Ring 0 |
ctl_char mode

When a terminal is in ctl_char mode, control characters are accepted as input.

Otherwise control characters are discarded.

If terminal is not in ctl_char mode, a second translation is done to translate all control characters into NULs.

NULs are normally discarded later in the input translation/conversion process.
Input Translation

Extract From Terminal Type File

terminal_type: ...

input_translation: ascii_caps_input_trans;

translation_table: ascii_caps_input_trans;

Extract from tty_convert.incl.pll

Used for get_input_translation and set_input_translation control orders

dcl CV_TRANS_VERSION fixed bin int static options (constant) init (2);

dcl 1 cv_trans_struc aligned based,
2 version fixed bin,
2 default fixed bin,
2 cv_trans like cv_trans;

dcl 1 cv_trans based aligned,
2 value (0:255) fixed bin (8) unal;
Canonicalization is a rule that applies when you see the string \textit{abc} in a file. When we see the string \textit{abc} in a file, we can't know if it was originally typed in \textit{abc<BS><BS><BS>}, \textit{a<BS>b<BS>c}, or even \textit{ab<BS><BS><BS><BS>a<BS><BS><BS><BS>c}}.

Similarly, if we see the line \textit{abc} on a CRT screen, it may have been typed an infinite number of different ways, such as \textit{abd<BS>c}, for example.

Canonicalization tries to apply a what-you-see-is-what-you-get rule. There are two separate cases, for hardcopy terminals and CRTs.

- \texttt{can_type-overstrike} is for hardcopy terminals. If multiple characters are typed in the same column, the first characters are overstruck on paper. Input is canonicalized to be column by column. Multiple characters in the same column are put in sorted order, separated by backspaces. In the example above, canonical form is \textit{<BS>a<BS>b<BS>c}.

- \texttt{can_type-replace} is for screen terminals. If multiple characters are typed in the same column, the first characters are replaced on screen. Input is canonicalized to be column by column. Only the last character in each column is retained, with no backspaces. Note that this does not make BS the erase character. \texttt{ab<SP><SP><BS>c} and \texttt{ab<SP><SP>#c} give different results.

\texttt{can mode}

Turns canonicalization on or off.
Erase and Kill Characters

# is normal erase character, @ is normal kill character
Any other characters may be substituted

erkl mode
Turns erase and kill processing on or off

-------------------------------------------------------------

Erase and Kill Characters
Extract From Terminal Type File

terminal_type: ...
.
erase: 'X;
kill: '^C;

--------------------------------------------------------------------------------

Extract from tty_editing_chars.incl.pll
Used for get_editing_chars and set_editing_chars control orders

dcl 1 editing_chars aligned based (editing_chars_ptr),
   2 version fixed bin,
   2 erase char (1) unaligned,
   2 kill char (1) unaligned;

dcl editing_chars_version_2 fixed bin internal static init (2);
--------------------------------------------------------------------------------

| Process input escape sequences
| Each input character is looked up in input conversion table
| Gets extra processing if marked
| Break character

| Used as end of line in get_line calls
| White space removed in front of break character
| Doesn’t mean that the character will cause line to be read immediately
| Other modes determine when FNP sends line to Multics
| Discard

ttyModes and Control Orders
Characters thrown away

Formfeed

Discarded if page length > 0

Dates from early days of EOP processing

Escape

Not to be confused with ASCII ESC (033)

Usually backslash character

Changes interpretation of following character(s)

\<NL> \# \@ \007 \\

Other escape sequences may be defined using input conversion table plus special table

es mode

Can be turned off to inhibit escape sequence processing

LSTOPS

Translation

Canonicalize

Handle erase kill

Process escape clear

tty Modes and Control Orders
Advanced MCS User Features
Input Conversion
Extract From Terminal Type File

terminal_type: ...

input_conversion: standard_input_conv;

conversion_table: standard_input_conv;

03 00 00 00 00 00 00 00 /* NUL SOH STX ETX EOT ENQ ACK BEL (000-007) */
00 00 01 00 04 00 00 00 /* BS TAB LF VT FF CR SO SI (010-017) */
00 00 00 00 00 00 00 00 /* DLE DC1 DC2 DC3 DC4 NAK SYN ETB (020-027) */
00 00 00 05 00 00 00 00 /* CAN EM SUB ESC FS GS RS US (030-037) */
00 00 00 00 00 00 00 00 /* SP ! " # $ % & ' ( ) * + , - . / (040-047) */
00 00 00 00 00 00 00 00 /* 0 1 2 3 4 5 6 7 (060-067) */
00 00 00 00 00 00 00 00 /* 8 9 : ; < = > ? (070-077) */
00 00 00 00 00 00 00 00 /* @ A B C D E F G (100-107) */
00 00 00 00 00 00 00 00 /* H I J K L M N O (110-117) */
00 00 00 00 00 00 00 00 /* P Q R S T U V W (120-127) */
00 00 00 00 02 00 00 00 /* X Y Z [ \ ] ^ _ (130-137) */
00 00 00 00 00 00 00 00 /* a b c d e f g (140-147) */
00 00 00 00 00 00 00 00 /* h i j k l m n o (150-157) */
00 00 00 00 00 00 00 00 /* p q r s t u v w (160-167) */
00 00 00 00 00 00 00 00 /* x y z ( | ) ~ DEL (170-177) */

Extract from tty_convert.incl.pll
Used for get_input_conversion and set_input_conversion control orders

dcl CV_TRANS_VERSION fixed bin int static options (constant) init (2);

dcl 1 cv_trans_struc aligned based,
  2 version fixed bin,
  2 default fixed bin,
  2 cv_trans like cv_trans;

dcl 1 cv_trans based aligned,
  2 value (0:255) fixed bin (8) unal;

dcl (INPUT_CONVERT_ORDINARY init (0),
  INPUT_CONVERT_BREAK init (1),
  INPUT_CONVERT_ESCAPE init (2),
  INPUT_CONVERT_DISCARD init (3),
  INPUT_CONVERT_FORMFEED init (4),
  INPUT_CONVERT_PRECEDENCE_DISCARD init (5)
) fixed bin (8) unaligned internal static options (constant);
Special Characters Table
(Only Parts Used In Input Conversion Are Shown Here)
Extract From Terminal Type File

terminal_type: ...
 .
special: ascii_caps_special;
 .
special_table: ascii_caps_special;
(most of table is used in output conversion)

input_escapes: /* <escape> <lc-alpha> -> <uc-alpha> (eg: \a -> A) */
  "a" "A", "b" "B", "c" "C", "d" "D", "e" "E",
  "f" "F", "g" "G", "h" "H", "i" "I", "j" "J",
  "k" "K", "l" "L", "m" "M", "n" "N", "o" "O",
  "p" "P", "q" "Q", "r" "R", "s" "S", "t" "T",
  "u" "U", "v" "V", "w" "W", "x" "X", "y" "Y", "z" "Z":

Extract from tty_convert.incl.pll
(Only Parts Used In Input Conversion Are Shown Here)
Used for get_special and set_special control orders

dcl SPECIAL_VERSION fixed bin int static options (constant) init (1);

dcl 1 c_chars based aligned,
  2 count fixed bin (8) unaligned,
  2 chars (3) char (1) unaligned;

dcl 1 special_chars_struc aligned based,
  2 version fixed bin,
  2 default fixed bin,
  2 special_chars,
(most of table is used in output conversion)
  3 escape_length fixed bin,
  .
  3 input_escapes aligned,
    4 len fixed bin (8) unaligned,
    4 str char (sc_input_escape_len refer
      (special_chars_struc.input_escapes.len)) unaligned,
  3 input_results aligned,
    4 pad bit (9) unaligned,
    4 str char (sc_input_escape_len refer
      (special_chars_struc.input_escapes.len)) unaligned;

dcl 1 get_special_info_struc based aligned,
  2 area_ptr pointer,
  2 table_ptr pointer;
All of the input translation and conversion steps can be bypassed

rati mode

Output Conversion, Translation, etc.

All terminal output goes through three steps of conversion and translation

Capitalization

Formatting

Translation from ASCII

All of these steps are performed in ring 0

All of these steps are exactly the same for all line types: asynchronous lines, non-multiplexed synchronous lines, multiplexer subchannels

Capitalization

capo mode

If in capo mode, translate all alphabets to uppercase

Formatting

ll mode

^ll or llNN, e.g. 1179

Insert <NL> after NN characters

pl mode

^pl or plNN, e.g. pl23

Stop with End of Page string after NN lines

End of Page string specifiable in special table

Continue with CR or FF

CR thrown away if on line by itself

FF always thrown away if NN > 0

Originally only FF was accepted

scroll mode
Used in conjunction with pI mode
scroll intended for scrolling terminals
  EOP after NN lines of continuous output
  Line counter reset after user input
^scroll intended for non-scrolling screens, e.g. TEK4014
  Counter not reset, user input adds to counter
  EOP after NN lines of input or output
edited mode
  In ^edited mode, non-printing characters are output as octal escapes are printed, e.g. \000
  In edited mode, non-printing characters are discarded on output
tabs mode
  In tabs mode, white space is optimized into minimum number of tabs and blanks
  Terminal's tabs must have been set by initial string
red mode
  In red mode, SO (\016 or BRS, produced by ioa_ ^R) causes terminal to shift to color
    SI (\017 or BRS, produced by ioa_ ^B) changes back to normal
    Used in some system error messages
    Actually sends sequence from special table
    On old terminals, shifted to red ribbon
    On CRTs, can change to inverse video
vertsp mode
  In vertsp mode, FF and VT are sent to terminal
    Actually sends sequence from special table
    In ~vertsp mode, FF and VT are sent as octal escapes
Output Conversion

Output conversion uses the output_conversion and special tables.
Output Conversion
Extract From Terminal Type File

terminal_type: ...
.
output_conversion: ascii_output_conv;
.
conversion_table: ascii_output_conv;

07 07 07 07 07 07 07 07 07 07 12 /* NUL SOH STX ETX EOT ENQ ACK BEL (000-007) */
04 03 01 05 06 02 10 11 /* BS TAB LF VT FF CR SO SI (010-017) */
07 07 07 07 07 07 07 07 07 07 07 /* DLE DC1 DC2 DC3 DC4 NAK SYN ETB (020-027) */
07 07 07 07 07 07 07 07 07 07 07 /* CAN EM SUB ESC FS GS RS US (030-037) */
00 00 00 00 00 00 00 00 00 00 00 /* SP ! " $ % & ' ( ) * + , - . / 010-017 */
00 00 00 00 00 00 00 00 00 00 00 /* ) * + , - . / 010-017 */
00 00 00 00 00 00 00 00 00 00 00 /* 01 2 3 4 5 6 7 060-067 */
00 00 00 00 00 00 00 00 00 00 00 /* 08 9 : ; < = > ? (070-077) */
00 00 00 00 00 00 00 00 00 00 00 /* @ A B C D E F G (100-107) */
00 00 00 00 00 00 00 00 00 00 00 /* H I J K L M N O (110-117) */
00 00 00 00 00 00 00 00 00 00 00 /* P Q R S T U V W (120-127) */
00 00 00 00 00 00 00 00 00 00 00 /* X Y Z [ \ ] ^ _ (130-137) */
00 00 00 00 00 00 00 00 00 00 00 /* ` a b c d e f g (140-147) */
00 00 00 00 00 00 00 00 00 00 00 /* h i j k l m n o (150-157) */
00 00 00 00 00 00 00 00 00 00 00 /* p q r s t u v w (160-167) */
00 00 00 00 00 00 00 00 00 00 14; /* x y z { | } ~ DEL (170-177) */
Extract from tty_convert.incl.pll
Used for `get_output_conversion` and `set_output_conversion` control orders

```plaintext
dcl CV_TRANS_VERSION fixed bin int static options (constant) init (2);

dcl 1 cv_trans_struc aligned based,
    2 version fixed bin,
    2 default fixed bin,
    2 cv_trans like cv_trans;

dcl 1 cv_trans based aligned,
    2 value (0:255) fixed bin (8) unal;

dcl (OUTPUT_CONVERT_ORDINARY init (0),
    OUTPUT_CONVERT_NEWLINE init (1),
    OUTPUT_CONVERT_CR init (2),
    OUTPUT_CONVERT_HT init (3),
    OUTPUT_CONVERT_BS init (4),
    OUTPUT_CONVERT_VT init (5),
    OUTPUT_CONVERT_FF init (6),
    OUTPUT_CONVERT_OCTAL init (7),
    OUTPUT_CONVERT_RRS init (8),
    OUTPUT_CONVERT_BRS init (9),
    OUTPUT_CONVERT_NO_MOTION init (10),
    OUTPUT_CONVERT_PRECEDENCE NO_MOTION init (11),
    OUTPUT_CONVERT_DONT_SEND init (12),
    OUTPUT_CONVERT_FIRST_SPECIAL init (17)
) fixed bin (8) unaligned internal static options (constant);
```
Special Characters Table
Extract From Terminal Type File

terminal_type: ...

special: ascii_caps_special;

special_table: ascii_caps_special;

new_line: CR LF;
carriage_return: CR;
backspace: BS;
tab: TAB;
vertical_tab: VT CR;
form_feed: FF CR;
printer_on: ; pasword masking shift not in echo mode IF terminal
printer_off: ; enables local echoing
red_shift: 
black_shift: 
end_of_page: E 0 P;

input_escapes: /* <escape> <lc-alpha> -> <uc-alpha> (eg: \a -> A) */
    "a" "A",
    "b" "B",
    "c" "C",
    "d" "D",
    "e" "E",
    "f" "F",
    "g" "G",
    "h" "H",
    "i" "I",
    "j" "J",
    "k" "K",
    "l" "L",
    "m" "M",
    "n" "N",
    "o" "O",
    "p" "P",
    "q" "Q",
    "r" "R",
    "s" "S",
    "t" "T",
    "u" "U",
    "v" "V",
    "w" "W",
    "x" "X",
    "y" "Y",
    "z" "Z";
Extract from tty_convert.incl.pll

Used for get_special and set_special control orders

dcl SPECIAL_VERSION fixed bin int static options (constant) init (1);

dcl 1 c_chars based aligned,
    2 count fixed bin (8) unaligned,
    2 chars (3) char (1) unaligned;

dcl 1 special_chars_struc aligned based,
    2 version fixed bin,
    2 default fixed bin,
    2 special_chars,
        3 nl_seq aligned like c_chars,
        3 cr_seq aligned like c_chars,
        3 bs_seq aligned like c_chars,
        3 tab_seq aligned like c_chars,
        3 vt_seq aligned like c_chars,
        3 ff_seq aligned like c_chars,
        3 printer_on aligned like c_chars,
        3 printer_off aligned like c_chars,
        3 red_ribbon_shift aligned like c_chars,
        3 black_ribbon_shift aligned like c_chars,
        3 end_of_page aligned like c_chars,
        3 escape_length fixed bin,
        3 not_edited_escapes (sc_escape_len refer
to special_chars_struc.escape_length) like c_chars,
        3 edited_escapes (sc_escape_len refer
to special_chars_struc.escape_length) like c_chars,
        3 input_escapes aligned,
            4 len fixed bin (8) unaligned,
            4 str char (sc_input_escape_len refer
to special_chars_struc.input_escapes.len) unaligned,
        3 input_results aligned,
            4 pad bit (9) unaligned,
            4 str char (sc_input_escape_len refer
to special_chars_struc.input_escapes.len) unaligned;

dcl 1 get_special_info_struc based aligned,
    2 area_ptr pointer,
    2 table_ptr pointer;

tty Modes and Control Orders

Advanced MCS User Features

F86 Page 3-19
Translation

| Translate from ASCII to terminal's character set
| To bypass all of the above
| rawo mode
| All of the above output conversion/translation is done in ring 0

Echoing

| Echoing is done in the FNP
| Echoing makes sense only for asynchronous terminals
| Asynchronous terminals connected via X.25 are a special case
| FNP cannot do echoing, because it does not know about logical channels
| Echoing could be done in Ring 0, but would be expensive
| X.25 has parameters to tell PAD to do echoing

fulldpx mode

| Specifies that line can send and receive at same time
| Echoing implies fulldpx
| fulldpx does not imply echoing
| If ^fulldpx, Multics will not send while receiving

echoplex mode

| Each character sent by terminal is echoed back
| No selection, translation, conversion, etc. possible.

lfecho mode

| When CR is received, LF is sent to terminal and placed in input stream
| NL (same as LF) is only true end-of-line character for Multics
| When in ^lfecho mode, CRs will appear to be ignored

crecho mode
When LF is received, CR is echoed
Not allowed for X.25 lines
There is no X.25 parameter to have PAD do echo

When tab is received, FNP sends back spaces to move carriage/cursor to next tab stop
Not allowed for X.25 lines
There is no X.25 parameter to have PAD do echo

Quit Handling
There are a number of steps in handling break condition
Handled both in ring 0 and in FNP
True break condition can only occur on asynchronous lines
Synchronous protocols such as POLLED_VIP, X.25 have ways to achieve similar effect
There are two separate controls on this handling
Quits are enabled or disabled by control orders
quit_enable control order
quit_disable control order
hndlquit mode

If quit enabled:
1) FNP sends NL, if in hndlquit mode
2) FNP flushes buffers, if in hndlquit mode
3) FNP tells Ring 0
4) Ring 0 flushes buffers, if in hndlquit mode
5) Ring 0 wakes up user process, signalling "quit" condition
6) Ring 4 default quit handler prints "QUIT"

For an X.25 line, FNP cannot handle individual logical channel
1) PAD/X.25 network flush buffers, if in hndlquit mode
2) PAD/X.25 network tells FNP (via control packet)
3) FNP sends control packet to ring 0
4) Ring 0 sends NL, if in hndlquit mode
5) Ring 0 flushes buffers, if in hndlquit mode
6) Ring 0 wakes up user process, signalling "quit" condition
7) Ring 4 default quit handler prints "QUIT"

Polite, Replay

Sending output while user is typing can be disruptive

Modes exist to make it less disruptive

Valid only for asynchronous lines and X.25 subchannels

Handled in FNP

polite mode

If any output occurs while user is typing input line will hold output until user finishes input line

Timeout after 30 seconds

Some X.25 networks have parameter to have PAD handle this

prefixnl mode

Interrupting output is prefixed with a newline

If input line interrupted because ^polite, or 30 second timeout

Not possible for X.25 lines

replay mode

Replays interrupted input line

If input line interrupted because ^polite, or 30 second timeout

Not possible for X.25 lines

Non-line-oriented input

Normally the FNP buffers a line of input and then sends entire line to Multics

tty Modes and Control Orders

Advanced MCS User Features
On the Multics (Ring 0) side, the user process is woken up whenever input is sent from FNP to Multics.

There are ways to change when the FNP forwards data and when a user process is woken up.

**breakall mode**

- Causes FNP to forward each character to Multics.
- User process woken up for each character.
- Necessary or useful for certain applications.
  - Input from devices that never send CR or LF.
  - **dial_out**
  - Emacs and Video System.
  - Comparatively expensive.

**Echo Negotiation**

- Echo negotiation is an optimization to make it less expensive for Emacs and Video System.
- Typical situation in Emacs/Video is entering text at end of line.
- For each normal character that user types, all that needs to be done is echo and store.
- This is almost what FNP does in echoplex mode.
- Additional information needed.
  - List of characters that require more than simple echo.
  - Number of columns remaining on the current line of the screen.

No documented user interface for echo negotiation.

To use echo negotiation:

- `set_echo_break_table` control order.
  - Input is table of 128 bits in table, telling FNP what chars to stop echoing on.
  - Then call `hcs_tty_read_echoed`.
    - Input: `buffer_ptr`, `buffer_size`, `columns_left`.
Output: #characters returned, #characters echoed

For X.25 lines, cannot be handled in FNP, nor in PAD

Handled in Ring 0

wake_tbl mode

Changes when user process is woken up

Handled in Ring 0

Valid for all types of lines

Normally woken up for every input

Sometimes better to wake up less often

Consider qedx input mode

qedx has almost nothing to do, and nothing to print, until a \ is read

wake_tbl specifies that wakeup should be sent only when specified characters are in input

Table must be set by control order before turning on wake_tbl mode
Wakeup Table
Not settable in TTF

Extract from set_wakeup_table_info.incl.pll
Used for set_wakeup_table control order

dcl swt_info_version_l fixed bin static options (constant) init (1);

dcl 1 swt_info aligned based (swt_infop),
    2 version fixed bin,
    2 new_table like wakeup_table,
    2 old_table like wakeup_table;

dcl wakeup_tablep ptr;

dcl 1 wakeup_table aligned based (wakeup_tablep),
    2 wake_map (0:127) bit (1) unal,
    2 mbz bit (16) unal;

Padding

| Many hardcopy terminals handle normal characters at line speed
| But require more time for carriage movement
| Padding (i.e. delay) characters can be sent to allow time for carriage movement
| Padding is done with NUL characters, which terminal discards
| Padding can be done for CR/LF, Tabs, BS, FF
| Specified in number of delay characters, or number of delay characters per column
| Padding requirements are different at different speeds
| Determining padding requirements is usually a matter of guessing and experimenting
| Padding for output is done in Ring 0
| Padding for echoed characters is done in FNP
| Can be done for any type of line, but only makes sense for asynchronous lines and X.25 subchannels
For X.25 no padding will be done for echoed characters

Not a problem

But X.25 does not know speed of terminal, only of multiplexed line

This has bad effects for padding
Delays
Extract From Terminal Type File

terminal_type: ...

