F86 Class Schedule

	Day 1	Day 2	Day 3
Morning	OVERVIEW General Telecomm Concepts MCS Parts FNP Description	ADVANCED MCS USAGE tty_ TTF	FNP INTERNALS Programs and Databases
After-noon	MCS ADMINISTRATION CMF FNP Core Image Operator Commands	RING 0 MCS INTERNALS Programs and Databases Multics-FNP Dialogue	MCS METERING AND DEBUGGING Ring 0 MCS Meters FNP Meters debug_fnp Dump Analysis Tools

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

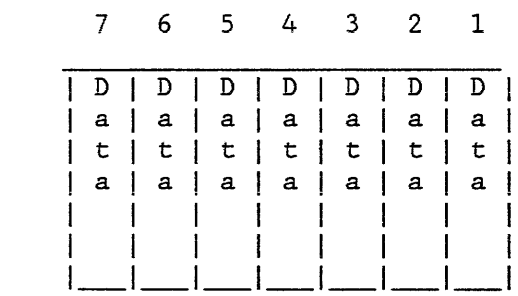
DW8

Standard Communications Concepts

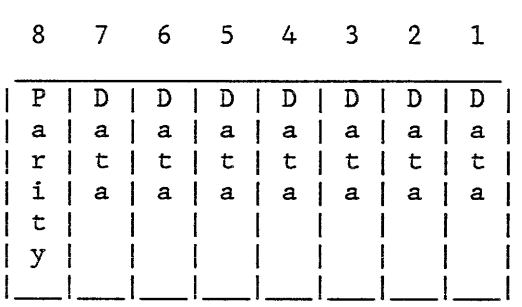
Some communications concepts that are not necessarily specific to Multics need to be understood in order to understand MCS.