<table>
<thead>
<tr>
<th>bauds</th>
<th>110</th>
<th>133</th>
<th>150</th>
<th>300</th>
<th>600</th>
<th>1200</th>
</tr>
</thead>
<tbody>
<tr>
<td>vert_nl_delays</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>9</td>
<td>18</td>
</tr>
<tr>
<td>horz_nl_delays</td>
<td>0.00</td>
<td>0.012</td>
<td>0.025</td>
<td>0.019</td>
<td>0.060</td>
<td>0.120</td>
</tr>
<tr>
<td>const_tab_delays</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>...</td>
</tr>
<tr>
<td>var_tab_delays</td>
<td>0</td>
<td>0.180</td>
<td>0.250</td>
<td>0.250</td>
<td>0.500</td>
<td>1.000</td>
</tr>
<tr>
<td>backspace_delays</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>...</td>
</tr>
<tr>
<td>vt_ff_delays</td>
<td>9</td>
<td>24</td>
<td>29</td>
<td>59</td>
<td>100</td>
<td>200</td>
</tr>
</tbody>
</table>

Extract from tty_convert.incl.pl1
Used for get_delay and set_delay control orders

dcl DELAY_VERSION fixed bin int static options (constant) init (1);

dcl 1 delay_struc aligned based,
    2 version fixed bin,
    2 default fixed bin,
    2 delay like delay;

dcl 1 delay based aligned,
    2 vert_nl fixed bin,
    2 horz_nl float bin,
    2 const_tab fixed bin,
    2 var_tab float bin,
    2 backspace fixed bin,
    2 vt_ff fixed bin;
Output Flow Control

Some terminals cannot accept output as fast as Multics can send it.

For synchronous lines, this is handled by protocol.

For asynchronous lines, several modes exist to solve the problem.

These solutions are not valid for X.25.

Flow control must be between PAD and terminal.

No X.25 parameters for this problem.

All flow control for asynchronous lines is handled in FNP.

Overflow mode

Turns on either of the two types of output flow control.

ETB/ACK output flow control.

Suspend-Resume output flow control.
ETB-ACK Output Flow Control Protocol

<table>
<thead>
<tr>
<th>Characters</th>
<th>Meanings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multics: thisisETB</td>
<td>Here's a block of output for you.</td>
</tr>
<tr>
<td>Terminal: ACK</td>
<td>OK, I can take another block of up to 256 characters.</td>
</tr>
<tr>
<td>Multics: theoutETB</td>
<td>Here's a block of output for you.</td>
</tr>
<tr>
<td>Terminal: ACK</td>
<td>OK, I can take another block of up to 256 characters.</td>
</tr>
<tr>
<td>Multics: putforETB</td>
<td>Here's a block of output for you.</td>
</tr>
<tr>
<td>Terminal: ACK</td>
<td>OK, I can take another block of up to 256 characters.</td>
</tr>
</tbody>
</table>
### Suspend/Resume Output Flow Control Protocol

<table>
<thead>
<tr>
<th>Character</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multics: t</td>
<td>Here's a character for you.</td>
</tr>
<tr>
<td>Multics: h</td>
<td>Here's a character for you.</td>
</tr>
<tr>
<td>Multics: i</td>
<td>Here's a character for you.</td>
</tr>
<tr>
<td>Multics: s</td>
<td>Here's a character for you.</td>
</tr>
<tr>
<td>Multics: i</td>
<td>Here's a character for you.</td>
</tr>
<tr>
<td>Multics: s</td>
<td>Here's a character for you.</td>
</tr>
<tr>
<td>Terminal: X-OFF</td>
<td>Wait, I can't handle any more for the moment.</td>
</tr>
</tbody>
</table>

(Silence while Multics waits for OK to resume.)

<table>
<thead>
<tr>
<th>Character</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terminal: X-ON</td>
<td>OK, I can handle some more now.</td>
</tr>
<tr>
<td>Multics: t</td>
<td>Here's a character for you.</td>
</tr>
<tr>
<td>Multics: h</td>
<td>Here's a character for you.</td>
</tr>
<tr>
<td>Multics: e</td>
<td>Here's a character for you.</td>
</tr>
<tr>
<td>Multics: o</td>
<td>Here's a character for you.</td>
</tr>
<tr>
<td>Multics: u</td>
<td>Here's a character for you.</td>
</tr>
<tr>
<td>Terminal: X-OFF</td>
<td>Wait, I can't handle any more for the moment.</td>
</tr>
<tr>
<td>Multics: t</td>
<td>Here's a character for you.</td>
</tr>
</tbody>
</table>

(Silence while Multics waits for OK to resume.)

<table>
<thead>
<tr>
<th>Character</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terminal: X-ON</td>
<td>OK, I can handle some more now.</td>
</tr>
<tr>
<td>Multics: p</td>
<td>Here's a character for you.</td>
</tr>
<tr>
<td>Multics: u</td>
<td>Here's a character for you.</td>
</tr>
</tbody>
</table>
Output Flow Control
Extracts From Terminal Type File

output_suspend: ^S;
output_resume: ^Q;
output_end_of_block: ETX;
output_acknowledge: ACK;
buffer_size: 256;

Extract from flow_control_info.incl.pl1
Used for get_ofc_info and output_flow_control_chars control orders

dcl 1 output_flow_control_info aligned based,
2 flags unaligned,
  3 suspend_resume bit (1),
  3 block_acknowledge bit (1),
  3 mbz bit (16),
2 buffer_size fixed bin (18) unsigned unaligned,
2 suspend_or_etb_seq unaligned,
  3 count fixed bin (9) unsigned,
  3 chars char (3),
2 resume_or_ack_seq unaligned,
  3 count fixed bin (9) unsigned,
  3 chars char (3);

Advanced MCS User Features
Input Flow Control

Multics normally expects asynchronous lines to send data slowly

At typing speed

Some terminals and microcomputers can send input to Multics at line speed

Multics cannot always keep up

For synchronous lines, this is handled by protocol

For asynchronous lines, several modes exist to solve the problem

These solutions are not valid for X.25

Flow control must be between PAD and terminal

No X.25 parameters for this problem

All flow control for asynchronous lines is handled in FNP

blk_xfer mode

For some terminals that send a screenful of data

Data must be bracketed by special characters

If read with a get_chars operation, whole screenful is returned

Causes bigger input buffers to be allocated
Block Transfer
Extract From Terminal Type File

terminal_type: ...
  .
  .

framing_chars: STX ETX;

Structure for get_framing_chars and set_framing_chars control orders

dcl 1 framing_chars aligned,
   2 frame_begin char(1) unaligned,
   2 frame_end char(1) unaligned;

-----------------------------

| iflow mode

  | Suspend-Resume input flow control

  | Same as used in output flow control

  | 1-second timeout possible

  | Causes bigger input buffers to be allocated
Suspend/Resume Input Flow Control Protocol with Timeout

<table>
<thead>
<tr>
<th>Character</th>
<th>Sender Sent</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terminal: t</td>
<td>Here's a character for you.</td>
<td></td>
</tr>
<tr>
<td>Terminal: h</td>
<td>Here's a character for you.</td>
<td></td>
</tr>
<tr>
<td>Terminal: i</td>
<td>Here's a character for you.</td>
<td></td>
</tr>
<tr>
<td>Terminal: s</td>
<td>Here's a character for you.</td>
<td></td>
</tr>
<tr>
<td>Multics: X-OFF</td>
<td>Wait, I can't handle any more for the moment.</td>
<td></td>
</tr>
</tbody>
</table>

(Silence while Multics waits for OK to resume.)

<table>
<thead>
<tr>
<th>Character</th>
<th>Sender Sent</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multics: X-ON</td>
<td>OK, I can handle some more now.</td>
<td></td>
</tr>
<tr>
<td>Terminal: t</td>
<td>Here's a character for you.</td>
<td></td>
</tr>
<tr>
<td>Terminal: h</td>
<td>Here's a character for you.</td>
<td></td>
</tr>
<tr>
<td>Terminal: e</td>
<td>Here's a character for you.</td>
<td></td>
</tr>
<tr>
<td>Terminal: i</td>
<td>Here's a character for you.</td>
<td></td>
</tr>
<tr>
<td>Terminal: n</td>
<td>Here's a character for you.</td>
<td></td>
</tr>
<tr>
<td>Multics: X-OFF</td>
<td>Wait, I can't handle any more for the moment.</td>
<td></td>
</tr>
<tr>
<td>Terminal: p</td>
<td>Here's a character for you.</td>
<td></td>
</tr>
</tbody>
</table>

(Silence while Multics waits for OK to resume.)

<table>
<thead>
<tr>
<th>Character</th>
<th>Sender Sent</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multics: X-ON</td>
<td>OK, I can handle some more now.</td>
<td></td>
</tr>
</tbody>
</table>

(Silence for one second)

<table>
<thead>
<tr>
<th>Character</th>
<th>Sender Sent</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multics: X-ON</td>
<td>I repeat, I can handle some more now.</td>
<td></td>
</tr>
<tr>
<td>Terminal: u</td>
<td>Here's a character for you.</td>
<td></td>
</tr>
<tr>
<td>Terminal: t</td>
<td>Here's a character for you.</td>
<td></td>
</tr>
</tbody>
</table>
Input Flow Control  
Extract From Terminal Type File

terminal_type: ...
  .
input_suspend: ^S;
input_resume: ^Q;

Extract from flow_control_info.incl.pll  
Used for get_ifc_info and input_flow_control_chars control orders

dcl 1 input_flow_control_info aligned based,
  2 suspend_seq unaligned,
    3 count fixed bin (9) unsigned,
    3 chars char (3),
  2 resume_seq unaligned,
    3 count fixed bin (9) unsigned,
    3 chars char (3),
  2 timeout bit (1);
| Parity
| Normally send even parity, ignore input parity and strip input parity bits
| All parity is handled in FNP
| 8bit mode
| Affects input parity
| Inhibits stripping of parity bit
| Treat parity bit as 8th data bit and receive any 8-bit sequence
| oddp mode
| Affects output parity
| Generates odd parity instead of even
| no_outp mode
| Affects output parity
| Generates no parity bits
| If used with rawo any 8-bit sequence can be sent

| Pseudo-modes
| default
| erkl, can, ^rawi, ^rawo, ^wake_tbl, esc
| init
| All modes off, except 1150
| force
| Hide errors in following modes
| force, replay, polite works on async and X.25 lines

| Miscellaneous Orders
| hangup
| Break connection with terminal
| interrupt
| Send break to terminal

| get_channel_info
| Can get devx to use in direct hcs_calls

| start
| Sends an extra wakeup
| Should be used by programs that wakeup and do terminal I/O

| terminal_info
| Get information such as line speed

| write_status
| Find if any buffers waiting to be sent to terminal

| read_status
| Find if any input buffers waiting to be read

| abort
| Flush input and output buffers

| resetread
| Flush input buffers

| resetwrite
| Flush output buffers

| get_event_channel
| Find out event channel for wakeups
| Useful if need to wait on more than one channel

| set_event_channel
| Change event channel used
| Useful if want to use event-call channel instead of event-wait channel
tty_attach description

| tty_ (device) (-control_args)

| tty_LINE

| Attach LINE as slave line

| LINE can be generic destination

| tty_LINE -destination DESTINATION

| Attach LINE as autocall line with specified phone number or network address

| LINE can be generic destination

| tty_ -dial_id STR

| Wait for 'dial STR' on a login line and attach

| tty_ -login_channel

| Attach login channel without knowing name

| -resource STR

| Ask for line of a certain speed

| -no_hangup_on_detach/-hangup_on_detach

| Default is -hangup_on_detach

| -suppress_dial_manager

| Use if process already has line owned or on loan

| Used by Initializer

| Used for login channel

| Used if dial_manager_calls have already been made

| -no_block

| Never go blocked on input or output
Databases

tty_buf

Wired, contains most wired data for Ring 0 MCS

All information used during interrupt handling must be wired

Buffers

WTCBs wired Terminal control block

LCT Logical channel table

Size is specified with PARM TTYB card in config deck

Default is 6K

Too small for average configuration

Size requirement can be calculated

CC75 Appendix C

Meters can show utilization most practical

Header contains meters and pointers to other data within tty_buf

LCT

Logical Channel Table

In tty_buf

One entry (LCTE) for each entry in CDT

For each FNP, multiplexer, non-multiplexed channel

LCTE has pointer to data for FNP, multiplexer, or non-multiplexed channel

WTCB or fnp_info or multiplexer-specific data

For FNP or mpx, this data contains pointers to descendant LCTEs

LCTE has pointer to LCTE of its parent multiplexer

Index of LCTE in LCT is called devx
devx is what is used in hcs calls to identify line

LCNT
Logical Channel Name Table
Parallel to LCT
In tty_area, a paged supervisor segment
Each entry has names of channel
 Allows conversion of devx into channel name
 Used for error messages

WTCB
Wired Terminal Control Block
Primary database for non-multiplexed channel
Database pointer in LCTE of non-multiplexed channel points to it
In tty_buf
Data in WTCB used during interrupts, so it must be wired
Event channel for waking up process when input arrives
Pointers to active input and output buffer chains
Contains pointer to TCB
TCB contains data about channel that is not used at interrupt time

TCB
Terminal Control Block
In tty_area, a paged supervisor segment
Has pointers to conversion and translation tables
Current and default
Bits to indicate modes set

tty_tables
A paged supervisor segment

Databases
Contains tables used during input/output conversion and translation

- Input translation
- Output translation
- Input conversion
- Output conversion
- Special
- Delay

Tables are shared among lines (and hence among processes)

Often many lines use same tables

Reference count tells when table can be freed

fnp_info

- In dn355_data, a wired supervisor segment
- Pointed to by LCTE for FNP
- Contains pointer to mailbox area for FNP
- Contains pointer to PCB array for FNP

PCBs

- Physical Channel Blocks
- In tty_buf
- Array of PCBs is pointed to by fnp_info
- Indexed by channel number
- Has pointer to LCTE of subchannel
- Pointers to I/O buffer chains for subchannel

dn355_mailboxes

- Contains mailboxes for each FNP
- Used in FNP<>Multics communication
Programs

tty_write

- Called via hcs$_tty_write
- Copies user’s output from user ring buffer into buffer in tty_buf
- May accept only part of user’s output
- User ring program must block, await wakeup and call again
- Performs output conversion and translation
- Calls next level of programs (usually fnp_multiplexer) to send data on to FNP

tty_index

- Called via hcs$_tty_index, hcs$_tty_order
- Handles control orders and mode changes
- Pass on to FNP (via fnp_multiplexer) if order or mode is handled in FNP

fnp_multiplexer

- Formats a mailbox and gives it to dn355

dn355

- Manages DIA
- Sends interrupts to FNP
- Receives interrupts from FNP
- Exchanges information with FNP using mailboxes

tty_interrupt

- Called by dn355 for interrupts from FNP-
- Tells FNP where to put input data
- Allocates input buffers
- Sends wakeup to process blocked for input when input is available
- Touches only wired data, i.e. LCT, WTCB, buffers

tty_read
Called via hcs_tty_read

Usually called after wakeup is received

Performs input conversion and translation

Copies converted data from tty_buf into user ring buffer

tty_space_man

Utility to handle all allocation/freeing of buffers in tty_buf
Multiplexing

Connections just described between
dn355 and tty_interrupt on interrupt side
tty_read/write/index and fnp_multiplexer on call side
are not always direct
For multiplexed lines, some intermediate processing is needed
Adding control information to blocks of data to be transmitted
Interpreting and removing control information from blocks of data received
Multiplexing is generalized using cmtv and channel_manager
cmtv
Channel Manager Transfer Vector
Is indexed by
Type of line (type of LCTE)
FNP, X.25 mpx, HASP mpx, non-multiplexed line, etc.
Type of operation
Read, write, interrupt, control, etc.
channel_manager
Is called by each program that needs to pass data up or down a level
For instance
By tty_write to send data from user down another level towards the FNP
By dn355 to send data from FNP up another level toward user
There are different entrypoints in channel_manager for the different types of operation
channel_manager looks in LCTE to get type of line
Uses type of operation and type of line to look in cmtv to find program to call
There is also priv_channel_manager, called for privileged operations such as loading an FNP
LCTE Chaining for Multiplexing
Non-multiplexed subchannel

Interrupt-side
Operations

\[
\text{dn355$interrupt} \quad \text{cmtv(write):} \quad \text{LCTE} \quad \text{fnp_info}
\]

\[
\text{cmtv(lcte.}) \quad \text{lcte.} \quad \text{lcte.} \quad \text{lcte.} \quad \text{pcb_array(n).devx}
\]

\[
\text{cmtv(lcte.}) \quad \text{data_base_ptr} \quad \text{maj_devx}
\]

\[
\text{cmtv(lcte.}) \quad \text{lcte.} \quad \text{WTxCB}
\]

\[
\text{cmtv(lcte.}) \quad \text{data_base_ptr}
\]

Call-side
Operations

MCS Ring 0 Internals
LCTE Chaining for Multiplexing

X.25 Subchannel

Interrupt-side
Operations

\[\text{dn355$interrupt} \quad \text{lcte.} \quad \text{fnp_info} \]

\[\text{cmtv(write):} \quad \text{fnp_mpx$write} \quad \text{lcte.} \quad \text{data_base_ptr} \]

\[\text{cmtv(interrupt)} \quad \text{LCTE} \quad \text{"a"} \quad \text{channel_type} \]

\[\text{dn355$interrupt} \quad \text{fnp_info} \]

\[\text{cmtv(write):} \quad \text{x25_mpx$write} \quad \text{lcte.} \quad \text{x25_info} \]

\[\text{cmtv(interrupt)} \quad \text{LCTE} \quad \text{"a.hl120"} \quad \text{channel_type} \]

\[\text{tty_write} \quad \text{data_base_ptr} \]

\[\text{cmtv(write):} \quad \text{tty_write} \quad \text{lcte.} \quad \text{WTCB} \]

\[\text{cmtv(interrupt)} \quad \text{LCTE} \quad \text{"a.hl140"} \quad \text{channel_type} \]

\[\text{tty_interrupt} \quad \text{data_base_ptr} \]

\[\text{Call-side}
Operations\]

MCS Ring 0 Internals
Multics-FNP Interface

DIA Interrupts

Submailboxes

TIMW

All interrupts to MULT at level 2
It looks at TIMW to find corresponding word
mailbox on MULTICS to find what mailbox to go to
All work in the FNP is driven by interrupts

HSLA
- Input from a line
- Status change for a line

DIA
- Mailbox interrupt from Multics
- Terminate interrupt from I/O started by FNP

Timer

No work ever seems to be done directly

Everything that needs to be done is put in a queue to be done later as time permits
FNP WORK SCHEDULING

MASTER DISPATCHER--INTERRUPT-DRIVEN

INTERRUPT

+---------+---------+
| INTERRUPT VECTOR | +---------+---------+
|                  |            |
|                  |            |
|                  |            |
| 3-WORD JUMP TABLE |            |
|                  |            |
|                  |            |
+---------+---------+

+---------+---------+
| SKED$INV | ||    |
|          |          |
|          |          |
|          |          |
+---------+---------+

+---------+---------+
| master  | DISPATCHER |
| STAMES  | machine conditions |
|          |          |
|          |          |
+---------+---------+

DIA MANAGER

+---------+---------+---------+---------+
| DIA_MAN$TERM | DIA_MAN$MAIL | HSLA_MAN$HINTR | SECONDARY DISPATCHER |
+---------+---------+---------+---------+

++--------+
| MAILBOX | TIMER | SOFTWARE | HARDWARE |
| QUEUE   | QUEUE | STATUS   | STATUS   |
|         |       | QUEUE    | QUEUE    |
| ** list of** | ** work deferred ** | * for line * | * for line * |
| ** jobs not ready ** | ** until later ** | * for line * | * for line * |
++--------+

Programs

FNP Internals

F86 Page 6-2
Three-Word Jump Table

Interrupt Vector | tsy indirect | Handler
0: addr(handler) | -------------> | IC at intp time
1: addr(handler) | +-------------> | Handler Code
2: addr(handler) | +-------------> | ic/oc/devid/mod
3: addr(handler) | +-------------> | IC at intp time
4: addr(handler) | +-------------> | ic/oc/devid/mod
5: addr(handler) | +-------------> | IC at intp time
...

Interrupt Vector | tsy indirect | Three-word Jump Table | tsy | Handler
0: addr(handler) | -------------> | IC at intp time | +-------------> | IC from 3wjt tsy
1: addr(handler) | +-------------> | tsy to handler | +-------------> | Handler Code
2: addr(handler) | +-------------> | ic/oc/devid/mod | +-------------> | ic/oc/devid/mod
3: addr(handler) | +-------------> | IC at intp time | +-------------> | ic/oc/devid/mod
4: addr(handler) | +-------------> | tsy to handler | +-------------> | ic/oc/devid/mod
5: addr(handler) | +-------------> | ic/oc/devid/mod | +-------------> | ic/oc/devid/mod
...