- 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

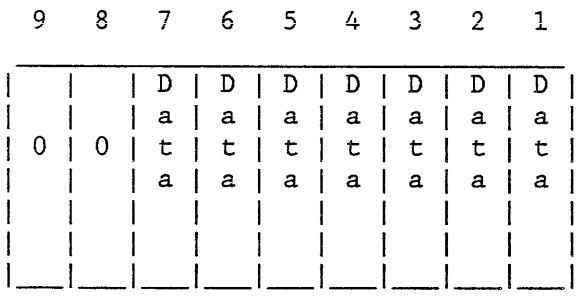
7-bit Character



7-bit Character Plus Parity Bit



7-bit Character Stored in 9-bit Byte



- 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

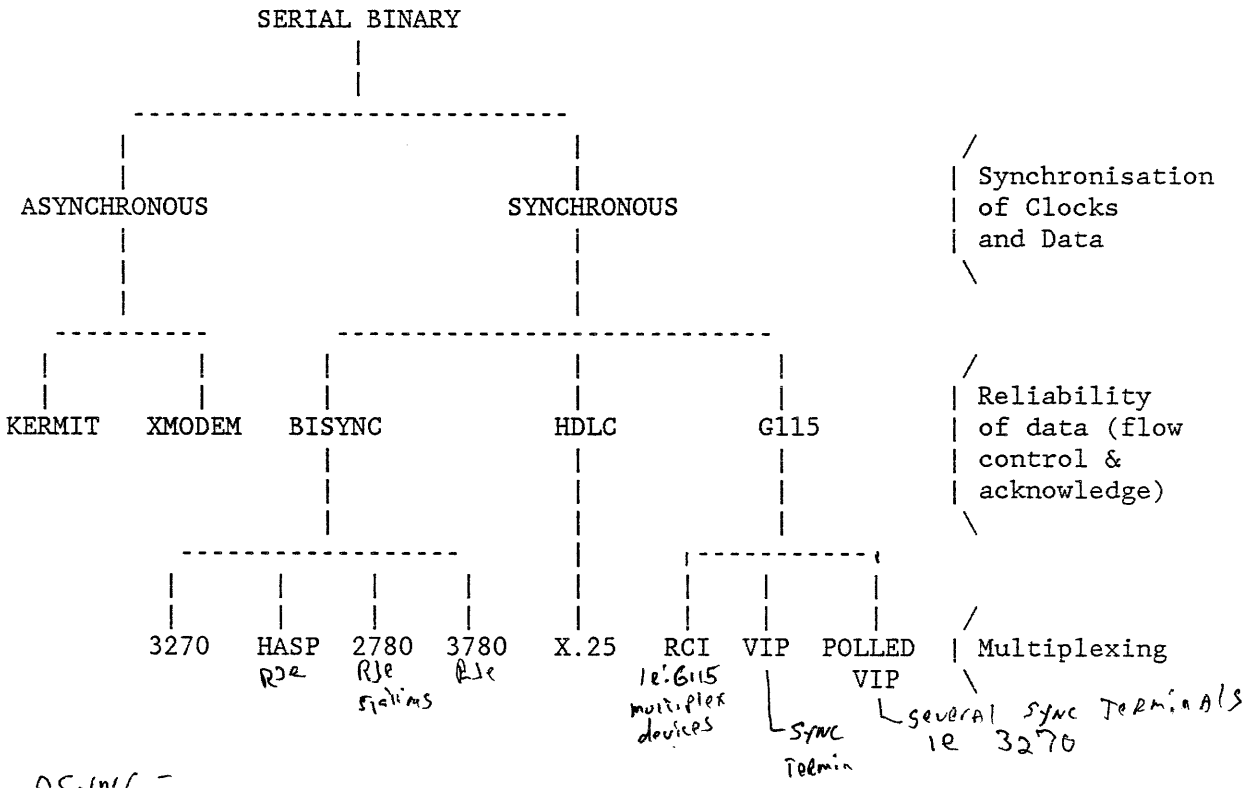
Bin	Oct	Dec	Hex	Multics Def.	Key/Name	Other Definitions
0000000	000	0	00	NUL	Control-@	Control-SP on some terminals
0000001	001	1	01		Control-A	SOH, Start of Header
0000010	002	2	02		Control-B	STX, Start of Text
0000011	003	3	03		Control-C	ETX, End of Text
0000100	004	4	04		Control-D	EOT, End of Transmission
0000101	005	5	05		Control-E	ENQ, Enquiry
0000110	006	6	06		Control-F	ACK, Acknowledge
0000111	007	7	07	BEL	Control-G	Bell
0001000	010	8	08	BS	Control-H	Backspace
0001001	011	9	09	HT	Control-I	Horizontal Tab
0001010	012	10	0A	NL	Control-J	Newline, Line Feed, LF
0001011	013	11	0B	VT	Control-K	Vertical Tab
0001100	014	12	0C	NP	Control-L	New Page, Form Feed, FF
0001101	015	13	0D	CR	Control-M	Carriage Return
0001110	016	14	0E	RRS	Control-N	Red Ribbon Shift, SO, Shift Out
0001111	017	15	0F	BRS	Control-O	Black Ribbon Shift,SI, Shift In
0010000	020	16	10		Control-P	DLE, Data Link Escape
0010001	021	17	11		Control-Q	DC1, X-ON
0010010	022	18	12		Control-R	DC2
0010011	023	19	13		Control-S	DC3, X-OFF
0010100	024	20	14		Control-T	DC4
0010101	025	21	15		Control-U	NAK, Negative Acknowledge
0010110	026	22	16		Control-V	SYN, Synchronous Idle
0010111	027	23	17		Control-W	ETB, End of Transmission Block
0011000	030	24	18		Control-X	CAN, Cancel
0011001	031	25	19		Control-Y	EM, End of Medium
0011010	032	26	1A		Control-Z	SUB
0011011	033	27	1B		ESC, Control-[	Escape, Alt-mode
0011100	034	28	1C		Control-\	FS, File Separator
0011101	035	29	1D		Control-]	GS, Group Separator
0011110	036	30	1E		Control-^	RS, Record Separator
0011111	037	31	1F		Control-_	US, Unit Separator
0100000	040	32	20	SP	blank	Space
0100001	041	33	21	!	exclamation point	
0100010	042	34	22	"	double quote	
0100011	35	35	23	#	number sign	
0100100	044	36	24	$	dollar sign	
0100101	045	37	25	%	percent	
0100110	046	38	26	&	ampersand	
0100111	047	39	27	'	acute accent	
0101000	050	40	28	(	left parenthesis	
0101001	051	41	29	)	right parenthesis	
0101010	052	42	2A	*	asterisk	
0101011	053	43	2B	+	plus	
0101100	054	44	2C	,	comma	
0101101	055	45	2D	-	minus	
0101110	056	46	2E	.	period	
0101111	057	47	2F	/	right slash	