Devid

Line number on HSLA

Programs
scheduler (sked)

Schedules work to be done

sked$invp is the primary interrupt handler.
sked$mdisp is the master dispatcher
sked$dspqur schedules work for later handling
The secondary dispatcher runs work queued by dspqur

hsla man (hsla)

Controls HSLAs
hsla$hintr handles HSLA interrupts
hsla$hstprc does work queued at interrupt time
hsla$hdcw accepts output commands from control tables
hsla$hcfg handles configuration changes
hsla$hmode handles mode changes
hsla$hgeti handles replay and polite modes

dia man (dia)

Controls the DIA
dia$dterm handles DIA terminate interrupts
dia$dmail handles DIA mailbox interrupts
dia$dgetwk does work queued at interrupt time
dia$denq accepts data to be sent to CS

interpreter (intp)

Control Tables Interpreter
control_tables, etc.
Implement the communications protocols

utilities (util)
Utility subroutines

mclt
| COLTS hardware tests
| trace (trac)
| Puts trace information in trace table
| meters (metr)
| Collect meters for channel_comm_meters
| init
| FNP initialization program
Databases

<table>
<thead>
<tr>
<th>Communications Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global database for FNP</td>
</tr>
<tr>
<td>Pointers to other databases in FNP</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IOM Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Used during initialization and to refind HSLA Table to find TIB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TIB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terminal Information Block</td>
</tr>
<tr>
<td>Main per-channel database</td>
</tr>
<tr>
<td>Contains data used by many programs</td>
</tr>
<tr>
<td>Stored in extended memory</td>
</tr>
<tr>
<td>TIB, SFCM and two buffers permanently allocated to channel are stored in a 256-word page</td>
</tr>
<tr>
<td>Page is addressed using page table entry 4177 (addresses 77400-77777)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SFCM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Communications Region</td>
</tr>
<tr>
<td>Per-channel</td>
</tr>
<tr>
<td>Contains information for HSLA for channel</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HWCM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware Communications Region</td>
</tr>
<tr>
<td>Per-channel</td>
</tr>
<tr>
<td>Contains control words for HSLA</td>
</tr>
<tr>
<td>Pointer to CCT for channel</td>
</tr>
<tr>
<td>CCTs are shared, similar to tables in tty_tables</td>
</tr>
<tr>
<td>Pointer to SFCM for channel</td>
</tr>
<tr>
<td>Stored in low-order memory, address known by hardware</td>
</tr>
</tbody>
</table>

| HSLA Table |
Pointed to by IOM table
One two-word entry per channel
Entry contains pointer to TIB
Finding the TIB for a given channel is its main function

TIB Table
One 2-word entry per channel
Not arranged in any particular order
Each entry contains a pointer to TIB and a pointer to outstanding DIA requests for channel
DIA Requests are stored in low-order memory
TIB Table plus these DIA requests make up the DIA Queue

Buffers
Are mainly allocated in extended memory
Addressed using page table entry 4176 (addresses 77000-77377)
Free space chained from .crnxe
If no free space in extended memory, can use free space in low-order memory
Normally used for CCTs, DIA requests, etc.
Free space in low-order memory is chained from .crnxa

Buffer Requirements
Can be calculated using CC75, Appendix B
There are meters to show utilization
FNP Databases Excluding I/O Buffers
(Per-Channel Databases Shown for a.h205)

640->
Comm Region .crnxa-> Free Space --> Free Space --> Free Space
 .crnxs-> Free Space --> Free Space --> Free Space
++.crttb-.criom----------> IOM Table +----> HSLA Tbl
 .crcct+ +14---> ----> +16---> ----> +20---> ----> +----> HSLA Tbl

1000-->
HWCM HSLA 0

2000-->
HWCM HSLA 1

3000-->
HWCM HSLA 2

3000-->
.h.baw---------------------+
+5*16
"h.sfcm++

++-----> SFCM <--------+
 "sf.tib*-

++-----> TIB <--------+
 \.sfcm*--

++-----> TIB Table <--------+

V

DIA Request

DIA Request

DIA Request

Databases

FNP Internals

F86 Page 6-9
FNP MEMORY LAYOUT

<table>
<thead>
<tr>
<th>Start</th>
<th>Length</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>420</td>
<td>Interrupt Vectors</td>
</tr>
<tr>
<td>420</td>
<td>20</td>
<td>Fault Status Words</td>
</tr>
<tr>
<td>440</td>
<td>20</td>
<td>Fault Vector</td>
</tr>
<tr>
<td>450</td>
<td>20</td>
<td>Mailboxes</td>
</tr>
<tr>
<td>475</td>
<td>1</td>
<td>Page Table Address</td>
</tr>
<tr>
<td>500</td>
<td>140</td>
<td>LSLA HWCMs</td>
</tr>
<tr>
<td>640</td>
<td>136</td>
<td>Communication Region</td>
</tr>
<tr>
<td>776</td>
<td>2</td>
<td>Missing Module Code</td>
</tr>
<tr>
<td>1000</td>
<td>#hsla*1000</td>
<td>HSLA HWCMs</td>
</tr>
<tr>
<td>.crcpt</td>
<td>200</td>
<td>CPU Page Table</td>
</tr>
<tr>
<td>.criom</td>
<td>40</td>
<td>IOM Table</td>
</tr>
<tr>
<td>.criom+15,*</td>
<td>#hsla*100</td>
<td>HSLA Tables</td>
</tr>
<tr>
<td>.crmod-22</td>
<td>variable</td>
<td>Program modules</td>
</tr>
<tr>
<td>.crttb</td>
<td>#lines*2</td>
<td>TIB Table</td>
</tr>
<tr>
<td>.crbuf</td>
<td>variable</td>
<td>Low memory buffer pool (for TIB extensions, CCTs, DIA queue, delay tables)</td>
</tr>
<tr>
<td>(=.crtte)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>77000</td>
<td>77777</td>
<td>Unused Pages</td>
</tr>
<tr>
<td>100000</td>
<td>#lines*400</td>
<td>SFCM/TIB pairs, I/O Buffer pool</td>
</tr>
<tr>
<td>100000+</td>
<td>variable</td>
<td>I/O Buffer pool</td>
</tr>
<tr>
<td>#lines*400</td>
<td></td>
<td></td>
</tr>
<tr>
<td>.crtrb</td>
<td>.crmem-</td>
<td>Trace Buffer</td>
</tr>
<tr>
<td>.crtrb</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
display_fnp_symbols: proc;
/* Program to list all the symbols which are understood by debug_fnp */
/* Coded 19 August 1980 by Jim Homan */
dcl i fixed bin;
dcl ioa_entry options (variable);
%page;
%include debug_fnp_data;
%page;
symbol_tablep = addr (db_fnp_symbols$db_fnp_symbols_);

call ioa_ ("Symbol Value Length Explanation^/");

do i = 1 to symbol_table.cnt;
   expextxp = addrel (symbol_tablep, symbol_table.entry (i).explain);
   call ioa_ ("^10a^10o^10d ^a", symbol_table.entry (i).name,
      symbol_table.entry (i).value, symbol_table.entry (i).len,
      expext.data);
end;
end display_fnp_symbols;
### Communications Region

<table>
<thead>
<tr>
<th>Name</th>
<th>Address</th>
<th>Length</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>.crldt</td>
<td>640</td>
<td>4</td>
<td>date and time of binding</td>
</tr>
<tr>
<td>.crbdt</td>
<td>644</td>
<td>4</td>
<td>date and time of bootloading</td>
</tr>
<tr>
<td>.crbuf</td>
<td>650</td>
<td>1</td>
<td>starting address of buffer area</td>
</tr>
<tr>
<td>.crmem</td>
<td>651</td>
<td>1</td>
<td>last location of memory</td>
</tr>
<tr>
<td>.crnbf</td>
<td>652</td>
<td>1</td>
<td>number of buffers available</td>
</tr>
<tr>
<td>.criom</td>
<td>653</td>
<td>1</td>
<td>start of iom table</td>
</tr>
<tr>
<td>.crnhs</td>
<td>654</td>
<td>1</td>
<td>number of hsla's configured</td>
</tr>
<tr>
<td>.crnls</td>
<td>655</td>
<td>1</td>
<td>number of lsla's configured</td>
</tr>
<tr>
<td>.crlcon</td>
<td>656</td>
<td>1</td>
<td>console enabled flag</td>
</tr>
<tr>
<td>.crmod</td>
<td>657</td>
<td>1</td>
<td>starting address of module chain</td>
</tr>
<tr>
<td>.crnxa</td>
<td>660</td>
<td>1</td>
<td>ptr to next available buffer</td>
</tr>
<tr>
<td>.crtra</td>
<td>661</td>
<td>1</td>
<td>trace entry enable mask</td>
</tr>
<tr>
<td>.crtrb</td>
<td>662</td>
<td>1</td>
<td>base address of trace table</td>
</tr>
<tr>
<td>.crtrc</td>
<td>663</td>
<td>1</td>
<td>next available location in trace table</td>
</tr>
<tr>
<td>.crreg</td>
<td>664</td>
<td>1</td>
<td>disaster fault register storage location</td>
</tr>
<tr>
<td>.crttb</td>
<td>665</td>
<td>1</td>
<td>location of tib table</td>
</tr>
<tr>
<td>.crtte</td>
<td>666</td>
<td>1</td>
<td>location of end of tib table</td>
</tr>
<tr>
<td>.crly</td>
<td>667</td>
<td>1</td>
<td>head of delay table chain</td>
</tr>
<tr>
<td>.crver</td>
<td>670</td>
<td>2</td>
<td>mcs version number, 4 chars</td>
</tr>
<tr>
<td>.crbrk</td>
<td>672</td>
<td>1</td>
<td>addr of breakpoint control table</td>
</tr>
<tr>
<td>.crtsw</td>
<td>673</td>
<td>1</td>
<td>if non-zero, tracing will cease</td>
</tr>
<tr>
<td>.crtxs</td>
<td>674</td>
<td>1</td>
<td>next free small block</td>
</tr>
<tr>
<td>.crnbs</td>
<td>675</td>
<td>1</td>
<td>number of buffers devoted to small space</td>
</tr>
<tr>
<td>.crtcct</td>
<td>676</td>
<td>1</td>
<td>address of first cct descriptor</td>
</tr>
<tr>
<td>.crskd</td>
<td>677</td>
<td>1</td>
<td>address of scheduler data block</td>
</tr>
<tr>
<td>.cretb</td>
<td>700</td>
<td>1</td>
<td>list of echo-negotiation bit tables</td>
</tr>
<tr>
<td>.crctpt</td>
<td>701</td>
<td>1</td>
<td>address of cpu page table</td>
</tr>
<tr>
<td>.crpted</td>
<td>702</td>
<td>1</td>
<td>address of variable cpu page table entry</td>
</tr>
<tr>
<td>.crtsz</td>
<td>703</td>
<td>1</td>
<td>size of trace data buffer</td>
</tr>
<tr>
<td>.crmet</td>
<td>704</td>
<td>1</td>
<td>non-zero if metering enabled</td>
</tr>
<tr>
<td>.crtddt</td>
<td>705</td>
<td>1</td>
<td>address of tib for t&amp;d executive channel</td>
</tr>
<tr>
<td>.crbtm</td>
<td>706</td>
<td>1</td>
<td>address of time meters for getbuf/frebuf</td>
</tr>
<tr>
<td>.crnxm</td>
<td>707</td>
<td>1</td>
<td>next available space in extended memory</td>
</tr>
<tr>
<td>.crbpe</td>
<td>710</td>
<td>1</td>
<td>buffer paging window table entry</td>
</tr>
<tr>
<td>.crcpr</td>
<td>760</td>
<td>14</td>
<td>copyright notice</td>
</tr>
<tr>
<td>Name</td>
<td>Address</td>
<td>Length</td>
<td>Explanation</td>
</tr>
<tr>
<td>-------</td>
<td>---------</td>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>t.stat</td>
<td>0</td>
<td>1</td>
<td>holds current line status</td>
</tr>
<tr>
<td>t.flg</td>
<td>1</td>
<td>1</td>
<td>flag word</td>
</tr>
<tr>
<td>t.flg2</td>
<td>2</td>
<td>1</td>
<td>second word of flags</td>
</tr>
<tr>
<td>t.cur</td>
<td>3</td>
<td>1</td>
<td>current address in control table</td>
</tr>
<tr>
<td>t.line</td>
<td>4</td>
<td>1</td>
<td>10 bit line number</td>
</tr>
<tr>
<td>t.icp</td>
<td>5</td>
<td>1</td>
<td>first buffer in input chain</td>
</tr>
<tr>
<td>t.iplst</td>
<td>6</td>
<td>1</td>
<td>last buffer in input chain</td>
</tr>
<tr>
<td>t.icpl</td>
<td>7</td>
<td>1</td>
<td>count of buffers in icp chain</td>
</tr>
<tr>
<td>t.icch</td>
<td>10</td>
<td>1</td>
<td>address of next input character</td>
</tr>
<tr>
<td>t.elnk</td>
<td>11</td>
<td>1</td>
<td>link to tib extension</td>
</tr>
<tr>
<td>t.rcp</td>
<td>11</td>
<td>1</td>
<td>replay chain pointer (share t.elnk)</td>
</tr>
<tr>
<td>t.ocp</td>
<td>12</td>
<td>1</td>
<td>addr of output chain from cs</td>
</tr>
<tr>
<td>t.ocur</td>
<td>13</td>
<td>1</td>
<td>addr of current output buffer</td>
</tr>
<tr>
<td>t.olst</td>
<td>14</td>
<td>1</td>
<td>addr of last buffer in output chain</td>
</tr>
<tr>
<td>t.occh</td>
<td>15</td>
<td>1</td>
<td>addr of next output character</td>
</tr>
<tr>
<td>t.ocnt</td>
<td>16</td>
<td>1</td>
<td>count of buffers in t.ocur</td>
</tr>
<tr>
<td>t.type</td>
<td>17</td>
<td>1</td>
<td>line type</td>
</tr>
<tr>
<td>t.time</td>
<td>20</td>
<td>2</td>
<td>time at which next timeout will occur</td>
</tr>
<tr>
<td>t.reta</td>
<td>22</td>
<td>1</td>
<td>return address from calsub</td>
</tr>
<tr>
<td>t.dcwa</td>
<td>23</td>
<td>1</td>
<td>addr of dcw list to 'execute'</td>
</tr>
<tr>
<td>t.dcw</td>
<td>24</td>
<td>1</td>
<td>length of dcw list</td>
</tr>
<tr>
<td>t.echo</td>
<td>25</td>
<td>1</td>
<td>echo buffer address</td>
</tr>
<tr>
<td>t.dcp</td>
<td>26</td>
<td>1</td>
<td>addr of message chain for cs</td>
</tr>
<tr>
<td>t.dlst</td>
<td>27</td>
<td>1</td>
<td>last buffer in message chain for cs</td>
</tr>
<tr>
<td>t.ftse</td>
<td>30</td>
<td>1</td>
<td>first time slot entry in lsla table</td>
</tr>
<tr>
<td>t.sfcm</td>
<td>30</td>
<td>1</td>
<td>pointer to sfcm for hsla lines</td>
</tr>
<tr>
<td>t.bcnt</td>
<td>31</td>
<td>1</td>
<td>counting temporary</td>
</tr>
<tr>
<td>t.brkp</td>
<td>32</td>
<td>1</td>
<td>pointer to current break list</td>
</tr>
<tr>
<td>t.pos</td>
<td>33</td>
<td>1</td>
<td>current carriage position</td>
</tr>
<tr>
<td>t.char</td>
<td>34</td>
<td>1</td>
<td>pending line control char (lsla only)</td>
</tr>
<tr>
<td>t.ecch</td>
<td>34</td>
<td>1</td>
<td>address of current echo char (hsla only)</td>
</tr>
<tr>
<td>t.cntr</td>
<td>35</td>
<td>1</td>
<td>counter for control tables</td>
</tr>
<tr>
<td>t.flg3</td>
<td>36</td>
<td>1</td>
<td>third word of flags</td>
</tr>
<tr>
<td>t.dtp</td>
<td>37</td>
<td>1</td>
<td>pointer to delay table for this line</td>
</tr>
<tr>
<td>t.frmc</td>
<td>40</td>
<td>1</td>
<td>framing chars (2 chars)</td>
</tr>
<tr>
<td>t.dcpl</td>
<td>41</td>
<td>1</td>
<td>number of buffers in dcp chain</td>
</tr>
<tr>
<td>t.sc11</td>
<td>42</td>
<td>1</td>
<td>screenline length left, for echo neg.</td>
</tr>
<tr>
<td>t.sncc</td>
<td>43</td>
<td>1</td>
<td>Echo negotiation sync ctr.</td>
</tr>
<tr>
<td>t.entr</td>
<td>44</td>
<td>1</td>
<td>Echo negotiation break table ptr</td>
</tr>
<tr>
<td>t.ifch</td>
<td>45</td>
<td>1</td>
<td>input flow control characters</td>
</tr>
<tr>
<td>t.ofch</td>
<td>46</td>
<td>1</td>
<td>output flow control characters</td>
</tr>
<tr>
<td>t.omct</td>
<td>47</td>
<td>1</td>
<td>output message count (for flow control)</td>
</tr>
<tr>
<td>t.itim</td>
<td>50</td>
<td>2</td>
<td>time of last call to inproc (2 words)</td>
</tr>
<tr>
<td>t.metr</td>
<td>52</td>
<td>1</td>
<td>address of metering area</td>
</tr>
<tr>
<td>t.abfo</td>
<td>53</td>
<td>1</td>
<td>absolute address of first permanent buffer</td>
</tr>
<tr>
<td>t.abfl</td>
<td>54</td>
<td>1</td>
<td>absolute address of first permanent buffer</td>
</tr>
</tbody>
</table>
### Flags for tib.flg

<table>
<thead>
<tr>
<th>Name</th>
<th>Address</th>
<th>Length</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>tfosus</td>
<td>-400000</td>
<td>1</td>
<td>output suspended</td>
</tr>
<tr>
<td>tfecpx</td>
<td>200000</td>
<td>1</td>
<td>echoplex mode</td>
</tr>
<tr>
<td>tfisus</td>
<td>100000</td>
<td>1</td>
<td>input suspended</td>
</tr>
<tr>
<td>tfseotx</td>
<td>40000</td>
<td>1</td>
<td>eot expected from 2741</td>
</tr>
<tr>
<td>tfauto</td>
<td>20000</td>
<td>1</td>
<td>this is hs1a autobaud line</td>
</tr>
<tr>
<td>tfwabt</td>
<td>10000</td>
<td>1</td>
<td>do write abort</td>
</tr>
<tr>
<td>tfbec</td>
<td>4000</td>
<td>1</td>
<td>do space echo on tab</td>
</tr>
<tr>
<td>tfcrec</td>
<td>2000</td>
<td>1</td>
<td>do cr echo on lf</td>
</tr>
<tr>
<td>tfifec</td>
<td>1000</td>
<td>1</td>
<td>do lf echo on cr</td>
</tr>
<tr>
<td>tfctrl</td>
<td>400</td>
<td>1</td>
<td>do kybd/printer addressing</td>
</tr>
<tr>
<td>tfgquit</td>
<td>200</td>
<td>1</td>
<td>send nl on line break</td>
</tr>
<tr>
<td>tflsn</td>
<td>100</td>
<td>1</td>
<td>answer the phone</td>
</tr>
<tr>
<td>tfhang</td>
<td>40</td>
<td>1</td>
<td>hangup this line</td>
</tr>
<tr>
<td>tfpotr</td>
<td>20</td>
<td>1</td>
<td>printer addressed</td>
</tr>
<tr>
<td>tfkybd</td>
<td>10</td>
<td>1</td>
<td>keyboard addressed</td>
</tr>
<tr>
<td>tffdpx</td>
<td>4</td>
<td>1</td>
<td>full duplex mode</td>
</tr>
<tr>
<td>tfbel</td>
<td>2</td>
<td>1</td>
<td>send bell as next echoed character</td>
</tr>
<tr>
<td>tfwrit</td>
<td>1</td>
<td>1</td>
<td>output chain present</td>
</tr>
<tr>
<td>tfdlup</td>
<td>20</td>
<td>1</td>
<td>line is on dialup modem</td>
</tr>
</tbody>
</table>

### Flags for tib.flg2

<table>
<thead>
<tr>
<th>Name</th>
<th>Address</th>
<th>Length</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>tfpfnl</td>
<td>-400000</td>
<td>1</td>
<td>line is in prefixnl mode</td>
</tr>
<tr>
<td>tfsftr</td>
<td>200000</td>
<td>1</td>
<td>terminal is shifting device</td>
</tr>
<tr>
<td>tffip</td>
<td>100000</td>
<td>1</td>
<td>frame in progress</td>
</tr>
<tr>
<td>tffrm1</td>
<td>40000</td>
<td>1</td>
<td>frame mode</td>
</tr>
<tr>
<td>tfmrvc</td>
<td>20000</td>
<td>1</td>
<td>'message' receive mode</td>
</tr>
<tr>
<td>tfcrvc</td>
<td>10000</td>
<td>1</td>
<td>'control' receive mode</td>
</tr>
<tr>
<td>tfblak</td>
<td>4000</td>
<td>1</td>
<td>asynchronous block acknowledgement</td>
</tr>
<tr>
<td>tflbit</td>
<td>2000</td>
<td>1</td>
<td>polite mode</td>
</tr>
<tr>
<td>tfrron</td>
<td>1000</td>
<td>1</td>
<td>replay is occurring now</td>
</tr>
<tr>
<td>tfrrly</td>
<td>400</td>
<td>1</td>
<td>replay mode</td>
</tr>
<tr>
<td>tfupsf</td>
<td>200</td>
<td>1</td>
<td>terminal is upshifted now</td>
</tr>
<tr>
<td>tfocf</td>
<td>100</td>
<td>1</td>
<td>output flow control mode</td>
</tr>
<tr>
<td>tfiffc</td>
<td>40</td>
<td>1</td>
<td>input flow control mode</td>
</tr>
<tr>
<td>tfacu</td>
<td>20</td>
<td>1</td>
<td>do dial out</td>
</tr>
<tr>
<td>tfabort</td>
<td>10</td>
<td>1</td>
<td>do read abort</td>
</tr>
<tr>
<td>tfercv</td>
<td>4</td>
<td>1</td>
<td>enter receive mode</td>
</tr>
<tr>
<td>tfdild</td>
<td>2</td>
<td>1</td>
<td>terminal is dialed up</td>
</tr>
<tr>
<td>tfxhld</td>
<td>1</td>
<td>1</td>
<td>hold transmit on</td>
</tr>
</tbody>
</table>

---

**Databases**

FNP Internals
### Flags for tib.flg3

<table>
<thead>
<tr>
<th>Name</th>
<th>Address</th>
<th>Length</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>tfmask</td>
<td>20000</td>
<td>1</td>
<td>channel has been masked for excessive interrupts</td>
</tr>
<tr>
<td>tfafbl</td>
<td>10000</td>
<td>1</td>
<td>t.afb1 available</td>
</tr>
<tr>
<td>tforp</td>
<td>4000</td>
<td>1</td>
<td>output resume pending (waiting until tro)</td>
</tr>
<tr>
<td>tfsolp</td>
<td>2000</td>
<td>1</td>
<td>suspension of output in progress</td>
</tr>
<tr>
<td>tfafbo</td>
<td>1000</td>
<td>1</td>
<td>t.afb0 available</td>
</tr>
<tr>
<td>tfoddp</td>
<td>400</td>
<td>1</td>
<td>odd parity</td>
</tr>
<tr>
<td>tf8out</td>
<td>200</td>
<td>1</td>
<td>don't strip output parity</td>
</tr>
<tr>
<td>tf8in</td>
<td>100</td>
<td>1</td>
<td>don't strip input parity</td>
</tr>
<tr>
<td>tfsked</td>
<td>40</td>
<td>1</td>
<td>input timeout routine scheduled</td>
</tr>
<tr>
<td>tfbrel</td>
<td>20</td>
<td>1</td>
<td>break on all characters</td>
</tr>
<tr>
<td>tfecho</td>
<td>10</td>
<td>1</td>
<td>echoing has priority over output (lsla)</td>
</tr>
<tr>
<td>tkpar</td>
<td>4</td>
<td>1</td>
<td>keep parity bits'</td>
</tr>
<tr>
<td>tfitim</td>
<td>2</td>
<td>1</td>
<td>timeout if input stops</td>
</tr>
<tr>
<td>tfbkpt</td>
<td>1</td>
<td>1</td>
<td>line is stopped at breakpoint</td>
</tr>
</tbody>
</table>

### Flags for tib.stat

<table>
<thead>
<tr>
<th>Name</th>
<th>Address</th>
<th>Length</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>tsfcrq</td>
<td>20000</td>
<td>1</td>
<td>acu call request</td>
</tr>
<tr>
<td>tsfrts</td>
<td>10000</td>
<td>1</td>
<td>request to send</td>
</tr>
<tr>
<td>tsftrc</td>
<td>4000</td>
<td>1</td>
<td>tally runout enable (hdlc)</td>
</tr>
<tr>
<td>tsfsxt</td>
<td>4000</td>
<td>1</td>
<td>supervisory transmit</td>
</tr>
<tr>
<td>tsfdtr</td>
<td>2000</td>
<td>1</td>
<td>data terminal ready</td>
</tr>
<tr>
<td>tsfbre</td>
<td>1000</td>
<td>1</td>
<td>send line break</td>
</tr>
<tr>
<td>tsfxmt</td>
<td>400</td>
<td>1</td>
<td>transmit mode</td>
</tr>
<tr>
<td>tsfrcv</td>
<td>200</td>
<td>1</td>
<td>receive mode</td>
</tr>
<tr>
<td>tsftrm</td>
<td>100</td>
<td>1</td>
<td>send terminate status</td>
</tr>
<tr>
<td>tsfmrk</td>
<td>40</td>
<td>1</td>
<td>send marker status</td>
</tr>
<tr>
<td>tsfst</td>
<td>20</td>
<td>1</td>
<td>store status</td>
</tr>
<tr>
<td>tsfsrc</td>
<td>10</td>
<td>1</td>
<td>supervisory receive</td>
</tr>
<tr>
<td>tsfdsr</td>
<td>4</td>
<td>1</td>
<td>data set ready</td>
</tr>
<tr>
<td>tsfcst</td>
<td>2</td>
<td>1</td>
<td>clear to send</td>
</tr>
<tr>
<td>tsfcd</td>
<td>1</td>
<td>1</td>
<td>carrier detect</td>
</tr>
</tbody>
</table>

---

**Databases**

FNP Internals

F86 Page 6-16
<table>
<thead>
<tr>
<th>Name</th>
<th>Address</th>
<th>Length</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>sf.hcm</td>
<td>0</td>
<td>1</td>
<td>addr of hwcmand</td>
</tr>
<tr>
<td>sf.nxa</td>
<td>1</td>
<td>1</td>
<td>addr of next available queue entry</td>
</tr>
<tr>
<td>sf.mxp</td>
<td>2</td>
<td>1</td>
<td>addr of next queue entry to process</td>
</tr>
<tr>
<td>sf.tly</td>
<td>3</td>
<td>1</td>
<td>tally of status queue</td>
</tr>
<tr>
<td>sf.tib</td>
<td>4</td>
<td>1</td>
<td>addr of tib for this line</td>
</tr>
<tr>
<td>sf.flg</td>
<td>5</td>
<td>1</td>
<td>flag word</td>
</tr>
<tr>
<td>sf.ib0</td>
<td>6</td>
<td>1</td>
<td>pointer to input buffer 1</td>
</tr>
<tr>
<td>sf.ib1</td>
<td>7</td>
<td>1</td>
<td>pointer to input buffer 2</td>
</tr>
<tr>
<td>sf.oib</td>
<td>10</td>
<td>1</td>
<td>pointer to output buffer 1</td>
</tr>
<tr>
<td>sf.ob1</td>
<td>11</td>
<td>1</td>
<td>pointer to output buffer 2</td>
</tr>
<tr>
<td>sf.pcw</td>
<td>12</td>
<td>1</td>
<td>current pcw 2nd word</td>
</tr>
<tr>
<td>sf.cct</td>
<td>13</td>
<td>1</td>
<td>cct addr for this line, if non-zero</td>
</tr>
<tr>
<td>sf.rct</td>
<td>14</td>
<td>1</td>
<td>repeat count for status queue overflows</td>
</tr>
<tr>
<td>sf.hsl</td>
<td>15</td>
<td>1</td>
<td>address of hsla table entry for this line</td>
</tr>
<tr>
<td>sf.bsz</td>
<td>16</td>
<td>1</td>
<td>max buffer size</td>
</tr>
<tr>
<td>sf.fbs</td>
<td>17</td>
<td>1</td>
<td>buffer size to be used during frame input</td>
</tr>
<tr>
<td>sf.mns</td>
<td>20</td>
<td>1</td>
<td>maximum synchronous message size</td>
</tr>
<tr>
<td>sf.csz</td>
<td>20</td>
<td>1</td>
<td>current asynchronous buffer size</td>
</tr>
<tr>
<td>sf.rms</td>
<td>21</td>
<td>1</td>
<td>remaining unallocated message length</td>
</tr>
<tr>
<td>sf.nic</td>
<td>21</td>
<td>1</td>
<td>char. address of next asynchronous input char</td>
</tr>
<tr>
<td>sf.noc</td>
<td>22</td>
<td>1</td>
<td>char address of next asynchronous output char</td>
</tr>
<tr>
<td>sf.ssl</td>
<td>23</td>
<td>1</td>
<td>number of entries in software status queue</td>
</tr>
<tr>
<td>sf.cfg</td>
<td>24</td>
<td>2</td>
<td>2 words for config pcw</td>
</tr>
<tr>
<td>sf.sta</td>
<td>26</td>
<td>12</td>
<td>hardware status q, sicw here</td>
</tr>
<tr>
<td>sf.waq</td>
<td>42</td>
<td>40</td>
<td>wrap around queue, software status q</td>
</tr>
</tbody>
</table>
Flags for sf.flg

<table>
<thead>
<tr>
<th>Name</th>
<th>Address</th>
<th>Length</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>sffhdl</td>
<td>-400000</td>
<td>1</td>
<td>uses hdlc channel board</td>
</tr>
<tr>
<td>sffssb</td>
<td>200000</td>
<td>1</td>
<td>copied alternate buffer back to output chain</td>
</tr>
<tr>
<td>sffnib</td>
<td>100000</td>
<td>1</td>
<td>need to allocate new input buffer(s)</td>
</tr>
<tr>
<td>sffofr</td>
<td>40000</td>
<td>1</td>
<td>old setting of tffrmi (lines up on tffrmi)</td>
</tr>
<tr>
<td>sffmsp</td>
<td>20000</td>
<td>1</td>
<td>marker status pending</td>
</tr>
<tr>
<td>sffsyn</td>
<td>10000</td>
<td>1</td>
<td>any synchronous line type</td>
</tr>
<tr>
<td>sffsqo</td>
<td>4000</td>
<td>1</td>
<td>status queue overflow pending</td>
</tr>
<tr>
<td>sffbsc</td>
<td>2000</td>
<td>1</td>
<td>binary synchronous device</td>
</tr>
<tr>
<td>sffstp</td>
<td>400</td>
<td>1</td>
<td>stop channel, rxmit done</td>
</tr>
<tr>
<td>sffdct</td>
<td>200</td>
<td>1</td>
<td>dynamic (sharable) cct in use for channel</td>
</tr>
<tr>
<td>sffech</td>
<td>100</td>
<td>1</td>
<td>tab, cr, lf echo going on now</td>
</tr>
<tr>
<td>sffebd</td>
<td>40</td>
<td>1</td>
<td>ebcDIC data code on this line</td>
</tr>
<tr>
<td>sffsct</td>
<td>20</td>
<td>1</td>
<td>short cct flag</td>
</tr>
<tr>
<td>sffisc</td>
<td>10</td>
<td>1</td>
<td>inactive subchannel flag</td>
</tr>
<tr>
<td>sffcoi</td>
<td>4</td>
<td>1</td>
<td>on if alternate output icw is active</td>
</tr>
<tr>
<td>sffcii</td>
<td>2</td>
<td>1</td>
<td>on if alternate input icw is active</td>
</tr>
<tr>
<td>sffskd</td>
<td>1</td>
<td>1</td>
<td>status processor is scheduled</td>
</tr>
</tbody>
</table>
## HWCM

<table>
<thead>
<tr>
<th>Name</th>
<th>Address</th>
<th>Length</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>h.ric0</td>
<td>0</td>
<td>2</td>
<td>primary receive icw</td>
</tr>
<tr>
<td>h.ricl</td>
<td>2</td>
<td>2</td>
<td>alternate receive icw</td>
</tr>
<tr>
<td>h.sic0</td>
<td>4</td>
<td>2</td>
<td>primary send icw</td>
</tr>
<tr>
<td>h.sicl</td>
<td>6</td>
<td>2</td>
<td>alternate send icw</td>
</tr>
<tr>
<td>h.baw</td>
<td>10</td>
<td>1</td>
<td>base address word</td>
</tr>
<tr>
<td>h.sfcm</td>
<td>11</td>
<td>1</td>
<td>software comm. region address</td>
</tr>
<tr>
<td>h.mask</td>
<td>12</td>
<td>2</td>
<td>mask register</td>
</tr>
<tr>
<td>h.aicw</td>
<td>14</td>
<td>2</td>
<td>active status icw</td>
</tr>
<tr>
<td>h.cnfg</td>
<td>16</td>
<td>2</td>
<td>configuration status</td>
</tr>
</tbody>
</table>
## FNP Line Meters

<table>
<thead>
<tr>
<th>Name</th>
<th>Address</th>
<th>Length</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>m.dql</td>
<td>0</td>
<td>2</td>
<td>cumulative length of dia request queue</td>
</tr>
<tr>
<td>m.dqu</td>
<td>2</td>
<td>2</td>
<td>updates of m.dql</td>
</tr>
<tr>
<td>m.nst</td>
<td>4</td>
<td>2</td>
<td>cumulative no. of pending status</td>
</tr>
<tr>
<td>m.nsu</td>
<td>6</td>
<td>2</td>
<td>updates of m.nst</td>
</tr>
<tr>
<td>m.over</td>
<td>10</td>
<td>1</td>
<td>output overlaps</td>
</tr>
<tr>
<td>m.par</td>
<td>11</td>
<td>1</td>
<td>parity errors</td>
</tr>
<tr>
<td>m.ssqo</td>
<td>12</td>
<td>1</td>
<td>software status queue overflows</td>
</tr>
<tr>
<td>m.hsqo</td>
<td>13</td>
<td>1</td>
<td>hardware status queue overflows</td>
</tr>
<tr>
<td>m.inaf</td>
<td>14</td>
<td>1</td>
<td>input allocation failures</td>
</tr>
<tr>
<td>m.cql</td>
<td>15</td>
<td>1</td>
<td>current length of dia request queue</td>
</tr>
<tr>
<td>m.exh</td>
<td>16</td>
<td>2</td>
<td>exhaust status</td>
</tr>
<tr>
<td>m.xte</td>
<td>20</td>
<td>1</td>
<td>software xte status</td>
</tr>
<tr>
<td>m.leng</td>
<td>22</td>
<td>1</td>
<td>length of common meters (must be even)</td>
</tr>
<tr>
<td>m.prex</td>
<td>22</td>
<td>2</td>
<td>pre-exhaust status</td>
</tr>
<tr>
<td>m.ebof</td>
<td>24</td>
<td>2</td>
<td>echo buffer overflows</td>
</tr>
<tr>
<td>m.quit</td>
<td>26</td>
<td>1</td>
<td>bell-quits</td>
</tr>
<tr>
<td>m.asyl</td>
<td>30</td>
<td>1</td>
<td>total length of asynchronous meters</td>
</tr>
<tr>
<td>m.nim</td>
<td>22</td>
<td>2</td>
<td>number of input messages</td>
</tr>
<tr>
<td>m.iml</td>
<td>24</td>
<td>2</td>
<td>cumulative length of input</td>
</tr>
<tr>
<td>m.mini</td>
<td>26</td>
<td>1</td>
<td>minimum length of input message</td>
</tr>
<tr>
<td>m.maxi</td>
<td>27</td>
<td>1</td>
<td>maximum length of input message</td>
</tr>
<tr>
<td>m.nom</td>
<td>30</td>
<td>2</td>
<td>number of output messages</td>
</tr>
<tr>
<td>m.oml</td>
<td>32</td>
<td>2</td>
<td>cumulative length of output</td>
</tr>
<tr>
<td>m.mino</td>
<td>34</td>
<td>1</td>
<td>minimum length of output message</td>
</tr>
<tr>
<td>m.maxo</td>
<td>35</td>
<td>1</td>
<td>maximum length of output message</td>
</tr>
<tr>
<td>m.cnt1</td>
<td>36</td>
<td>2</td>
<td>first type of counter</td>
</tr>
<tr>
<td>m.cnt2</td>
<td>40</td>
<td>2</td>
<td>second type of counter</td>
</tr>
<tr>
<td>m.cnt3</td>
<td>42</td>
<td>2</td>
<td>third type of counter</td>
</tr>
<tr>
<td>m.cnt4</td>
<td>44</td>
<td>2</td>
<td>fourth type of counter</td>
</tr>
<tr>
<td>m.cnt5</td>
<td>46</td>
<td>2</td>
<td>fifth type of counter</td>
</tr>
<tr>
<td>m.cnt6</td>
<td>50</td>
<td>2</td>
<td>sixth type of counter</td>
</tr>
<tr>
<td>m.cnt7</td>
<td>52</td>
<td>2</td>
<td>seventh type of counter</td>
</tr>
<tr>
<td>m.cnt8</td>
<td>54</td>
<td>2</td>
<td>eighth type of counter</td>
</tr>
<tr>
<td>m.synl</td>
<td>56</td>
<td>1</td>
<td>total length of synchronous meters</td>
</tr>
</tbody>
</table>
MCS Metering

- display_cdt
- tty_lines
- system_comm_meters
- channel_comm_meters
- debug_fnp
  - bstat
  - sample_ic
  - fnp_idle
- interrupt_meters
- fnp_throughput
- meter_fnp_idle
- display_fnp_idle

Better Alternatives:
- fnp_idle from debug_fnp
CDTE at 515|2460

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>in_use</td>
<td>1 (hung up)</td>
</tr>
<tr>
<td>name:</td>
<td>a.h006</td>
</tr>
<tr>
<td>Multiplexer data:</td>
<td></td>
</tr>
<tr>
<td>service_type:</td>
<td>1 (active)</td>
</tr>
<tr>
<td>mpx_type:</td>
<td>11 (x25)</td>
</tr>
<tr>
<td>state:</td>
<td>4 (up)</td>
</tr>
<tr>
<td>initial load:</td>
<td>02/09/84 1533.7 hfh Thu</td>
</tr>
<tr>
<td>last load:</td>
<td>02/20/84 1730.2 hfh Mon</td>
</tr>
<tr>
<td>last crash:</td>
<td>02/20/84 1601.2 hfh Mon</td>
</tr>
<tr>
<td>load started:</td>
<td>02/20/84 1730.2 hfh Mon</td>
</tr>
<tr>
<td>flags.go:</td>
<td>1</td>
</tr>
<tr>
<td>flags.listening:</td>
<td>1</td>
</tr>
<tr>
<td>current_service_type:</td>
<td>1 (active)</td>
</tr>
<tr>
<td>current_mpx_type:</td>
<td>11 (x25)</td>
</tr>
<tr>
<td>n_bootloads:</td>
<td>11</td>
</tr>
<tr>
<td>last_tbf:</td>
<td>293</td>
</tr>
<tr>
<td>comment:</td>
<td>X.25 major channel</td>
</tr>
<tr>
<td>charge_type:</td>
<td>0 (none)</td>
</tr>
<tr>
<td>service_type:</td>
<td>8 (multiplexer)</td>
</tr>
<tr>
<td>current_service_type:</td>
<td>8 (multiplexer)</td>
</tr>
<tr>
<td>dim:</td>
<td>1 (tty)</td>
</tr>
<tr>
<td>line_type:</td>
<td>17 (X25LAP)</td>
</tr>
<tr>
<td>terminal_type:</td>
<td>X25_TRANSPAC</td>
</tr>
<tr>
<td>baud_rate:</td>
<td>1200</td>
</tr>
<tr>
<td>fnp_no:</td>
<td>1 (a)</td>
</tr>
<tr>
<td>event:</td>
<td>000330011502407777000673</td>
</tr>
<tr>
<td>tra_vec:</td>
<td>0 ()</td>
</tr>
<tr>
<td>count:</td>
<td>0</td>
</tr>
<tr>
<td>twx:</td>
<td>0</td>
</tr>
<tr>
<td>state:</td>
<td>0 ()</td>
</tr>
<tr>
<td>current_terminal_type:</td>
<td>0 (none)</td>
</tr>
<tr>
<td>cur_line_type:</td>
<td>0 (none)</td>
</tr>
<tr>
<td>tty_id_code:</td>
<td>77777</td>
</tr>
<tr>
<td>process:</td>
<td>never</td>
</tr>
<tr>
<td>next_channel:</td>
<td>0</td>
</tr>
<tr>
<td>n_dialups:</td>
<td>0</td>
</tr>
<tr>
<td>n_logins:</td>
<td>0</td>
</tr>
<tr>
<td>dialed_up_time:</td>
<td>0 hrs 0 mins 0 secs.</td>
</tr>
<tr>
<td>dialup_time:</td>
<td>never</td>
</tr>
<tr>
<td>recent_wakeup_count:</td>
<td>0</td>
</tr>
<tr>
<td>recent_wakeup_time:</td>
<td>never</td>
</tr>
</tbody>
</table>
display_cdt

CDTE at 515|2620

| in_use:       | 1 (hung up)                     |
| name:         | a.h006.d01                      |
| comment:      | X.25 dial_out sub-channel       |
| charge_type:  | 0 (none)                        |
| service_type: | 6 (dial out)                    |
| current_service_type: | 6 (dial out)              |
| dim:          | 1 (tty)                         |
| line_type:    | 0 (none)                        |
| terminal_type:|                                |
| baud_rate:    | 300                             |
| fnp_no:       | 0 ()                            |
| event:        | 000470002046407777000151         |
| tra_vec:      | 18 (wait_slave_request)         |
| count:        | 0                               |
| twx:          | 77                              |
| state:        | 1 (hung up)                     |
| current_terminal_type: | 0 (none)              |
| cur_line_type:| none                            |
| tty_id_code:  | 77777|1                          |
| process:      | 0                               |
| next_channel: | 1649                            |
| n_dialups:    | 1208                            |
| dialed_up_time: | 210 hrs 58 mins 37 secs.        |
| dialup_time:  | 02/20/84 1623.0 hfh Mon          |
| recent_wakeup_count: | 02/20/84 1730.2 hfh Mon          |

MCS Metering
CDTE at 515|3560

in_use: 5 (logged in & proc)
name: a.h006.001
comment: X.25 login sub-channel
charge_type: 0 (none)
service_type: 1 (login)
current_service_type: 1 (login)
dim: 1 (tty)
line_type: 0 (none)
terminal_type: ASCII_CRT
baud_rate: 1200
fnp_no: 0 ()
flags.attributes: dont_read_answerback,check_acs;
event: 000330011356407777000132
tra_vec: 8 (wait_logout_sig)
count: 1
twx: 62
state: 5 (dialed up)
current_terminal_type: ASCII_CAPS
cur_line_type: 1 (ASCII)
tty_id_code: none
process: 366|1330
next_channel: 0
n_dialups: 2788
n_logins: 2180
dialed_up_time: 886 hrs 32 mins 32 secs.
dialup_time: 02/20/84 1801.3 hfh Mon
recent_wakeup_count: 2
recent_wakeup_time: 02/20/84 1801.6 hfh Mon
**interrupt_meters**