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

1100110	146	102	66	f	small f	
1100111	147	103	67	g	small g	
1101000	150	104	68	h	small h	
1101001	151	105	69	i	small i	
1101010	152	106	6A	j	small j	
1101011	153	107	6B	k	small k	
1101100	154	108	6C	l	small l	
1101101	155	109	6D	m	small m	
1101110	156	110	6E	n	small n	
1101111	157	111	6F	o	small o	
1110000	160	112	70	p	small p	
1110001	161	113	71	q	small q	
1110010	162	114	72	r	small r	
1110011	163	115	73	s	small s	
1110100	164	116	74	t	small t	
1110101	165	117	75	u	small u	
1110110	166	118	76	v	small v	
1110111	167	119	77	w	small w	
1111000	170	120	78	x	small x	
1111001	171	121	79	y	small y	
1111010	172	122	7A	z	small z	
1111011	173	123	7B	{	left brace	
1111100	174	124	7C	|	vertical bar	
1111101	175	125	7D	}	right brace	
1111110	176	126	7E	~	tilde	
1111111	177	127	7F	PAD	DEL	Rubout

- 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 acknowlegement of received data.
    - Multiplex data for different devices on single line

Communications Protocols Supported by Multics



- 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%

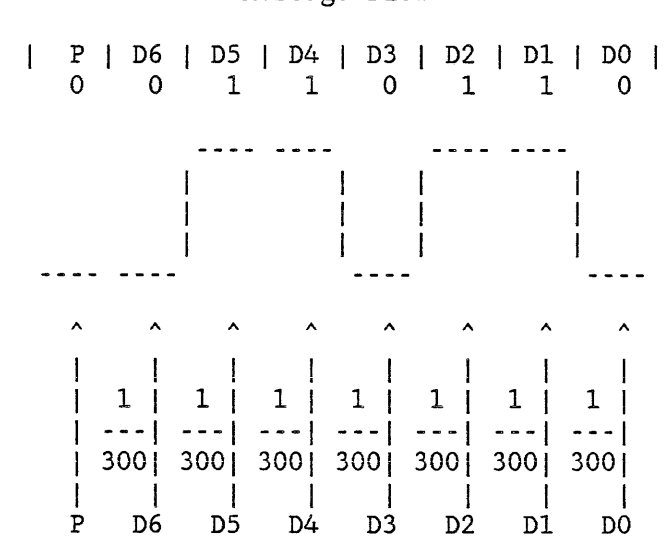
Asynchronous Protocol

Problem: Synchronisation of clocks and of data start

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

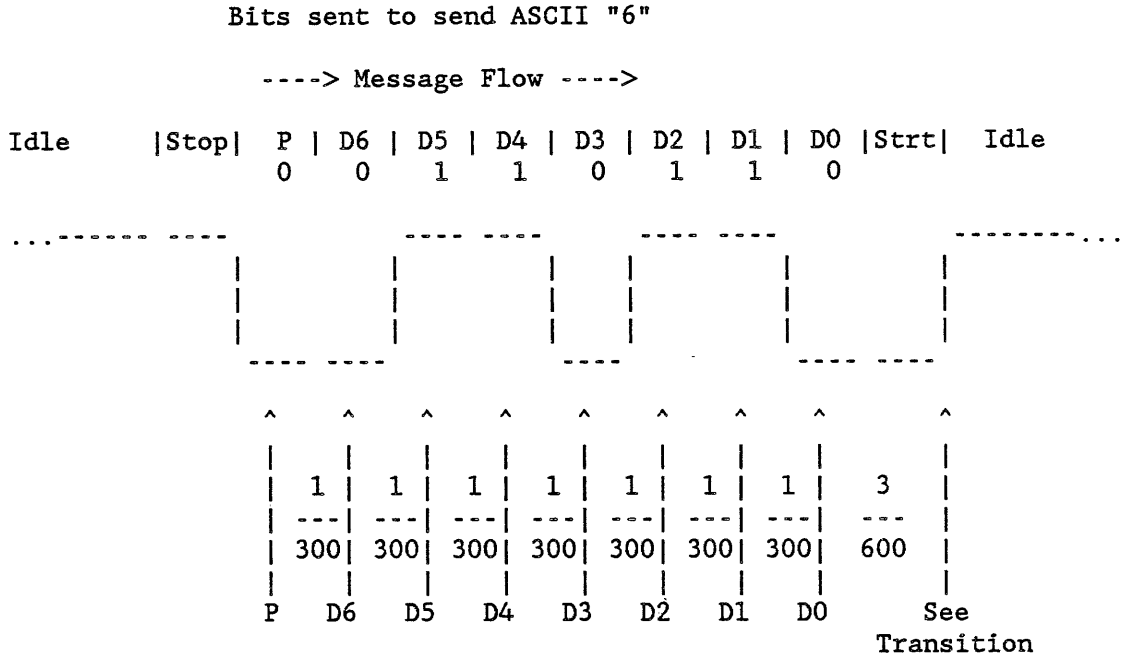
At 300 baud, 1/300 sends between bits.

Bits sent to send ASCII "6"

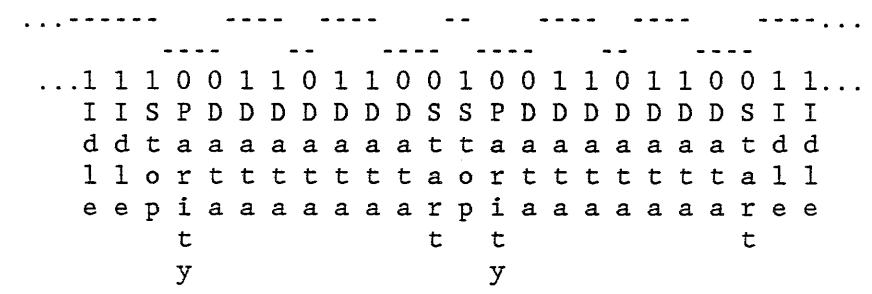


Sampling needed every 1/300 second to read received bits.

Bits sent to send ASCII "6"



Asynchronous throughput

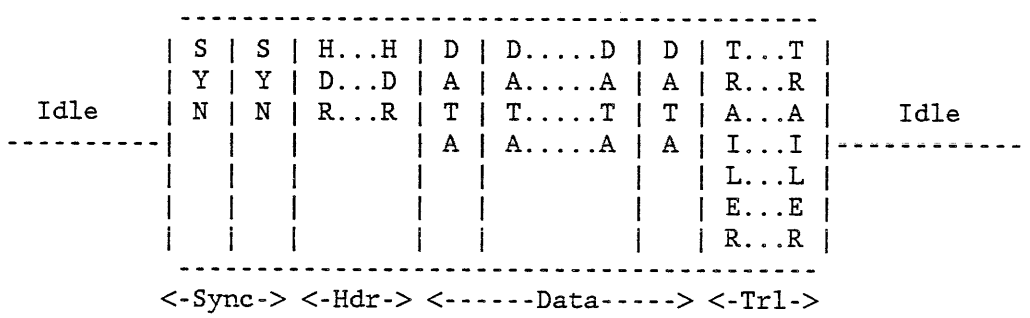


Maximum throughput: 70%

- 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



Typical Values:	2	2	100	2

2 Sync bytes
2 Header bytes
2 Trailer bytes
--
100 data bytes 8 Overhead bytes

Throughput: 100/108 = 93%

- In Multics, maximum speed for ~~asynchronous~~ 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