Total metering time 0:10:19

<table>
<thead>
<tr>
<th>IOM Ch</th>
<th>Int</th>
<th>Ave Time</th>
<th>% CPU</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>A 6.</td>
<td>1</td>
<td>1.795</td>
<td>0.00</td>
<td>IOM A special</td>
</tr>
<tr>
<td>A 8.</td>
<td>1417</td>
<td>0.552</td>
<td>0.13</td>
<td>dska</td>
</tr>
<tr>
<td>A 9.</td>
<td>10</td>
<td>0.457</td>
<td>0.00</td>
<td>dska</td>
</tr>
<tr>
<td>A 12.</td>
<td>90</td>
<td>0.563</td>
<td>0.01</td>
<td>dskc</td>
</tr>
<tr>
<td>A 14.</td>
<td>174</td>
<td>0.531</td>
<td>0.01</td>
<td>dska</td>
</tr>
<tr>
<td>A 16.</td>
<td>2579</td>
<td>0.545</td>
<td>0.23</td>
<td>dskb</td>
</tr>
<tr>
<td>A 17.</td>
<td>60</td>
<td>0.495</td>
<td>0.00</td>
<td>dskb</td>
</tr>
<tr>
<td>A 18.</td>
<td>324</td>
<td>0.579</td>
<td>0.03</td>
<td>dskd</td>
</tr>
<tr>
<td>A 20.</td>
<td>2825</td>
<td>0.570</td>
<td>0.26</td>
<td>dskd</td>
</tr>
<tr>
<td>A 22.</td>
<td>418</td>
<td>0.481</td>
<td>0.03</td>
<td>dskb</td>
</tr>
<tr>
<td>A 23.</td>
<td>1</td>
<td>0.276</td>
<td>0.00</td>
<td>dskb</td>
</tr>
<tr>
<td>A 26.</td>
<td>9</td>
<td>1.237</td>
<td>0.00</td>
<td>prta</td>
</tr>
<tr>
<td>A 28.</td>
<td>7748</td>
<td>2.732</td>
<td>3.42</td>
<td>fnp a</td>
</tr>
</tbody>
</table>

Chan 15656 1.632 4.13
Ovhd 15605 0.159 0.40
Total 15605 1.796 4.53

3 ½% of CPU for fnp interrupts
Total metering time: 0:10:03

**THROUGHPUT**

<table>
<thead>
<tr>
<th></th>
<th>before conversion</th>
<th>after conversion</th>
<th>ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total characters input</td>
<td>5,041</td>
<td>4,705</td>
<td>0.93</td>
</tr>
<tr>
<td>Total characters output</td>
<td>153,567</td>
<td>123,256</td>
<td>0.80</td>
</tr>
<tr>
<td>Average length of input</td>
<td>5.1 characters</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average length of output</td>
<td>73.3 characters</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input characters preconverted</td>
<td></td>
<td>0 (0.0% of total)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>read</th>
<th>write</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of calls</td>
<td>1,436</td>
<td>2,495</td>
</tr>
<tr>
<td>Average time per call</td>
<td>2.13 msec.</td>
<td>5.09 msec.</td>
</tr>
<tr>
<td>Average chars. processed</td>
<td>3.5</td>
<td>61.6</td>
</tr>
<tr>
<td>Average chars. per msec.</td>
<td>1.7</td>
<td>12.1</td>
</tr>
</tbody>
</table>

**CHANNEL INTERRUPTS**

<table>
<thead>
<tr>
<th></th>
<th>input</th>
<th>output</th>
<th>other</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>software &quot;interrupts&quot;</td>
<td>3,243</td>
<td>3,155</td>
<td>113</td>
<td>6,511</td>
</tr>
<tr>
<td>average time (msec.)</td>
<td>2.45</td>
<td>1.02</td>
<td>2.36</td>
<td>1.76</td>
</tr>
</tbody>
</table>

**TTY_BUF SPACE MANAGEMENT**

- Total size of buffer pool: 19,368 words
- Number of channels configured: 191
- Number of multiplexed channels: 18
- % of buffer pool in use:
  - input: 8.3
  - output: 13.8
  - control structures: 51.1
  - total: 73.2

Smallest amount of free space ever: 2,594 words (13% of buffer pool) should be less than 70% on normal system.

**CHANNEL LOCK CONTENTION**

- Number of calls to tty_lock: 12,963
- Times channel lock found locked: 554 (4% of attempts)
- Average time spent waiting for lock: 7.93 msec.
- Maximum time spent waiting for lock: 3675.17 msec.
Interrupts queued because channel locked 553 (8.5% of interrupts)

ECHO NEGOTIATION

Average time of transaction 1.7 msec.
Chars. echoed by supervisor 389 (7.72% of input chars) 353 (7.00% of input chars)
Chars. echoed by FNPs

ABNORMAL EVENTS

Input restarts 0 (0.00% of read calls)
Output restarts 0 (0.00% of write calls)
Output space overflows 0 (0.00% of write calls)
"needs_space" calls 0

*times couldn't do output conversion, retry but fully check motors on prev page.*
channel_comm_meters

Total metering time 78:15:27

FNP has been up for 78:15:25
Number of channels configured 73
Average number dialed up 54.2
FNP idle 67.0%
Idle at peak load 0.0% not reliably use FNP idle

<table>
<thead>
<tr>
<th>Characters transmitted</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>7,941,871</td>
<td>38,672,473</td>
</tr>
<tr>
<td>Characters per second</td>
<td>28</td>
<td>137</td>
</tr>
</tbody>
</table>

Abnormal DIA status 0
Memory EDAC errors 0 not consistently check memory

Memory size 64K
Total available buffer pool 17,888 words
Avg. amount of free space 2,448 words dubious
Average % of buffer pool available -13.7
Number of buffer allocations 3,245,838
Number of buffers preallocated 900,660 (27.8% of allocations)
Used preallocated buffer 900,656
No preallocated buffer available 126
Buffer allocation failures 0 0 0
Output restricted by space 0

Number of interrupts from this FNP 1,996,459
Avg. time/interrupt (ms) 3.3
% of total CPU time 2.3

Mailbox transactions:

| Input data | 756,349 |
| Output data | 557,909 |
| Input control | 563,976 |
| Output control | 122,001 |
| Total        | 2,000,235 |

Average inbound mailboxes in use 0.2
Average outbound mailboxes in use 1.0
Maximum outbound mailboxes in use 8
No outbound mailbox available 56,638
Input rejects 3 not enough space in buffer
% of input transactions rejected 0.00

Total metering time 78:15:30 goes with next page
channel_comm_meters

a.h000

<table>
<thead>
<tr>
<th>before conversion</th>
<th>after conversion</th>
<th>ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total characters input</td>
<td>3,471</td>
<td>3,274</td>
</tr>
<tr>
<td>Total characters output</td>
<td>874,287</td>
<td>1,243,897</td>
</tr>
<tr>
<td>Average length of input</td>
<td>5.0</td>
<td>4.7</td>
</tr>
<tr>
<td>Average length of output</td>
<td>60.9</td>
<td>86.6</td>
</tr>
</tbody>
</table>

Number of calls read: 690, write: 14,361, control total: 98

Average time per call (msec.) read: 3.2, write: 8.6
Average chars. processed per call read: 5.0, write: 60.9

Number of software interrupts: 10,891
Average time per interrupt (msec.) read: 1.2

Effective speed (cps) read:.input: 0.0, output: 0.0

Output overlaps in FNP: 2,875
Average length of DIA request queue: 0.5

Exhaust status
Software transfer timing error: 0
Pre-exhaust status: 0
Bell/quits: 0
Echo buffer overflows: 0
Parity errors: 0

Avg. number of pending status events: 0.6
Software status queue overflows: 0
Hardware status queue overflows: 0
Input buffer allocation failures: 0

Always run by not interesting usually good.

Things going to go from main to DIAL
But not measured: time. It is usually busy.

Bad things: FNP could not keep up with input.

Possible errors: 0

0 couldn't allocate buffer: exhaust status
Bell/quits on ASYNV
### channel_comm_meters

**a.h006**

<table>
<thead>
<tr>
<th></th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Messages transmitted</td>
<td>226,489</td>
<td>178,924</td>
</tr>
<tr>
<td>Minimum message length</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Maximum message length</td>
<td>133</td>
<td>133</td>
</tr>
<tr>
<td>Average message length</td>
<td>5.19</td>
<td>25.25</td>
</tr>
</tbody>
</table>

Frames dumped: 413
Frames retransmitted: 4
Receiver reset request: 0
Transmitter reset: 0
Frame check errors: 46
Frame aborts received: 35

Output overlaps in FNP 37,741
Average length of DIA request queue 1.5

Exhaust status 19
Software transfer timing error 0
Parity errors 0

Avg. number of pending status events 1.8
Software status queue overflows 0
Hardware status queue overflows 0
Input buffer allocation failures 0
channel_comm_meters

a.h008

<table>
<thead>
<tr>
<th>Idle time</th>
<th>42.19%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>Output</td>
</tr>
<tr>
<td>Blocks NAKed</td>
<td>2</td>
</tr>
<tr>
<td>Transmission timeouts</td>
<td>0</td>
</tr>
<tr>
<td>NAKs for DIA backlog</td>
<td>68</td>
</tr>
<tr>
<td>All transmission suspended</td>
<td>0</td>
</tr>
<tr>
<td>Blocks transmitted</td>
<td>4,812</td>
</tr>
<tr>
<td>Records transmitted</td>
<td>43,204</td>
</tr>
<tr>
<td>Average records per block</td>
<td>8.98</td>
</tr>
<tr>
<td>Duplicate input blocks</td>
<td>106</td>
</tr>
<tr>
<td>Output reprocessing</td>
<td>0</td>
</tr>
<tr>
<td>Blocks reprocessed</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Messages transmitted</td>
<td>4,812</td>
</tr>
<tr>
<td>Minimum message length</td>
<td>8</td>
</tr>
<tr>
<td>Maximum message length</td>
<td>397</td>
</tr>
<tr>
<td>Average message length</td>
<td>*B 344.85</td>
</tr>
<tr>
<td>Output overlaps in FNP</td>
<td>1,522</td>
</tr>
<tr>
<td>Average length of DIA request queue</td>
<td>0.5</td>
</tr>
<tr>
<td>Exhaust status</td>
<td>0</td>
</tr>
<tr>
<td>Software transfer timing error</td>
<td>2</td>
</tr>
<tr>
<td>Parity errors</td>
<td>0</td>
</tr>
</tbody>
</table>

| Avg. number of pending status events | 0.8 |
| Software status queue overflows | 0 |
| Hardware status queue overflows | 0 |
| Input buffer allocation failures | 0 |
Total metering time 78:15:36

<table>
<thead>
<tr>
<th>Name</th>
<th>user</th>
</tr>
</thead>
<tbody>
<tr>
<td>a.h000</td>
<td>Initializer</td>
</tr>
<tr>
<td>a.h001</td>
<td>Initializer</td>
</tr>
<tr>
<td>a.h002</td>
<td>Initializer</td>
</tr>
<tr>
<td>a.h003</td>
<td>Initializer</td>
</tr>
<tr>
<td>a.h004</td>
<td>Initializer</td>
</tr>
<tr>
<td>a.h005</td>
<td>Initializer</td>
</tr>
<tr>
<td>a.h006</td>
<td>Initializer</td>
</tr>
<tr>
<td>a.h007</td>
<td>Initializer</td>
</tr>
<tr>
<td>a.h008</td>
<td>Initializer</td>
</tr>
<tr>
<td>a.h009</td>
<td>Initializer</td>
</tr>
<tr>
<td>a.h010</td>
<td>Initializer</td>
</tr>
<tr>
<td>a.h011</td>
<td>Initializer</td>
</tr>
<tr>
<td>a.h012</td>
<td>Initializer</td>
</tr>
<tr>
<td>a.h013</td>
<td>Initializer</td>
</tr>
<tr>
<td>a.h014</td>
<td>Initializer</td>
</tr>
<tr>
<td>a.h015</td>
<td>Initializer</td>
</tr>
<tr>
<td>a.h016</td>
<td>Initializer</td>
</tr>
<tr>
<td>a.h017</td>
<td>Initializer</td>
</tr>
<tr>
<td>a.h018</td>
<td>Initializer</td>
</tr>
<tr>
<td>a.h019</td>
<td>Initializer</td>
</tr>
<tr>
<td>a.h020</td>
<td>Initializer</td>
</tr>
<tr>
<td>a.h021</td>
<td>Initializer</td>
</tr>
<tr>
<td>a.h022</td>
<td>Initializer</td>
</tr>
<tr>
<td>a.h023</td>
<td>Initializer</td>
</tr>
<tr>
<td>a.h024</td>
<td>Initializer</td>
</tr>
<tr>
<td>a.h025</td>
<td>Initializer</td>
</tr>
<tr>
<td>a.h026</td>
<td>Initializer</td>
</tr>
<tr>
<td>a.h027</td>
<td>Initializer</td>
</tr>
<tr>
<td>a.h028</td>
<td>Initializer</td>
</tr>
<tr>
<td>a.h029</td>
<td>Initializer</td>
</tr>
<tr>
<td>a.h030</td>
<td>Raich</td>
</tr>
<tr>
<td>a.h031</td>
<td>Raich</td>
</tr>
</tbody>
</table>

**Legend:**

- i: Invalid message
- o: Output message retransmitted
- t: Timeout waiting for acknowledge
- x: Pre-exhaust status
- X: Exhaust status
- s: Software transfer timing error
- b: Bell/quit
- e: Echo buffer overflow
- p: Parity error
- Q: Software status queue overflow
- q: Hardware status queue overflow
- a: Input buffer allocation failure
- A: Asynchronous channel
- B: Synchronous channel
- E: Echoplex mode
<table>
<thead>
<tr>
<th>%busy</th>
<th>0</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>07/13/84 1545.</td>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1546.</td>
<td></td>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1547.</td>
<td></td>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1548.</td>
<td></td>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1549.</td>
<td></td>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1550.</td>
<td></td>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1551.</td>
<td></td>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1552.</td>
<td></td>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1553.</td>
<td></td>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

FNP a idle time from 07/13/84 1545.2 to 1554.2: 81.3%
Busiest sample interval:
07/13/84 1546.2 to 1547.2: 76.8% idle Normal!
Busiest single sample: 07/13/84 1554.2: 0.0% idle

10-10 minute intervals are typical.

You can balance lines between FNP's

For x.25, remove tracing module 4 or re-tuning, or reduce lines.
Upgrade cache
0 or if GTP 1 model upgrade it so it can use cache.
MCS Debugging

- FNP Problems/Dumps
  - >scl>fnp_crash_notify.ec
  - dump management
  - debug_fnp
debug_fnp example

- db_fnp
- dump fnp.a.011084.1126

No TIB available

dump list dumps in dumps dir dumps changes dir dumps Looks in

- what

Dump in >udd>SysAdmin>Homan>fnp.a.011084.1126, version 6.5d

- why

illegal opcode fault at 46355 (util/6145) offset in util
utilities: tried to free space already free

- map

SOFTWARE configured: address does not conflict

Illegal opcode was intentional to force FNP crash,
Called IT executed CIP macro
<bind_fnp-list does some of this also, more detail

- regs

VALUES OF FNP REGISTERS

ic 046355 (util/6145) instruction counter
ac 137400 000040
ir 520077 (zero,carry,interrupt inhibit) indication register
x1 077340 (init/11034)
x2 004176
x3 077000 (init/10474)
er 777700
et 776700

- stat

BUFFER STATUS looks at TIB's in all lines

571 free

12 small space

FNP old default buffer size (32 word buffers) 571 x 32 words = free space

(50-100) is risky

low buffers causes BRK/QUIT & ignores input, not always fatal

LINE INPUT DIA OUTPUT TOTAL

a.h010 1 0 0 1
a.h015 1 0 0 1
a.h021 1 0 0 1

FNP Problems/Dumps
".crbdt"  = date and time of bootloading
00644 000000 112346 660120 236056 01/04/84 1050.2 hfe Wed
explain .crbuf;d .crbuf
".crbuf"  = starting address of buffer area
00650 066476
explain .crmem;d .crmem
".crmem"  = last location of memory
00651 177777
explain .crnbf;d .crnbf
".crnbf"  = number of buffers available
00652 001073
explain .criom;d .criom
".criom"  = start of iom table
00653 004200
explain .crnhs;d .crnhs
".crnhs"  = number of hsla's configured
00654 000002
explain .crnls;d .crnls
".crnls"  = number of lsla's configured
00655 000000
explain .crcon;d .crcon
".crcon"  = console enabled flag
00656 000000
explain .crmod;d .crmod
".crmod"  = starting address of module chain
00657 004562
explain .crnxa,bufc .crnxa,*
".crnxa"  = ptr to next available buffer
67300 67440 000040 000000 000000 000000 000000 000000 000000 000000 007000 007000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000
67310 67330 67300 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000
67330 =
67440 67550 000040 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000
67550 67570 =
explain .crtra;d .crtra
".crtra"  = trace entry enable mask
00661 317777
explain .crtrb;d .crtrb
".crtrb"  = base address of trace table
00662 174000
explain .crtrc;d .crtrc
".crtrc"  = next available location in trace table
00663 177536
explain .crreg;d .crreg
".crreg"  = disaster fault register storage location
00664 040522
explain .crttb;d .crttb
".crttb"  = location of tib table
00665 066316

FNP Problems/Dumps
explain .crtte;d .crtte
".crtte" = location of end of tib table
00666 066476
explain .crdly;d .crdly
".crdly" = head of delay table chain
00667 067142
explain .crver;d .crver
".crver" = mcs version number, 4 chars
00670 066056 065144 6.5d
explain .crbrk;d .crbrk
".crbrk" = addr of breakpoint control table
00672 000000
explain .crtsw;d .crtsw
".crtsw" = if non-zero, tracing will cease
00673 000000
explain .crnxs;d .crnxs;* 40
".crnxs" = next free small block
00674 066602
66602 066674 000056 400112 000000 066616 000004 400112 000000
66612 066616 000004 400112 000000 066642 000000 400112 000000
66622 066642 000004 400112 000000 066642 000004 400112 000000
66632 =
explain .crnbs;d .crnbs
".crnbs" = number of buffers devoted to small space
00675 000014
explain .crcct;d .crcct
".crcct" = address of first cct descriptor
00676 067166
explain .crskd;d .crskd
".crskd" = address of scheduler data block
00677 006442
explain .cretb;d .cretb
".cretb" = list of echo-negotiation bit tables
00700 066660
explain .crcpt;d .crcpt
".crcpt" = address of cpu page table
00701 004000
explain .crpte;d .crpte
".crpte" = address of variable cpu page table entry
00702 004177
explain .crtsz;d .crtsz
".crtsz" = size of trace data buffer
00703 004000
explain .crmet;d .crmet
".crmet" = non-zero if metering enabled
00704 000001
explain .crtdt;d .crtdt
".crtdt" = address of tib for t&d executive channel
00705 133512
explain .crbtm;d .crbtm
".crbtm" = address of time meters for getbuf/frebuf
00706 045270 util|5060
explain .crnxe;bufc .crnxe,* -bf
".crnxe" = next available space in extended memory

FNP Problems/Dumps
dump dump
.