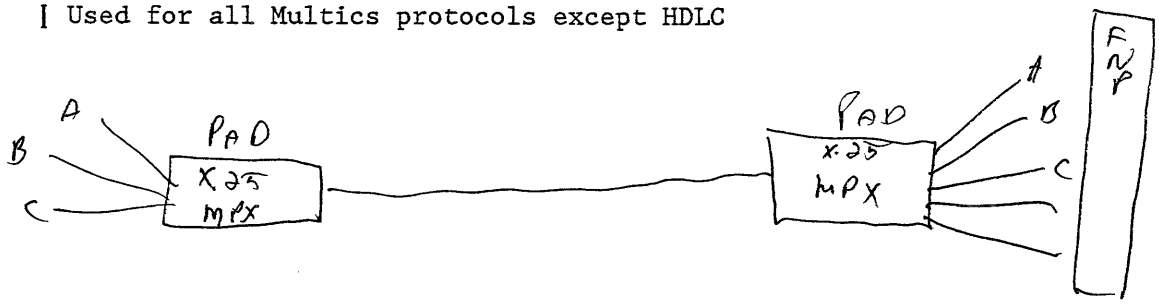
  | Data Circuit-terminating Equipment (DCE): Modems

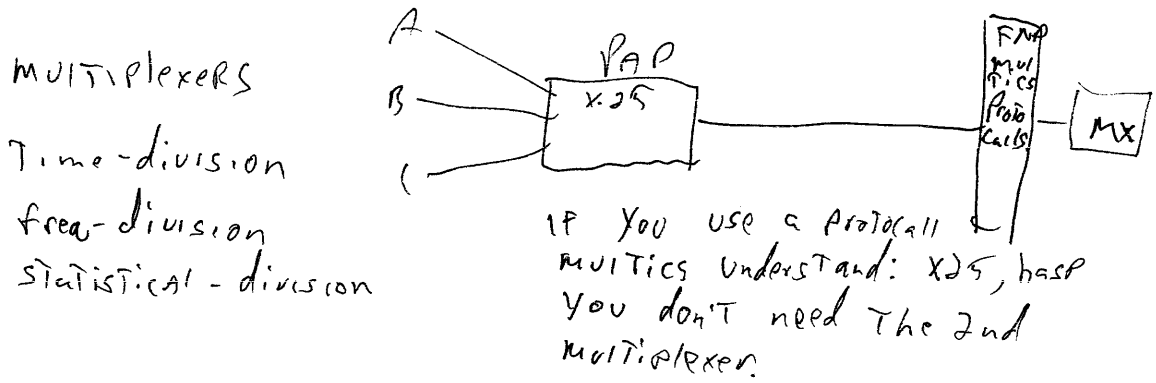
| Connection is by 25 pin connectors and cables

| Diagram shows the pins used by Multics in asynchronous connections

| Synchronous connection also uses pin 24 for timing (clock signal)

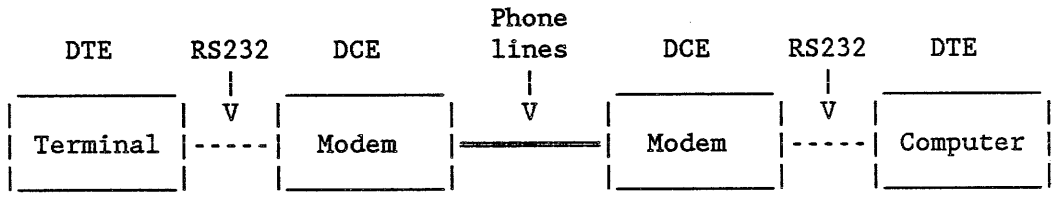
| Used for all Multics protocols except HDLC





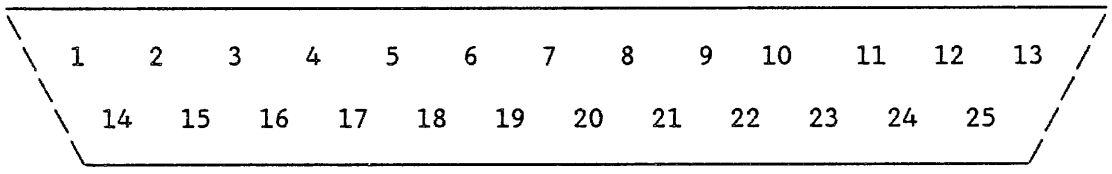
DCE - Data Circuit equip
DTE - Data Terminal equip

RS-232/V24

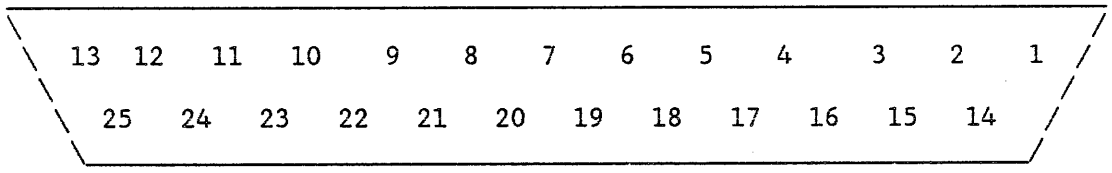


Pin		CCITT Circuit No.	Name	Nom	Description	Description
1		101	GND	TP	Ground	Terre de protection
7		102	GND	TS	Ground	Terre de signalisation
2	-->	103	TX	ED	Transmission	Emission de donnees
3	<--	104	RX	RD	Reception	Reception des donnees
4	-->	105	RTS	DPE	Request to send	Demande pour emettre
5	<--	106	CTS	PAE	Clear to send	Pret a emettre
6	<--	107	DSR	PDP	Dataset ready	Poste de donne pret
8	<--	109	CD	DP	Carrier Detect	Detection de la porteuse
20	-->	108	DTR	CPD	Data terminal ready	Connectez le poste de donnees

MALE



FEMALE



Modem Dialup Sequence

		TTY DTE	Modem DCE		Modem DCE	FNP DTE		
g	(TX)	2	------->	2	2	<-------	2	(TX) f
f	(RX)	3	<-------	3	3	------->	3	(RX) g
c	(RTS)	4	------->	4	4	<-------	4	(RTS) b
e	(CTS)	5	<-------	5	5	------->	5	(CTS) e
d	(DSR)	6	<-------	6	6	------->	6	(DSR) a
e	(CD)	8	<-------	8	8	------->	8	(CD) e
c	(DTR)	20	------->	20	20	<-------	20	(DTR) b

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:

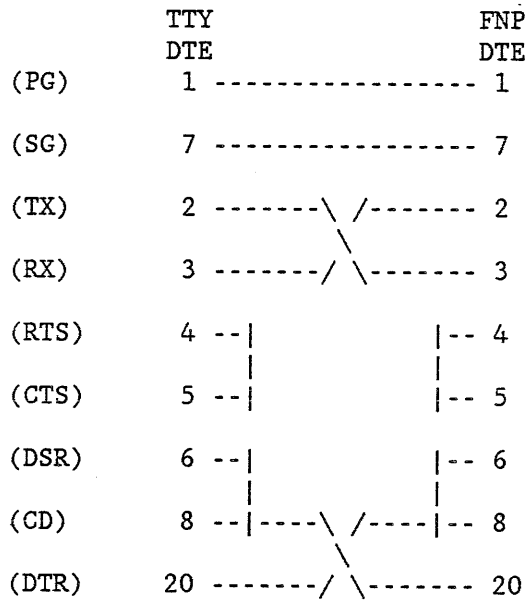
a) Modem on FNP is powered on. raises DSR
b) FNP boots and listens to the line. raises RTS DTR
c) The terminal is turned on. raises RTS DTR
d) The terminal's modem is turned on. Raises DSR
e) The telephone call is made and the modems are connected. raises (CTS CD)
f) Multics sends the login banner.
g) User types login line.