No TIB available

tib

dump

124512 012207 207304 400002 015334 001116 123740 114340 000004
124522 377352 000000 000000 000000 000000 277002 000000 000001
124532 001431 310450 000000 015332 000000 131740 000000 000000
124542 077400 000000 013463 000013 000000 000000 000000 000000
124552 000000 000000 000000 000000 000000 000000 000000 000000
124562 000000 000000 077570 124620 124660

explanation

124512 012207 015334 001116 123740 114340 000004
124522 377352 000000 000000 000000 000000 277002 000000 000001
124532 001431 310450 000000 015332 000000 131740 000000 000000
124542 077400 000000 013463 000013 000000 000000 000000 000000
124552 000000 000000 000000 000000 000000 000000 000000 000000
124562 000000 000000 077570 124620 124660

explanation

124512 012207 207304 400002 015334 001116 123740 114340 000004
124522 377352 000000 000000 000000 000000 277002 000000 000001
124532 001431 310450 000000 015332 000000 131740 000000 000000
124542 077400 000000 013463 000013 000000 000000 000000 000000
124552 000000 000000 000000 000000 000000 000000 000000 000000
124562 000000 000000 077570 124620 124660

explanation

124512 012207 207304 400002 015334 001116 123740 114340 000004
124522 377352 000000 000000 000000 000000 277002 000000 000001
124532 001431 310450 000000 015332 000000 131740 000000 000000
124542 077400 000000 013463 000013 000000 000000 000000 000000
124552 000000 000000 000000 000000 000000 000000 000000 000000
124562 000000 000000 077570 124620 124660

explanation

124512 012207 207304 400002 015334 001116 123740 114340 000004
124522 377352 000000 000000 000000 000000 277002 000000 000001
124532 001431 310450 000000 015332 000000 131740 000000 000000
124542 077400 000000 013463 000013 000000 000000 000000 000000
124552 000000 000000 000000 000000 000000 000000 000000 000000
124562 000000 000000 077570 124620 124660

explanation

124512 012207 207304 400002 015334 001116 123740 114340 000004
124522 377352 000000 000000 000000 000000 277002 000000 000001
124532 001431 310450 000000 015332 000000 131740 000000 000000
124542 077400 000000 013463 000013 000000 000000 000000 000000
124552 000000 000000 000000 000000 000000 000000 000000 000000
124562 000000 000000 077570 124620 124660

explanation

124512 012207 207304 400002 015334 001116 123740 114340 000004
124522 377352 000000 000000 000000 000000 277002 000000 000001
124532 001431 310450 000000 015332 000000 131740 000000 000000
124542 077400 000000 013463 000013 000000 000000 000000 000000
124552 000000 000000 000000 000000 000000 000000 000000 000000
124562 000000 000000 077570 124620 124660

explanation
explain t.cur; d t.cur * oplock
"t.cur" = current address in control table
124515 015334 ctrl|1734
explain t.line; d t.line
"t.line" = 10 bit line number
124516 001116
explain t.icp; d t.icp; bufc t.icp,*
"t.icp" = first buffer in input chain
124517 123740
123740 134440 000074 155077 153055 173157 173077 157157 057075 \ .<m?k-{o(?oo/=?
123750 175075 067157 165075 125167 077165 173165 155077 057075 )=7ou-Uw?u(um?=??
123760 075077 077057 077077 067077 077077 153153 173173 157167 =>??/?????
123770 077155 155153 173077 133173 057057 155153 173177 ?mmk{??{{/\//mk{.
134440 142040 000074 157177 177177 177172 177177 165135 b .<o...z..u]..
134450 157177 177177 177157 073173 177133 077176 177133 055055 o...o;{.[?-.[--
134460 077177 073133 157077 073176 176177 057177 177153 177077 ?;[o;--;./..k.? 
134470 077073 177177 177133 133173 137177 077127 ?;...-]]=[[[...?W
142040 114340 000074 135177 073167 177177 177165 177165 176176 b'.<].;w...u.u--
142050 073067 133057 157157 131531 133133 133153 135153 157065 ;7[/o(om[[[]ko5
142060 155177 177177 147073 177175 177157 177177 157133 157157 157157 m...g.;i...)o[oo
142070 153057 073175 155167 177177 173057 077077 153157 057155 k;/;mw../??ko/m
114340 000000 000021 133133 133177 153065 165077 125135 155057 ....[[k.(5u?U)m/
114350 135077 173177 177000 000000 000000 000000 000000 000000 ]{[........................
114360 000000 000000 000000 000000 000000 000000 000000 000000 000000 ............
114370 =
explain t.ilst; d t.ilst
"t.ilst" = last buffer in input chain
124520 114340
explain t.icpl; d t.icpl
"t.icpl" = count of buffers in icp chain
124521 000004
explain t.icch; d t.icch
"t.icch" = address of next input character
124522 377352
explain t.elnk; d t.elnk
"t.elnk" = link to tib extension
124523 000000
explain t.rcp; d t.rcp
"t.rcp" = replay chain pointer (share t.elnk)
124524 000000
explain t.ocp; d t.ocp; bufc t.ocp,*
"t.ocp" = addr of output chain from cs
124524 000000
Invalid buffer address: 0
explain t.ocur; d t.ocur
"t.ocur" = addr of current output buffer
124525 000000
explain t.olst; d t.olst
"t.olst" = addr of last buffer in output chain
124526 000000
explain t.occh; d t.occh
"t.occh" - addr of next output character
124527 277002
explain t.ocnt;d t.ocnt
"t.ocnt" - count of buffers in t.ocur
124530 000000
explain t.type;d t.type
"t.type" - line type
124531 000001
explain t.time;d t.time
"t.time" - time at which next timeout will occur
124532 001431 310450
explain t.reta;d t.reta
"t.reta" = return address from calsub
124534 000000
explain t.dcwa;d t.dcwa
"t.dcwa" = addr of dcw list to 'execute'
124535 015332 ctrl|1732
explain t.dcwl;d t.dcwl
"t.dcwl" = length of dcw list
124536 000000
explain t.echo;d t.echo;d t.echo,* 40
"t.echo" = echo buffer address
124537 131740
131740 377354 377354 000157 057155 133133 153177 173065 165077
131750 125135 155057 135077 173177 177157 155135 133133 133135
131760 153157 065155 177177 177147 073177 177177 177157 177157
131770 133157 157153 057073 177155 167177 177173 057077 077153
explain t.dcp;d t.dcp
"t.dcp" = addr of message chain for cs
124540 000000
explain t.dlst;d t.dlst
"t.dlst" = last buffer in message chain for cs
124541 000000
explain t.ftse;d t.ftse
"t.ftse" = first time slot entry in ls1a table
124542 077400
d t.sfcm
124542 077400
explain t.bcmt;d t.bcmt
"t.bcmt" = counting temporary
124543 000000
explain t.brkp;d t.brkp
"t.brkp" = pointer to current break list
124544 013463
explain t.pos;d t.pos
"t.pos" = current carriage position
124545 000013
explain t.ecch;d t.ecch
"t.ecch" = address of current echo char (hsla only)
124546 000000
explain t.cntr;d t.cntr
"t.cntr" = counter for control tables
124547 000000
explain t.flg3;flags t.flg3
"t.flg3" = third word of flags
124550 000000
explain t.dtp;d t.dtp
"t.dtp" = pointer to delay table for this line
124551 000000
explain t.frmc;d t.frmc
"t.frmc" = framing chars (2 chars)
124552 000000
explain t.dcp1;d t.dcp1
"t.dcp1" = number of buffers in dcp chain
124553 000000
explain t.scll;d t.scll
"t.scll" = screenline length left, for echo neg.
124554 000000
explain t.sncc;d t.sncc
"t.sncc" = Echo negotiation sync ctr.
124555 000000
explain t.entp;d t.entp
"t.entp" = Echo negotiation break table ptr
124556 000000
explain t.ifch;d t.ifch
"t.ifch" = input flow control characters
124557 000000
explain t.ofch;d t.ofch
"t.ofch" = output flow control characters
124560 000000
explain t.omct;d t.omct
"t.omct" = output message count (for flow control)
124561 000000
explain t.itim;d t.itim
"t.itim" = time of last call to inproc (2 words)
124562 000000 000000
explain t.metr;d t.metr
"t.metr" = address of metering area
124564 077570
explain t.abf0;d t.abf0
"t.abf0" = absolute address of first permanent buffer
124565 124620
explain t.abfl;d t.abfl
"t.abfl" = absolute address of first permanent buffer
124566 124660
explain sfcm;d sfcm
Symbol "sfcm" has no explanation.
124400 002340 077472 077472 000012 077512 0000204 105740 103340
124410 000000 000000 000460 067200 000000 004374 000040 000040
124420 000040 377347 000000 000012 571526 0000004 000000 000000 700660
124430 000000 700460 000000 700460 000000 000000 000000 000000
124440 000000 000000 000000 600000 700460 000000 000000 032000 700660
124450 000000 000000 004000 700460 000000 000000 000000 600000 700460
124460 000000 000000 012000 700660 000000 000000
explain sf.hcm;d sf.hcm
"sf.hcm" = addr of hwcm
124400 002340
explain sf.nxa;d sf.nxa
"sf.nxa" = addr of next available queue entry
124401 077472
explain sf.nxp;d sf.nxp
"sf.nxp" = addr of next queue entry to process
124402 077472
explain sf.tly;d sf.tly
"sf.tly" = tally of status queue
124403 000012
explain sf.tib;d sf.tib
"sf.tib" = addr of tib for this line
124404 077512
explain sf.flg;flags
"sf.flg" = flag word
124405 000204 sffdet sffeoi
explain sf.ib0;d sf.ib0,* 40 -ch
"sf.ib0" = pointer to input buffer 1
105740 040377 373265 365277 325335 355257 335277 373377 .{5u?U]m/]?{... 
105750 373377 273173 377333 277376 377333 255255 277377 273133 .o;[...[--.];[
105760 357277 273376 376377 257377 377153 372777 277273 377377 o?;--;./..k.??;..
105770 376377 377377 333373 137137 377377 277327 335377 000000 ...[{___?W]...
explain sf.ib1;d sf.ib1,* 40 -ch
"sf.ib1" = pointer to input buffer 2
103340 040273 367377 377377 365377 365376 376273 267333 257157 ;w...u.u--;7[/o 
103350 173157 155135 333133 333135 153357 265355 377377 377347 {om][[[ko5m...g 
103360 273377 375377 357377 377157 333373 137137 377377 257273 377355 ;]}.o.o[ook/;m 
103370 167377 377373 257277 277353 357257 355333 333353 000000 w../??ko/m[k..
explain sf.ob0;bufc sf.ob0,*
"sf.ob0" = pointer to output buffer 1
Invalid buffer address: 0
explain sf.ob1;bufc sf.ob1,*
"sf.ob1" = pointer to output buffer 2
Invalid buffer address: 0
explain sf.pcw;d sf.pcw
"sf.pcw" = current pcw 2nd word
124412 000460
explain sf.cct;d sf.cct
"sf.cct" = cct addr for this line, if non-zero
124413 067200
explain sf.rct;d sf.rct
"sf.rct" = repeat count for status queue overflows
124414 000000
explain sf.hsl;d sf.hsl
"sf.hsl" = address of hsla table entry for this line
124415 004374
explain sf.bsz;d sf.bsz
"sf.bsz" = max buffer size
124416 000040
explain sf.fbs;d sf.fbs
"sf.fbs" = buffer size to be used during frame input
124417 000040
explain sf.mms;d sf.mms
"sf.mms" = maximum synchronous message size
124420 000040
explain sf.csz;d sf.csz
"sf.csz" = current asynchronous buffer size
124420 000000
explain sf.r.ms;d sf.r.ms
"sf.r.ms" = remaining unallocated message length
124421 377347
explain sf.nic;d sf.nic
"sf.nic" = char. address of next asynchronous input char
124421 377347
explain sf.noc;d sf.noc
"sf.noc" = char address of next asynchronous output char
124422 000000
explain sf.cfg;d sf.cfg
"sf.cfg" = 2 words for config pcw
124424 571626 000004
explain sf.waq;d sf.waq
"sf.waq" = wrap around queue, software status q
124442 600000 700460 000000 000000 032000 700660 000000 000000
124452 004000 700460 000000 000000 024000 700460 000000 000000
124447 700460 000000 000000 000000 600000 700460 000000 000000
124452 004000 700460 000000 000000 024000 700460 000000 000000
124452 012000 700660 000000 000000 024000 700460 000000 000000
124452 004000 700460 000000 000000 600000 700460 000000 000000
124452 004000 700460 000000 000000 600000 700460 000000 000000
explain sf.sta;d sf.sta
"sf.sta" = hardware status q, sicw here
124426 000000 700660 000000 700660 000000 700660 000000 000000
124436 000000 000000 000000 000000 000000
d hwcm
02340 105747 460057 103340 460075 131754 470000 131754 450000
02350 672000 124400 000000 000000 124426 420005 443026 010004
explain h.ric0;d h.ric0
"h.ric0" = primary receive icw
02340 105747 460057
explain h.ricl;d h.ricl
"h.ricl" = alternate receive icw
02342 103340 460075
explain h.sic0;d h.sic0
"h.sic0" = primary send icw
02344 131754 470000
explain h.sicl;d h.sicl
"h.sicl" = alternate send icw
02346 131754 450000
explain h.baw;d h.baw
"h.baw" = base address word
02350 672000
explain h.sfcm;d h.sfcm
"h.sfcm" = software comm. region address
02351 124400
explain h.mask;d h.mask
"h.mask" = mask register
02352 000000 000000
explain h.aiw;d h.aiw
"h.aiw" = active status icw
02354 124426 420005
explain h.cnfg;d h.cnfg
"h.cnfg" = configuration status

FNP Problems/Dumps

MCS Debugging
debug_fnp documentation

Name: debug_fnp, db_fnp

The debug_fnp command is a debugging aid intended to be used by FNP software developers and in FNP dump analysis. The command can be used to patch or dump memory in a running FNP, to examine a dump from a crashed FNP or a core image segment before it is loaded, to set breakpoints in a running FNP, symbolically display FNP control blocks, buffers, etc.

Usage

default

where initial_command_line is the first command(s) debug_fnp is to execute. Once the initial command(s), if any, are completed, debug_fnp reads command lines from user_input. Each line may contain multiple commands, separated by semi-colons. If an error occurs in any command, the remainder of the commands on that line will not be executed. Any debug_fnp command can be aborted by issuing a "Quit" followed by a Multics "program_interrupt" command.

Selecting debug_fnp mode

The debug_fnp command can be setup to operate on either a running FNP, a dump segment, or a core image segment. When first invoked, the command is setup to examine the first configured FNP. It is possible to switch between dumps, core images, and running FNP'S at any time. With few exceptions, most debug_fnp commands work the same regardless of whether a running fnp, a dump, or a core image is selected.

To select a running FNP:

    fnp tag

where "tag" is "a", "b", "c", or "d".

To select a core image:

    image path

To select a dump:

    dump path

where path is the Multics pathname of a segment containing the dump or the core image. Core image segments and dump segments have a different format, so these commands are not interchangeable. The pathnames on the dump and image commands can also be starnames, providing they match one and only one entry in the directory specified.
In most cases, it is not necessary to know the path name of the dump to be examined, as special commands are provided for selecting dumps.

To list all the dumps currently in the dump directory:

    dumps

The default dump directory is ">dumps" but this can be changed by:

    dump_dir (path)

where path is the pathname of the new dump directory. If "path" is omitted, the name of the current dump directory will be printed.

To select the latest dump:

    last_dump

The next earliest dump can be selected with:

    prev_dump

The prev_dump command can be used repeatedly as long as there are more dumps.

To select the next latest dump:

    next_dump

The next_dump and prev_dump commands can be used to peruse any or all of the dumps in the dump directory, going in either direction.

If dealing with a dump which contains multiple FNP's, such as a BOS fdump, the following command is used to select which FNP in the dump is examined:

    select_fnp tag

where tag is "a", "b", "c", or "d".

To find out what FNP, dump, or core image is selected:

    what

will print the fnp tag, or the pathname.

Expressions

Many of the following commands take numeric arguments such as addresses, lengths, etc. Any of these arguments can be expressed as a generalized FNP expression. Expressions can be arbitrarily complex, containing "("", ")", "+", "-", "+", "/", with their normal meanings and precedence. The symbol "/" is synonymous with "+", as in module|offset. Indirection can be specified by ",", following the address to indirect thru. Numeric constants are interpreted as octal, unless they are followed by a ".", in which case
they are decimal. The symbol "*" can be used for the current location counter, which is generally the last address used in a diaplay or patch command. Many common FNP symbols can also be used, including all fields in the system comm region, the hardware comm region, the software comm region, and the TIB. (Note: before TIB, hwcm, and sfcm addresses can be used, the addresses of these control blocks must be established. See the "line" and "set" commands). A symbol may also be any opblock mnemonic, the name of any FNP object module, or a machine instruction (specified by surrounding the instruction by apostrophes). In addition, user symbols can be defined.

Examples of expressions:

hslaj500
t.icp,*
*+30
tib|14,*+10
go
t 'ida 0,2,b.0'
cax3  (apostrophe not needed if no operand)

Displaying FNP memory

To display the contents of FNP words:

display address {length} {mode}
d address {length} {mode}

where "address" is the starting address, "length" is the number of words, and "mode" is the display mode. The symbol "*" will be set to the address specified. The following display modes can be used:

octal, oct
character, ch
address, addr (in form module|offset)
clock, ck (4 FNP words as a Multics clock)
instruction, inst (355 instruction format)
opblock, op (pseudo opblock format)
decimal, dec
bit
ebcdic, ebc

If omitted, the length defaults to 1 unless "address" is a predefined FNP symbol, in which case the appropriate length for that symbol will be used. Similarly, if the mode is omitted, octal is used, unless "address" is a predefined FNP symbol in which case the mode appropriate for that symbol is used.

To display a buffer:

buffer (address) {mode} { -brief|-bf}
buf (address) {mode} { -brief|-bf}

where "address" is the address of the buffer, "mode" is the mode to display it in (see display command), and -brief means display only the first 2 words
of the buffer. If "address" is omitted, the next buffer pointer from the previous buffer displayed is used. If "mode" is omitted, character mode is assumed. If -brief is not specified, the entire buffer is displayed. the length is determined automatically by reading the buffer header.

To display a buffer chain:

```
buffer_chain {address} {mode} {-brief|-bf}
bufc {address} {mode} {-brief|-bf}
```

If the data being displayed is in the form of threaded control blocks, the following commands can be used:

```
block {address} {-offset|-o offset} {-length|-l length}
blk {address} {-offset|-o offset} {-length|-l length}
```

will display a control block at the address specified. The length of the block is specified with -length. The default is 8 words. The offset to the forward pointer in the block is specified with -offset. The default is 0. If the address is not specified, the next block in the chain will be displayed (using the forward pointer from the previous block).

To display the entire chain of control blocks:

```
block_chain {address} {-offset|-o offset} {-length|-l length}
blkc {address} {-offset|-o offset} {-length|-l length}
```

will display control blocks until one with a zero forward pointer is encountered.

where the arguments are the same as in the buffer command. This command will follow the threads in the buffer chain, displaying each buffer.

If the data being displayed is a word of flags, the flags command can be used to show the setting of individual bits.

```
flags address {type}
```

where address is the the address of the word containing flags, and the type can be:

- **t.stat**: tib status word
- **t.flg**: first tib flag word
- **t.flg2**: second tib flag word
- **sf.flg**: hsla sfcm flags
- **istat**: interpreter status word
- **hs.1**: first word of hsla hardware status
- **hs.2**: second word of hsla hardware status

If {type} is ommitted, it is assumed to be the same as "address", which then must be one of the items in the above list. The flags are listed by name, as they appear in the macros.map355 source file. The explain command (see other commands) can be used to help with unfamiliar names. Occasionally, the value of a flag word is known (from a trace, for example), without knowing an
address of it. In this case, the following form can be used:

    flags =expression type

where expression is any valid expression, and type is one of the types shown above.

**Patching FNP Memory**

To patch the contents of FNP memory:

    patch address arg1 ... {argn}

where address is the starting address to patch, are the argi represent patch data. Each argi may be an expression representing the value to be stored in 1 FNP word, or a character string in quotes (which may contain more that 1 word of data). The total number of words patched cannot exceed 32. Before the patch is applied, the effects of the patch are displayed (old and new contents of every word) and the user is asked to verify that the patch is correct. The symbol "*" will be set to the address specified. Examples of patch commands:

```
patch 43102 203456 -1 2
patch .crver "3.1x"
pattern ctrl|1400 goto ctrl|1600
patch hsla|1541 'tze 13' cax3 'lda 0,2,b.1'
```

A shorthand form of this command is:

```
=arg1 ... {argn}
```

which is equivalent to:

```
patch * arg1 ... {argn}
```

Individual flag bits in words of flags can be manipulated with the following commands:

```
set_flag flag_symbol
```

will set the bit associated with the flag_symbol specified in the appropriate word. In a similar way,

```
clear_flag flag_symbol
```

will clear an individual bit. Currently, these commands are not indivisible operations: this means if other flags bits in the word are dynamically changing, these command may change their value if they happen to have been changed between the time the word was read and when it was rewritten.

**Dump Analysis Commands**
The following commands are only valid when using debug_fnp on a dump.

To find out the cause of a dump:

why

will print the type of fault which caused the crash, and if the crash was caused by a "die" opcode in the FNP, will interpret the reason for the crash.

The command:

regs

will print the contents of all machine registers at the time of the fault.

If the fault occured in a subroutine (as defined by the map355 'subr' macro), information about the call is available with:

    call_trace address {-long|-lg}

This command will start at the address specified and perform a backwards trace of all subroutine calls. If -long is specified, the registers saved at each subroutine level will also be printed. This command can also be used on a running FNP, but the information is probably changing too fast for the command to be useful.

FNP Trace Tables

A running FNP or a dump will contain a trace table of the most recent events occurring in the FNP. The trace table can be displayed with:

    print_trace {start}
    print_trace {start} {count}

where start indicates the starting trace message and count is the number of messages to display. If no arguments are given, the entire trace table is printed. If no count is given, the trace tables is displayed from the starting point specified to the end. If the start number is positive it is counted from the oldest message; if negative, it is counted from the most recent. For example:

    print_trace 200.

will skip the 199 oldest entries and print the rest.

    print_trace -50.

will print the 50 most recent messages.

Printing the trace table of a running FNP is only meaningful if tracing has been suspended; otherwise the table is changing too fast to be interpreted. Tracing can be suspendend in a running FNP by:
stop_trace

and restarted with:

start_trace

Tracing can also be stopped and started with some of the breakpoint commands explained below.

Which modules in the FNP are traced is determined by the trace mask, kept in FNP memory. This mask may be examined or updated with:

```
trace_mask (modules)
```

If used with no arguments, trace_mask will display and interpret the current trace mask. If modules are given, they represent modules to be added to or deleted from the current mask. The module should be specified as 'name' or '+name' to set the tracing bit for the module; it should be '-name' or '~name' to turn off the corresponding bit. In addition, all and none may be specified. For example:

```
trace_mask hsla ~dia -lsla
```

will turn on tracing for hsla_man and turn off tracing for dia_man and lsla_man.

**FNP Breakpoint facility**

The control table interpreter in the FNP allows breakpoints to be set in the interpreted control tables. A breakpoint will cause the line encountering it to stop execution in the interpreter until a command is given to restart it.

Breakpoints are often a useful tool but a certain amount of care must be exercised in their use. The following points are important:

1. Breakpoints can only be set in interpreted opblocks. They cannot be set at machine instructions.

2. While at a break, the line is executing an opblock equivalent to:

```plaintext
wait 0,0,0
```

followed by no status blocks. This means that timers can run out unnoticed, status will be ignored, hangups can be missed, etc. For this reason, it may be difficult to restart a channel after a breakpoint.

3. Breakpoints cannot be set at subroutine levels where waits would be illegal.

4. Breakpoints cannot be set when a restart may execute a waitm opblock.

5. Breakpoints cannot be set at a status opblock.
6. If a breakpoint is set at a wait opblock, it must be reset before the line is restarted. In addition, a breakpoint may not be set at a wait if any channels are currently waiting at that block.

7. Control tables that use local internal variables (as opposed to variables in the TIB extension) cannot depend of these variables being preserved during the break unless other channels that may use the same control tables are not running.

8. No notice is given when a channel encounters a breakpoint. The list_break command will list all breakpoints and show what channels are stopped at each one.

To set a breakpoint:

```
set_break address -line- {-stop_trace}
sb address -line- {-stop_trace}
```

will set a breakpoint at the address specified. If a tty channel is given, the breakpoint will apply to that line only. Any other channel encountering the breakpoint will continue execution. If -stop_trace is specified, the FNP will automatically suspend tracing if any channel stops at that breakpoint.