TX
RX } only changes during connection

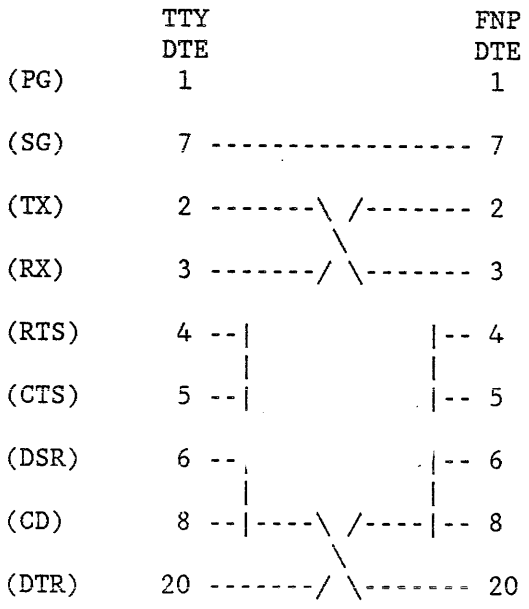
RTS - half duplex, Transmit mode or receive mode
DSR - connection established
CTS - flow control, Tied to DTR on terminal/printer side.

- Break/Interrupt
  - Out-of-band signal
  - Logic 0 on pin 2 for 100 to 600 msec
- Line Driver
  - Also known as short-haul modems or point-to-point modems
  - Inexpensive replacement for modems for point-to-point connections
  - No real standards
- Direct Connect
  - For very short distances, a direct RS232 connection is possible
  - Very inexpensive replacement for modems
  - Theoretically limited to about 50'
    - 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