To reset a break:

```
reset_break address
reset_break -all
rb address
rb -all
```

will reset a break at the address specified. Any lines stopped at the break are not automatically restarted. If -all is specified, all breaks will be reset.

To start a channel stopped at a breakpoint,

```
start line {address} {-reset} {-start_trace}
start -all
sr line {address} {-reset} {-start_trace}
sr -all
```

will restart the line specified. If an address is given, the line will be restarted at the address given, instead of where it was stopped. If -reset is specified, the break will be reset before the line is started. If -start_trace is specified, tracing will resume as the line is restarted. If -all is specified, all lines at breakpoints at the time the command is issued will be restarted.

To list fnp breakpoints:

```
list_break
lb
```
Performance Analysis Commands

The FNP software periodically samples the instruction counter to determine whether the FNP is running or idling. This meter can be displayed with the fnp_idle command, as follows:

\[ \text{idle_time} \{\text{-reset|rs}\} \]

will print the percent of time the FNP has been idling since bootload, or the last time the command was invoked with the -reset control argument.

The sampling interval used by the FNP for metering this data can be printed or set with the following command:

\[ \text{sample_time} \{\text{new_time}\} \]

where new_time, if specified, is the new sampling interval in milliseconds. The argument must be between 1 and 1000. If no argument is given, the current sampling interval is printed. The default sampling time when the FNP is booted is 50 milliseconds.

More detailed information on FNP usage can be collected by configuring the module 'ic_sampler' in the FNP core image. This module will periodically sample the instruction counter (at the rate set by the sample_time command) and add 1 to a bucket which represents a small range (typically 16) of FNP addresses. With this data it can be determined with some precision where the FNP is spending its time when it is running.

This instruction counter sampling feature is controlled by the ic_sample command, which is only accepted if the ic_sampler module is configured in the FNP. The following options of the command are used to control ic sampling:

\[ \text{ic_sample start} \]

starts the ic sampling feature. Sampling is normally disabled when the FNP is booted.

\[ \text{ic_sample stop} \]

stops ic_sampling.

\[ \text{ic_sample reset} \]

zeroes all the sampling buckets.

The following options are used to display the information collected:

\[ \text{ic_sample module} \]

prints a table showing each module in the core image and what percentage of samples collected occur in that module.
ic_sample histogram|hist {fraction}

prints a histogram showing each bucket address that has data, and the percent of non-idle time that bucket represents. The fraction argument, if specified must be a floating point number between 0.0 and 1.0. If this option is used, the histogram will only contain the most frequently used buckets. Enough buckets will be printed so that fraction specified of the total data collected will be printed. For example, if the fraction is .9, 10% of the data collected will not be display by discarding infrequently referenced buckets. This option is useful in deleting "noise" from the histogram.

Other commands

To select a specific tty line:

line (line)

will locate the TIB, software comm region, and hardware comm region of the line specified. Once these addresses are set, fields in these control blocks can be referenced by name in any expression in other commands. The line can be specified either in Multics form (a.h012) or as an FNP line number (1014). If no line is specified, the name of the current line is printed. If the line selected is not on the current FNP, the proper FNP will be selected automatically.

To print a summary of FNP buffer usage:

buffer_status {-brief|-bf}
bstat {-brief|-bf}

will print a table showing each line and how much buffer space in the FNP it is using. If -brief is used, only summary information is printed.

To set a symbol:

set symbol value

where symbol is '*, 'tib', 'hwcm', 'sfcm', or any user defined symbol. Setting control block addresses (tib, hwcm, sfcm) is more easily done with the line command, but can be manually done with the set command in case internal FNP tables have been damaged. Note that setting any of these control block addresses has no effect on the current value of other control blocks. Setting "*" is also done by any dump or patch command. Once set, a symbol may be used in any expression in any other command.

To display a list of modules in the core image:

map

will display a list of modules and their addresses.

To interpret an FNP address:
convert_address {address1} ... {addressn}
cva {address1} ... {addressn}

will convert the address to any other meaningful form that can be derived. For example, octal values will be converted to module|offset, and vice versa.

To find the explanation of any FNP symbol (usually the output of a flags or convert_address command):

    explain sym1 {sym2} ... {symn}

where symi are symbols to be explained. This command will print the comment form the line in macros.map355 that defined the symbol.

To execute any Multics command:

    e Command Line

will pass 'Command Line' to the command processor.

To exit from debug_fnp,

    quit
    q
Multics Problems/Dumps

| tty_analyze
| tty_dump
| trace_mcs
tty_dump

LCTE at 13100, channel c.h212, devx 257
channel type: tty (0)
flags: in_use initialized
physical channel devx 257, major channel devx 165, subchannel 72
input_words 16, output_words 0
data base at 71124

WTCB at 71124, channel c.h212, devx 257
line type = ASCII, baud rate = 2400
flags: listen,dialed,send_output,tcb_initialized,breakall
hevent = 000464004602 40777000311, event = 00777000001 400000000002
fblock = 201512, lblock = 201512, fchar = 0
at line 0, column 0, white_col = 0
0 read-ahead messages
write_first = 0, write_last = 0
maximum buffer size = 16, buffer pad = 0

line delimiter = " 
"

TCB at 55010
terminal type = , old type = 0
modes: rawi,rawo,fulldpx,ctl_char,breakall,can_type=overstrike
flags: uproced_attached
shift state = 00 (none) ll = 0, pl = 0
answerback id = none
erase #, kill @, frame_begin \000, frame_end \000
input message size 0 characters
input_translation 0
output_translation 0
input_conversion 0
output_conversion 0
special 0
delay 0
read
201512 size = 16, tally = 49, flags:
0000000000061 123145164164 151156147040 124145155160 ...1Setting Temp
157162141162 171040160157 151156164145 162163040146 orary pointers f
162157155040 067063174062 064060056015 012015012141 rom 73|240.....a
172155072040 040000000000 000000000000 000000000000 zm: ............

Multics Problems/Dumps
tty_dump

LCTE at 10000, channel c.h016, devx 175
channel type: x25 (11)
flags: in_use initialized
physical channel devx 175, major channel devx 165, subchannel 10
input_words 0, output_words 0
data base at 127752
X.25 devx 175, 32 lc, 33 sc, ACTIVE flags: started send_output no_d
  packet_threshold=129 address=56

LCTE at 41200, channel c.h016.001, devx 1021
channel type: tty (0)
flags: in_use initialized
physical channel devx 1021, major channel devx 175, subchannel 1
input_words 0, output_words 0
data base at 131350

WTCB at 131350, channel c.h016.001, devx 1021
line type = ASCII, baud rate = 4800
flags: listen,dialed,send_output,qenable,tcb_initialized,hndlquit,count_lines
hevent = 000464013026 407777000721, event = 007777000001 400000000001
process blocked on input
fblock = 0, lblock = 0, fchar = 0
at line 7, column 0, white_col = 0
0 read-ahead messages
write_first = 0, write_last = 0
maximum buffer size = 16, buffer pad = 5
line delimiter = \\

TCB at 150604
terminal type = VIP7205, old type = 7
modes: can,esc,erkl,ifecho,hndlquit,echoplex,polite,scroll,can_type=replace
flags: uproc_attached
shift state = 00 (none) 11 = 80, pl = 23
answerback id = 3106
erase #, kill @, frame_begin \000, frame_end \000
input message size 0 characters
input_translation 0
output_translation 0
input_conversion 60
output_conversion 170
special 344
delay 0

Multics Problems/Dumps
Begin analysis of ERF 66

Header Values:
bleft 6366, free 61212

Physical channel c.h016, 4800 baud, devx 175, pcb 65460, lcte 10000, line type X25LAP
flags: listen dialed send_output sync_line

multiplexer type: x25
X.25 devx 175, 32 lc, 33 sc, ACTIVE flags: started send_output no_d
packet_threshold=129  address=56
X.25 SC 1 c.h016.001 devx 1021: DIALED output_ready wru echoplex hndlquit polite Ifech
LC 16: state=p41(FLOW CONTROL READY),max_packet_size=128,window used=0/2
  iti baud=4800
  address=3106305005411

Subchannel: c.h016.001, devx 1021
wtcb at 131350
flags: listen,dialed,send_output,qenable,hndlquit,count_lines,scroll
blocked for input

Physical channel c.h212, 2400 baud, devx 257, pcb 66300, lcte 13100, line type ASCII
flags: listen dialed send_output
wtcb at 71124
flags: listen,dialed,send_output,breakall
Read chain trace
FBLOCK
201512  size = 16, tally = 49, flags:
0000000000061 123145164164 151156147040 124145155160 157162141162 171040160157 15115616
162157155040 067063174062 064060056015 012015012141 172155072040 040000000000 00000000

Begin free chain trace
61212 (2 words)
61474 (16 words)
62016 (6 words)
62122 (2 words)

202732 (140 words)
203312 (10466 words)

Begin unthreaded space check
154702 (40 words)
157276 (14402 words)
10 March 82: trace_mcs

Function: Controls the MCS tracing facility and prints MCS trace table entries.

Syntax: trace_mcs print (channel_name(s)) {-control_arguments}
trace_mcs modes (new_modes) {-control_arguments}
trace_mcs channel (channel_name(s)) {-control_arguments}
trace_mcs table_size {new_table_size} {-control_arguments}

List of keywords:
print, pr, p
Prints entries from the trace table.
reset, rs
Resets MCS tracing: sets global modes to off,'default,none, sets the
trace table size to zero, and turns off both channel tracing flags
for all channels. No additional arguments are allowed.
modes
Prints the current global tracing modes, or changes the specified
modes if a new_modes argument is present. See "List of modes", below.

channel, chn
Prints or changes the tracing flags for a single channel or group of
channels. At least one of printing or changing must be specified, and
at least one channel_name must be specified.
table_size, ts
Changes the size of the MCS trace table. The table size may only be
changed when tracing is disabled, and if a trace table exists, the
size must be first changed to zero and then to the new value in order
to change the size to a different nonzero value. If no new table size
is supplied, the current table size is printed.

Arguments:
channel_name(s)
Up to 20 different channel names may be specified. The channel names
may be starnames, and all channels which match any of the supplied
names is selected for the operation.
new_modes
Is the new mode string containing the global MCS tracing modes to be
changed. See "List of modes", below.
table_size
Is a decimal integer specifying the number of entries in the MCS
trace table. Each entry occupies 16 words in tty_buf. The program
queries if the new trace table size will occupy more than 50 percent
of the free space in tty_buf, to guard against errors.

Control arguments (all functions):
erf NNN

Multics Problems/Dumps
Takes the MCS trace table from the FDUMP for ERF NNN. If this argument is specified, no parameters (modes, channel flags, table size) may be changed, although they can be printed.

Control arguments (print):
- reset, -rs
  Resets the last trace entry indicator in trace_mcs. Normally, only those entries which have been added to the trace table since the last time entries were printed are printed. If -reset is specified, the next use of trace_mcs will print all the trace table entries.
-all, -a
  Prints all trace table entries, but without resetting or changing the last entry indicator.
-last NNN, -lt NNN
  Prints the last NNN entries in the trace table, without resetting or changing the last entry indicator.
-reverse, -rev
  Prints the entries in reverse order. This can only be specified if -all or -last is also specified.
(-channel) XXX, (-chn) XXX
  Selects a channel or group of channels for printing. This control argument need not be supplied before the channel name, and is provided only for compatibility.

Control arguments (channel):
- print, -pr
  Causes the state of the channel trace flags for the selected channels to be printed. If -on or -off is also specified, both the previous and new states are printed.
-on
  Turns on the "trace" flag for the channel(s). If this flag is different from the "default" global mode, and the force flag is not also set, the channel is traced. If the force flag is set, the channel is traced regardless of the state of the "default" mode.
-off
  Turns off the "trace" flag for the channel(s). Only one of -on and -off may be specified.
-force, -fc
  Sets the "force" flag for the channel. If the force flag is set, the channel is traced or not depending only on the state of its "trace" flag, and not on the "default" mode. If -force is specified, one of -on or -off must also be specified.

Control arguments (modes):
- brief, -bf
  Suppresses the printing of the new mdes after the change is applied. Normally, the modes now in effect are printed.
-long, -lg
  Prints the new mode string after the changes are applied (Default).
List of modes:

**on**
Whether tracing is enabled at all. The "on" mode may also be represented as "'off", and "'on" as "off".

**default**
Whether channels are traced by default. Normally, this is off, meaning that only channels whose trace flag is set are traced.

**all**
May only be specified as "all", not "'all". This mode is a shorthand for setting all the remaining modes (except "none"), used to turn on tracing for all MCS events.

**none**
May only be specified as "none", and not as "'none". This mode is shorthand for resetting all the remaining modes. It is usually used in combination with some other mode or modes, to trace only those specific operations.

**read**
Whether channel_manager$read operations are traced.

**write**
Whether channel_manager$write operations are traced.

**data**
Whether the data in read and write operations is to be recorded in the trace table, as well as the events themselves.

**control**
Whether channel_manager$control operations are traced.

**modes**
Whether channel_manager$check_modes, get_modes, and set_modes operations are traced.

**interrupt**
Whether channel_manager$interrupt, interrupt_later, and queued_interrupt operations are to be traced.

**init_mpx**
Whether to trace priv_channel_manager$init_multiplexer operations.

**start_mpx**
Whether to trace priv_channel_manager$start_multiplexer operations.

**stop_mpx**
Whether to trace priv_channel_manager$stop_multiplexer operations.

**space_man**
Whether to trace calls to tty_space_man requesting non-buffer type space. (Apparently not implemented)

Access required:
Access to phcs_ is required to print the trace table of the running system. Access to the FDUMP is required to access the trace table in an FDUMP. Access to hphcs_ is required to change any parameters for the running system.

Notes:
The MCS trace table is kept in a circular array, with old entries being overwritten by new ones. Each entry contains the time, the device index of the associated channel, and a short string identifying the operation.
Typical Problems

- FNP load fails
- FNP Crashes
- FNP Crashes Repeatedly
- Multiplexer won't load
- Channel won't dial up
- Channel dials up but no I/O
- Frequent quit signals
- User dials into wrong process
dn355_data.incl.pl1

segment in: /dev/include
entry modified: 06/21/85 1919.3
contents modified: 11/09/84 0855.8

/* BEGIN dn355_data.incl.pl1 */

/* Date Last Modified and Reason */

Created 07/25/74 by R. B. Snyder for new ttydim.
Modified 06/23/77 by J. Stern to add channel_work_reqd and cwork_count
Modified 08/14/78 by Robert Coren to remove devx_tab and invent PCBs
Modified 79 May 14 by Art Beattle to add fnp_mem_size
Modified December 1979 by Robert Coren to add FNP queue lock
Modified January 1980 by Larry Johnson to increase max number of FNP's to 8
Modified 02/12/80 by Robert Coren to add dcw_list_array_ptr
Modified 03/06/80 by Robert Coren to add some metering info
Modified 12/10/80 by Robert Coren to add get_meters_waiting flag
Modified 83-12-16 BIM to use a chanid instead of icm/channel fb's.
Modified 1984-07-26 BIM for paged icm.

/*

/* LOCKING RULES: A fnp is locked by its LCTE unless its LCTE is uninitialized.
In that case, the configuration_lock must be held.
if tty_lock@lock_lcte returns io_no_permission, then the caller must
lock@lock_fast the configuration lock and retry the LCTE lock. If
the lcte is now initialized, too bad. Otherwise, the config lock protects.

Configuration locking is interesting to init_multiplexer and
all of fnp ttd and reconfiguration. The guts of the multiplexer
pay no attention to it. Thus, if the LCTE can be locked, it MUST be
locked before changing the io_manager_assigned flag. */

/* format: style4,delnl,insnl,"ifthendo */

dcl max_no_355s fixed bin int static init (8) options (constant);
/* max no of 355s we can handle (arbitrary) */
dcl dn355_data$ external fixed bin;

dcl infop pointer;
dcl fnpp ptr;

dcl 1 datanet_info aligned based (infop),
   2 configuration_lock aligned,
   3 pid bit (36) aligned,
   3 event bit (36) aligned,
   3 flags aligned,
   4 notify_sw bit (1) unaligned,
   4 pad bit (35) aligned,
   2 no_of_355s fixed bin,  /* no. of FNP's */
   2 trace bit (1) aligned,
   2 debug_stop bit (1) aligned,
   2 pad (2) bit (36) aligned,
   2 per_datanet (max_no_355s) aligned like fnp_info;  /* data per datanet */
dcl 1 fnp_info aligned based (fnpn),
2 mbx_pt pointer,
2 pcb_array_ptr pointer,
2 dcv_list_array_ptr pointer,
2 no_of_channels fixed bin,
2 fnp_id,
3 fnp_tag char (1) unaligned,
3 fnp_number fixed bin (9) unsigned unaligned,
3 padc bit (18) unaligned,
2 io_chanid char (8) aligned,
2 astep ptr,
2 fnp_charulel fixed bin (35),
2 lastytr fixed bin,
2 bleft_355 fixed bin,
2 lsla_idx (0:5) fixed bin aligned,
2 no_of_channels fixed bin,
2 bootyrocelis_id,
2 free_count fixed bin,
2 free_size fixed bin (35),
2 free_count fixed bin,
2 mbx_in_use fixed bin (35),
2 mbx_unavailable fixed bin (35),
2 cumulative_mbx_in_use_fixed_bin (35),
2 max_mbx_in_use_fixed_bin,
2 mmx_in_use_fixed_bin,
2 cumulative_mbx_in_use_fixed_bin
2的生命体信息
2 fnp_space_restricted output fixed bin (35),
2 tand_pc bx fixed bin,
2 n_pages_wired fixed bin,
2 config_flags fixed aligned,
3 available bit (1) unaligned,
3 io_manager_assigned bit (1) unaligned,
3 pad bit (34) aligned,
2 ptx fixed bin,
2 ptp pointer unaligned;
/* number of times available FNP space restricted amount of output sent */
/* index of PCB for COLTS channel */
/* pages wired for loading */
/* reconfig says "yes" */
/* We have channel assigned to us */
/* page table index, used only at bootload */
/* page table for this FNP */

**** The following named constants are used to layout the
iom page tables. Each FNP has to have its own page
table because there is not enough room to have eight different
bootload images of 32 K and > 64 K of tty_buf

THE MAX TTY BUF LENGTH IS 192 K words. We could have another 16 K
easily, and then after that it would get hard. */

**** The layout

<table>
<thead>
<tr>
<th>Page</th>
<th>I/O address</th>
<th>Memory address</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>xxxxxxx</td>
<td>invalid PTW</td>
</tr>
<tr>
<td>1</td>
<td>2000</td>
<td>2000</td>
<td>write-enabled (mailbox)</td>
</tr>
<tr>
<td>2</td>
<td>4000</td>
<td>4000</td>
<td>write-enabled (mailbox)</td>
</tr>
<tr>
<td>3</td>
<td>6000</td>
<td>6000</td>
<td>write-enabled (mailbox)</td>
</tr>
<tr>
<td>4</td>
<td>10000</td>
<td>as needed</td>
<td>bootloader image segment page 0</td>
</tr>
<tr>
<td>35</td>
<td>110000</td>
<td>as needed</td>
<td>bootloader image segment page 31</td>
</tr>
<tr>
<td>36</td>
<td>112000</td>
<td>xxxxxxx</td>
<td>invalid PTW</td>
</tr>
<tr>
<td>63</td>
<td>160000</td>
<td></td>
<td>invalid PTW</td>
</tr>
<tr>
<td>64</td>
<td>200000</td>
<td>as needed</td>
<td>tty_buf page 0</td>
</tr>
<tr>
<td>127</td>
<td>260000</td>
<td>as needed</td>
<td>tty_buf page 63</td>
</tr>
<tr>
<td>255</td>
<td></td>
<td></td>
<td>tty_buf page 191</td>
</tr>
</tbody>
</table>

**** We assume that the page table starts at all zeros. */

declare FIRST_BOOTLOAD_PAGE fixed bin init (4) int static options (constant);
declare FIRST_TTY_BUF_PAGE fixed bin init (64) int static options (constant);

/* End include file dn355_data.incl.pl1 */
/* BEGIN INCLUDE FILE ... lct.incl pll */

/* Created by J. Stern 7/26/78 */
/* Metering information added by C. Hornig, March 1980. */
/* Unwired saved meters added by Robert Coren, December 1980 */

dcl lctp ptr;
dcl lctep ptr;
dcl lct_size fixed bin;
dcl 1 lct aligned based (lctp),
   2 max_no_lctes fixed bin,
   2 cur_no_lctes fixed bin,
   2 long_ptr ptr,
   2 queue_lock bit (36),
   2 pad (11) fixed bin,
   2 lcte_array (lct_size refer (lct.max_no_lctes)) like lcte; /* lct entries */

dcl 1 lcte aligned based (lctep),
   2 lock bit (36),
   2 data_base_ptr ptr unal,
   2 channel_type fixed bin (8) unal,
   2 flags unal,
   3 entry_in_use bit (1) unal,
   3 initialized bit (1) unal,
   3 notify_reqd bit (1) unal,
   3 locked_for_interrupt bit (1) unal,
   3 space_needed bit (1) unal,
   3 trace_force bit (1) unal,
   3 trace bit (1) unal,
   3 unused bit (1) unal,
   2 physical_channel_devx fixed bin (17) unal,
   2 major_channel_info,
   3 major_channel_devx fixed bin unal,
   3 subchannel fixed bin (17) unal,
   2 queue_entries,
   3 queue_head bit (18) unal,
   3 queue_tail bit (18) unal,
   2 word_counts,
   3 input_words fixed bin (17) unal,
   3 output_words fixed bin (17) unal,
   2 meters,
   3 in_bytes fixed bin (35),
   3 out_bytes fixed bin (35),
   3 in,
   4 calls fixed bin (35),
4 interrupts fixed bin (35),
4 call_time fixed bin (71),
4 interrupt_time fixed bin (71),
3 out like lote.meters.in,
3 control like lote.meters.in,
2 saved_meters_ptr ptr,
2 timer_offset bit (10) aligned,
2 pad (3) fixed bin (35);

dcl 1 lentp ptr;

dcl 1 lcntp aligned based (lentp),
   2 names (lct.max_no_lentes) char (32) unal;

dcl 1 saved_meters aligned based like lcte.meters;

/* END INCLUDE FILE ... lct.incl.pl1 */

/* pointer to unwired copy of meters saved at last dialup */
/* Head of list of timers for this channel */
/* ptr to logical channel name table */
/* logical channel name table */
/* channel names */
/* meters saved at dialup, allocated in tty_area */
pcb.incl.pl1

/* BEGIN INCLUDE FILE ... pcb.incl.pl1 */

/* Created 08/14/78 by Robert S. Coren */
/* Modified 02/19/80 by Robert S. Coren to add read_first & read_last */
/* Modified 12/10/80 by Robert S. Coren to add metering stuff */
/* Modified May 1981 by Robert S. Coren to add tandd_attached flag */

/* Describes physical channel blocks for FNP channels */

dcl n_pcbs fixed bin;
dcl pcbp ptr;

dcl 1 pcb_array (n_pcbs) based aligned like pcb;

dcl 1 pcb aligned based (pcbp),
   2 channel_desc unaligned,
   3 devx fixed bin (17),
   3 subchan fixed bin (7) unaligned,
   3 line_number unal,
   4 ls_hsla bit (1) unaligned,
   4 ls_no bit (3) unaligned,
   4 slot_no bit (6) unaligned,
   2 write_first fixed bin (17) unaligned,
   2 write_last fixed bin (17) unaligned,
   2 baud_rate fixed bin (17) unaligned,
   2 line_type fixed bin (17) unaligned,
   2 max_buf_size fixed bin (17) unaligned,
   2 write_ent fixed bin (17) unaligned,
   2 flags unaligned,
   3 listen bit (1),
   3 dialed bit (1),
   3 send_output bit (1),
   3 high_speed bit (1),
   3 sync_line bit (1),
   3 end_frame bit (1),
   3 hndquit bit (1),
   3 breakall_enabled bit (1),
   3 output_mbx_pending bit (1),
   3 copied_meters_ready bit (1),
   3 get_meters_waiting bit (1),
   3 tandd_attached bit (1),
   3 padb bit (24),
   2 read_first fixed bin (18) unsigned unaligned,
   2 read_last fixed bin (18) unsigned unaligned,
   2 saved_meters_ptr pointer unaligned,
   2 copied_meters_offset fixed bin (18) unsigned;

   /* physical channel block declaration */
   /* index of LCT entry */
   /* logical subchannel/lsla slot # correspondence */
   /* regular line number */
   /* on if hsla, off if lsla */
   /* line adapter (high or low speed) number */
   /* physical slot or subchannel number */
   /* offset of first buffer in output chain */
   /* offset of last buffer in output chain */
   /* baud rate of channel */
   /* line type */
   /* largest buffer to be allocated for output */
   /* number of characters in write chain */
   /* channel is ready for dialup */
   /* channel is dialed up or connected */
   /* channel is ready for output */
   /* needs large send_out threshold */
   /* synchronous channel */
   /* channel is waiting for formfeed */
   /* channel in hndquit mode */
   /* breakall mode allowed for this channel */
   /* a wtx mbx has been sent, but not relinquished */
   /* copy_meters operation has completed */
   /* waiting for get_meters operation to complete */
   /* this channel is in use by T & D */
   /* head of read chain (while reading from FNP) */
   /* tail of read chain (likewise) */
   /* pointer to (unwired) copy of meters at last dialup */
   /* offset in tty_buf of buffer meters copied to */

/* END INCLUDE FILE ... pcb.incl.pl1 */
BEGIN INCLUDE FILE ... tcb.incl.pl1 */

Date Last Modified and Reason
Created 04/19/77 by J. Stern (from part of tty.incl.pl1)
Modified 2/6/78 by Robert Coren to add input_msg_size
Modified 4/18/78 by Robert Coren to add framing_chars
Modified 8/31/78 by J. Nicholls to add scroll mode
Extracted 9/12/78 by J. Stern from tty_data.incl.pl1
Modified Oct.1979 by Robert Coren to expand to 36 possible modes
Modified 1/21/80 by Robert Coren to add seven bits
Modified 10/08/80 by Robert Coren to add meters for tty_read & tty_write
Modified: 10 November 1980 by G. Patler to add can_type and explicit padding
Modified 12/04/80 by Robert Coren to add saved copy of meters
Modified 2/24/81 by Robert Coren to add time spent in tty_read and _write
Modified April 1981 by Robert Coren to add time last dialed up

/*
dcl tcbp ptr;
dcl 1 tcb based (tcbp) aligned, /* declaration of per terminal control block */

/* terminal type name */
2 terminal_type char (32) unaligned,
2 tables,
  3 input_mvt table bit (18) unaligned,
  3 output_mvt table bit (18) unaligned,
  3 input_tct table bit (18) unaligned,
  3 output_tct table bit (18) unaligned,
  3 special char bit (18) unaligned,
  3 delay char bit (18) unaligned,
2 default_tables,
  3 df_input_mvt table bit (18) unaligned,
  3 df_output_mvt table bit (18) unaligned,
  3 df_input_tct table bit (18) unaligned,
  3 df_output_tct table bit (18) unaligned,
  3 df_special char bit (18) unaligned,
  3 df_delay char bit (18) unaligned,
2 special_input_chars unaligned,
  3 erase char (1) unaligned,
  3 kill char (1) unaligned,
2 old_type fixed bin (17) unaligned,
2 modes unaligned,
  3 edited bit (1) unaligned,
  3 insert output tabs mode */
  3 erase escape processing */
  3 don't convert input */
3 rawom bit (1) unaligned,
3 redm bit (1) unaligned,
3 vertsp bit (1) unaligned,
3 echo_cr bit (1) unaligned,
3 echo lf bit (1) unaligned,
3 echo_tab bit (1) unaligned,
3 hnd quit bit (1) unaligned,
3 full_duplex bit (1) unaligned,
3 echoplex bit (1) unaligned,
3 upper_case bit (1) unaligned,
3 replay bit (1) unaligned,
3 polite bit (1) unaligned,
3 blk_xfer bit (1) unaligned,
3 scroll bit (1) unaligned,
3 prefix nl bit (1) unaligned,
3 wake tbl bit (1) unaligned,
3 tflow bit (1) unaligned,
3 oflow bit (1) unaligned,
3 no_outp bit (1) unaligned,
3 eight bit bit (1) unaligned,
3 odd parity bit (1) unaligned,
3 modes_pad bit (7) unaligned,
2 id char (4) unaligned,
2 colmax fixed bin (8) unaligned,
2 linemax fixed bin (8) unaligned,
2 wrt_lchar fixed bin (17) unaligned,
2 input_msg_size fixed bin,
2 framing_chars unaligned,
3 frame_begin char (1) unaligned,
3 frame_end char (1) unaligned,
2 max_output_block fixed bin (18) unsigned unaligned,
3 input_suspend_seq unsigned,
3 count fixed bin (9) unsigned,
3 chars char (3),
3 input_resume_seq unsigned,
3 count fixed bin (9) unsigned,
3 chars char (3),
3 output_suspend_eta_seq unsigned,
3 count fixed bin (9) unsigned,
3 chars char (3),
3 output_resume_ack_seq unsigned,
/* don't convert output */
/* has red-shift function */
/* send real ff's and vt's if on, else escape them */
/* echo carriage returns */
/* echo line feeds */
/* echo tabs */
/* cr's on quit */
/* emit and receive simultaneously */
/* echo input characters on terminal */
/* map lower-case output into upper-case */
/* replay interrupted input */
/* output must start at left margin */
/* accept control characters */
/* block transfer or "frame" mode */
/* break on all characters */
/* scroll mode for CRT terminals */
/* prefix output with nl when input interrupted */
/* input wakes determined by wakeup table */
/* input flow control */
/* output flow control */
/* don't generate output parity */
/* don't strip input parity */
/* generate odd parity (if any) */
/* terminal id */
/* current maximum number of columns */
/* current maximum number of lines/frame */
/* char within last write block */
/* maximum input message size in chars */
/* frame-begin character */
/* frame-end character */
/* maximum size of output block in block_acknowledge */
/* sequence for input suspension */
/* likewise for input resumption */
/* sequence for output suspension or end_of_block */
/* likewise for resumption or ack */
3 count fixed bin (9) unsigned,
3 chars char (3),
2 flags unaligned,
 3 breakall_enabled bit (1) unaligned,
 3 dont_count_next bit (1) unaligned,
 3 keyboard_locking bit (1) unaligned,
 3 no_printer_off bit (1) unaligned,
 3 break_char_pending bit (1) unaligned,
 3 uproc_attached bit (1) unaligned,
 3 block_acknowledge bit (1) unaligned,
 3 flags_pad bit (27) unaligned,
2 actshift bit (2) unaligned,
2 cumulative_meters,
 3 read_calls fixed bin (35),
 3 write_calls fixed bin (35),
 3 read_chars fixed bin (35),
 3 write_chars fixed bin (35),
 3 read_time fixed bin (71),
 3 write_time fixed bin (71),
2 saved_meters like tcb.cumulative_meters,
2 can_type fixed binary (9) unaligned unsigned,
2 pad1 bit (27) unaligned,
2 time_dialed fixed bin (71);

/* END INCLUDE FILE ... tcb.incl.p11 */

/* tty dim flag bits */
/* channel is permitted to use breakall mode */
/* next output character is escaped */
/* ON if doing keybd locking for ASCII line type */
/* reject printer_off/printer_on orders */
/* break character is in preconverted buffer */
/* user process has attached device */
/* block acknowledgement output protocol */

/* tty shift, 00 none, 01 lower, 10 upper, 11 unknown */
/* continuously running meters */
/* number of calls to tty_read */
/* number of calls to tty_write */
/* after conversion */
/* before conversion */
/* total time spent in tty_read */
/* total time spent in tty_write */
/* meters saved at last dialup */

/* type of canonicalisation to use on this channel */
/* to word boundary */
/* clock time of last copy_meters order */
/* BEGIN INCLUDE FILE ... tty_buf.incl.pl1 */

/ * Date Last Modified and Reason. 
Created 04/19/77 by J. Stern (from part of tty.incl.pl1) 
Modified January 1978 by Robert Coren and Larry Johnson for variable-size buffers 
Modified 2/6/78 by Robert Coren to make circular_queue size settable 
Modified Aug 78 by J. Nicholls to move the buffer block format to a file of its own 
and wcb to its own plus other modification for ring 0 multiplexing, tty_buffer_block.incl.pl1 
Modified 7/17/79 by B. Greenberg for echo negotiation meters. 
Modified November 1979 by C. Hornig for MCS tracing. 
Modified December 1979 by Robert Coren to add FNP channel lock meter 
Modified February 1980 by Robert Coren to remove all references to circular buffer 
Modified March 1980 by Robert Coren to reorganize metering information 
Modified December 1980 by Robert Coren to add FNP-specific events 
Modified 24 March 1982, W. Olin Sibert, to add mcr_timer support, recoverable_error_severity 
Modified November 1984 by Robert Coren to add tty_area_lock */

dcl ttybp ptr, 
tty_buf$ ext static, /* tty buffer segment */
tty_ev fixed bin int static options (constant) init (57), /* event used for wait and notify */
abs_limit fixed bin (18) static options (constant) init (64), /* minimum number of words we will leave free */
input_bpart fixed bin (18) static options (constant) init (2), /* fraction of bleft we will allow for input */
output_bpart fixed bin (18) static options (constant) init (4), /* fraction of bleft we will allow for output */

dcl qblock_size fixed bin int static options (constant) init (16), /* size in words of a delay queue block */
dcl bsizec fixed bin int static options (constant) init (60), /* number of characters in smallest buffer */
dcl buf_per_second fixed bin int static options (constant) init (10), /* for figuring out max. buffer size based on speed */
dcl FNP_DUMP_PATCH_EVENT fixed bin int static options (constant) init (58),
dcl FNP_METER_EVENT fixed bin int static options (constant) init (59),
dcl TTY_AREA_LOCK_EVENT bit (36) aligned int static options (constant) init ("74"b3),

dol 1 tty_buf aligned based (ttybp), /* declaration of tty buffer seg */
  2 slock bit (36), /* per system lock */
  2 absorig fixed bin (24), /* abs address of this seg */
  2 borig bit (18), /* index of start of buffer area */
  2 bleft fixed bin (18), /* words left in pool */
  2 free bit (18), /* pointer to start of free pool */
  2 fnp_config_flags (8) bit (1) unaligned, /* flag(1) ON if fnp(1) configured */
  2 padb1 bit (28) unaligned, /* pointer to logical channel table */
  2 lct_ptr ptr,
  2 nrawread fixed bin (35), /* number of raw chars input, total */
  2 nrawwrite fixed bin (35), /* number of raw characters output */
  2 ninchars fixed bin (35), /* total input chars after conversion */
  2 noutchars fixed bin (35), /* total output chars before conversion */
  2 readblocked fixed bin (35), /* number of times go input blocked */
  2 nbloblocked fixed bin (35), /* number of times process output blocked */
/* min output buffer size */
/* divide by nblocked to get ave buffer size */
/* number of converted chars held in tty_read */
/* number of times tty_read has had to start over */
/* number of times tty_write has had to start over */
/* number of times tty_write has run out of buffers */
/* total time spent in tty_read */
/* total time spent in tty_write */

/* number of calls to tty_read */
/* number of calls to tty_write */
/* used in calls to iobm */
/* number of quits */
/* space_needed bit on in at least 1 lcte */
/* meter of uses of this facility */
/* count of times tty_buf.locked */
/* count of times necessary to loop to lock it */
/* total time looped trying to lock it */

/* total number of allocations performed in tty_buf */
/* total number of freeings in tty_buf */
/* time spent masked in tty_space_man$get_space */
/* time spent masked in tty_space_man$free_space */
/* number of unsuccessful attempts to allocate space */
/* cumulative amount of space allocated for input */
/* cumulative amount of space allocated for output */
/* cumulative amount of space allocated by tty_space_man$get_space */
/* number of increments to cumulative_input_space */
/* number of increments to cumulative_output_space */
/* number of increments to cumulative_control_space */
/* smallest amount of free space ever available */

/* amount of space currently allocated for input */
/* amount of space currently allocated for output */
/* amount of space currently allocated by get_space */
/* number of calls to tty_lock$lock entries */
/* number of times tty_lock found channel already locked */
/* longest time waited for any channel lock */
/* total amount of time spent waiting for channel locks */

/* cumulative time spent doing echo negotiation */
/* Echo-negotiated shipments */
/* Chars echoed by ring */
/* Chars echoed by mux */
/* Echo reinit */

/* Entries into negotiate */
/* For testing */
/* number of interrupts queued by tty_lock */
/* tracing information */
4 enable bit,
4 default_mode bit,
4 read bit,
4 write bit,
4 data bit,
4 control bit,
4 modes bit,
4 interrupt bit,
4 init bit,
4 start bit,
4 shutdown bit,
4 space_man bit,
4 pad_flags bit (6),
3 data_offset bit (18),
2 recoverable_error_severity fixed bin,
2 tty_area_lock like hc_fast_lock,
2 pad2 (13) fixed bin (35),
2 free_space fixed bin;

#include hc_fast_lock;

/* global tracing control */
/* whether to trace channels by default */
/* read */
/* write */
/* buffers on reads and writes */
/* control, priv_control, and hpriv_control */
/* (get set check) modes */
/* interrupt, interrupt_later */
/* init_multiplexer, terminate_multiplexer */
/* start, stop */
/* shutdown */
/* tty_space_man */
/* offset of tracing data */
/* Syserr severity for recoverable MCS errors */
/* Lock owned by mos_timer */
/* Offset of next timer to come due */
/* Number of timers outstanding */
/* Who is doing timers? */
/* How get get him */
/* CPU time spent spinning on timer lock */
/* Number of times timer lock locked */
/* Number of times timer lock waited on */
/* CPU time spent in call side timer operations */
/* CPU time spent polling (including channel_manager) */
/* Number of calls to mos_timer$set, set_wired */
/* Number of calls to mos_timer$reset, reset_wired */
/* Number of calls to mos_timer$change, change_wired */
/* Number of calls to mos_timer$poll */
/* Number of mos_timer calls pending */
/* to prevent contention in allocating/freeing in tty_area */
/* start of free space region */
/* BEGIN INCLUDE FILE ... tty_buffer_block.incl.pl1 */

/*
Separated from tty_buf.incl.pl1 aug 78 by J. Nicholls
Modified May 1979 by Larry Johnson to add max_buffer_tally array and to use unsigned variables.
*/

dcl blockp ptr;

dcl free_blockp ptr;

/* pointer which block entry is based on */

/* pointer to head of free space chain */

/* format of start of free block */

/* forward pointer to next free block */

/* number of words in this block */

dcl 1 free_block aligned based (free_blockp),

2 next bit (18),

2 size fixed bin;

/* buffer definition */

/* addr of next buffer */

/* number of characters in buffer */

/* room for 60 data characters */

/* the following array, if indexed by buffer.size_code will give maximum number of characters permitted in that buffer */

dcl max_buffer_tally (0:7) fixed bin int static options (constant) init (60, 124, 188, 252, 316, 380, 444, 508);

/* END INCLUDE FILE ... tty_buffer_block.incl.pl1 */
/* BEGIN INCLUDE FILE ... wtcb.incl.pl1 */

/* Moved from tty_buf.incl.pl1 Aug 78 by J. Nicholls plus changes for ring 0 demultiplexing 
Error code added Nov. 1982 by Robert Coren 
Modified December 1984 by Robert Coren to invent "more_flags" structure and 
its first flag, line_status_disabled */

dcl wtcbp ptr;  /* pointer to head of wtcb */

dcl 1 wtcb based (wtcbp) aligned,  /* wired terminal control block */
2 event fixed bin (71) aligned,  /* event channel for hangup/dialup signal */
2 event fixed bin (71) aligned,  /* users event channel, for uproc */
2 line_status bit (72) aligned,  /* actual line status sent by fnp */
2 tcb_ptr ptr unal,  /* pointer to tcb */
2 pad1 fixed bin (35),  /* formerly time dialedup */
2 hproc bit (36) aligned,  /* boss processid */
2 uproc bit (36) aligned,  /* tty user processid */
2 baud_rate fixed bin (18) unal uns,  /* baud rate of this line */
2 line_type fixed bin (18) unal uns,  /* line type for line protocol */

2 flags unaligned,
3 listen bit (1) unaligned,  /* if on, listen for dialups */
3 dialed bit (1) unaligned,  /* if on, line is dialed up */
3 send_output bit (1) unaligned,  /* on if DN355 requested more output */
3 qenable bit (1) unaligned,  /* if on, signal quits */
3 qflag bit (1) unaligned,  /* on after quit, causes writes to be ignored */
3 end_frame bit (1) unaligned,  /* write chain fills ards frame */
3 notify_reqd bit (1) unaligned,  /* if on, do notify after unlocking lock */
3 work_reqd bit (1) unaligned,  /* if on, call dn355 before returning */
3 dialing bit (1) unaligned,  /* if on, 355 is dialing a phone number */
3 dial_status_valid bit (1) unaligned,  /* if on, dial_status_code is valid */
3 input_available bit (1) unaligned,  /* input for this device is waiting in wired space */
3 tcb_initialized bit (1) unaligned,  /* if on, tcb has been initialized */
3 wflag bit (1) unaligned,  /* process blocked on output */
3 xflag bit (1) unaligned,  /* process blocked on input */
3 wru bit (1) unaligned,  /* reading answerback */
3 hndquit bit (1) unaligned,  /* on if in hndquit mode */
3 count_lines bit (1) unaligned,  /* on if tcb.linemax > 0 */
3 line_status_present bit (1) unaligned,  /* fnp has sent line status */
/* channel is synchronous line type */
/* channel is in breakall mode */
/* channel is in scroll mode */
/* ring zero to echo chars. */
/* on if in wake_tbl mode */
/* on to allow input wakeup in wake_tbl mode */
/* device must be told to enter receive mode */
/* write_with_mark call outstanding */
/* channel masked by FNP */
/* code returned by 355 after dialing a phone number */
/* oldest read pointer, block */
/* newest read block */
/* first block char index */
/* line number of current line */
/* tty column position */
/* current number of read-ahead msgs */
/* first write block */
/* last write block */
/* amount of chars in write chain */
/* column position resulting from trailing white space */
/* maximum-size buffer to be allocated for this channel */
/* amount of pad to be left in output buffers */
/* index into lot of channel's entry */
/* echo negotiation data ptr */
/* wakeup table offset */
/* number of chars in prompt string */
/* text of prompt message */
/* line delimiter for tty_read parse */
/* in addition to flags (above) */
/* "1"b => don't relay line_status interrupts */
/* error code returned by channel_manager to tty_interrupt */
<table>
<thead>
<tr>
<th>File Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>dn355_data.incl.pll</td>
<td>1</td>
</tr>
<tr>
<td>lct.incl.pll</td>
<td>4</td>
</tr>
<tr>
<td>pcb.incl.pll</td>
<td>6</td>
</tr>
<tr>
<td>tcb.incl.pll</td>
<td>7</td>
</tr>
<tr>
<td>tty_buf.incl.pll</td>
<td>10</td>
</tr>
<tr>
<td>tty_buffer_block.incl.pll</td>
<td>13</td>
</tr>
<tr>
<td>wtcb.incl.pll</td>
<td>14</td>
</tr>
</tbody>
</table>