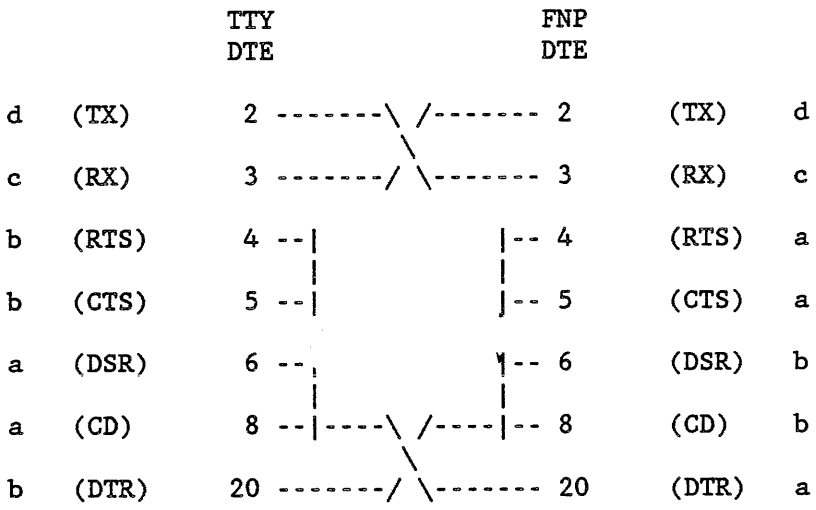
6-WIRE DIRECT CONNECT



5-WIRE DIRECT CONNECT



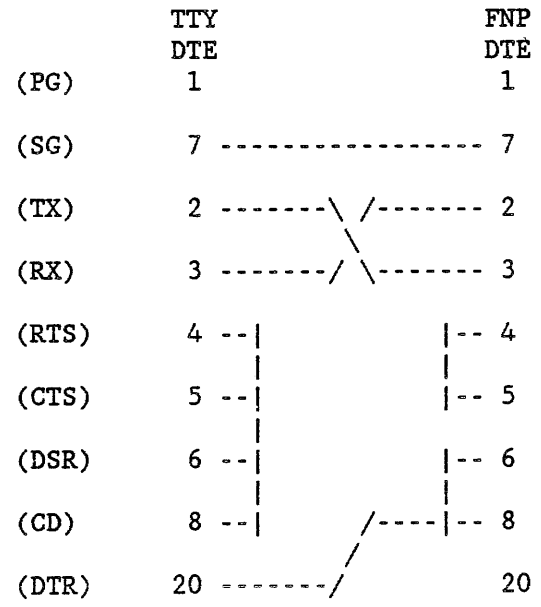
Direct-Connect Dialup Sequence



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



| Line Monitor

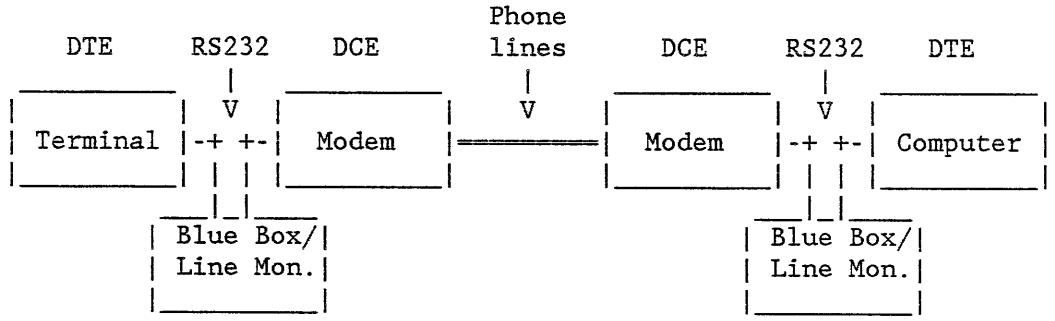
    | Monitor RS232 signals and display transmitted and received data

| Breakout Box

    | Monitor RS232 signals

    | Also known as a Blue Box.

Line Monitors, Blue Boxes



- 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

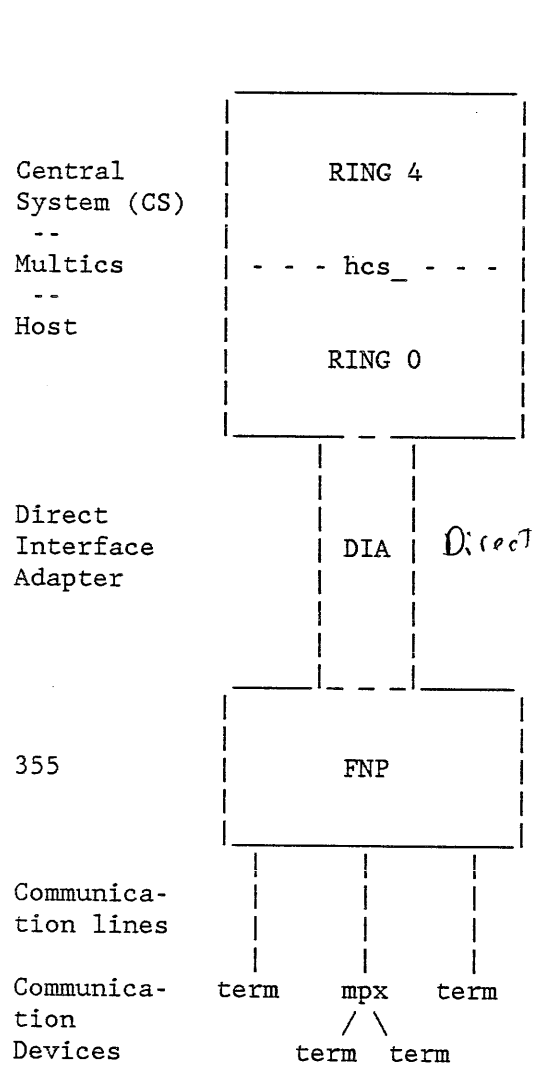
- Compare with Wait/Notify, to wait for events of short duration

  - 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
  - FNP
  - Ring 0
  - Ring 4



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 b
PL/I & ALM: Hardcore (MST)

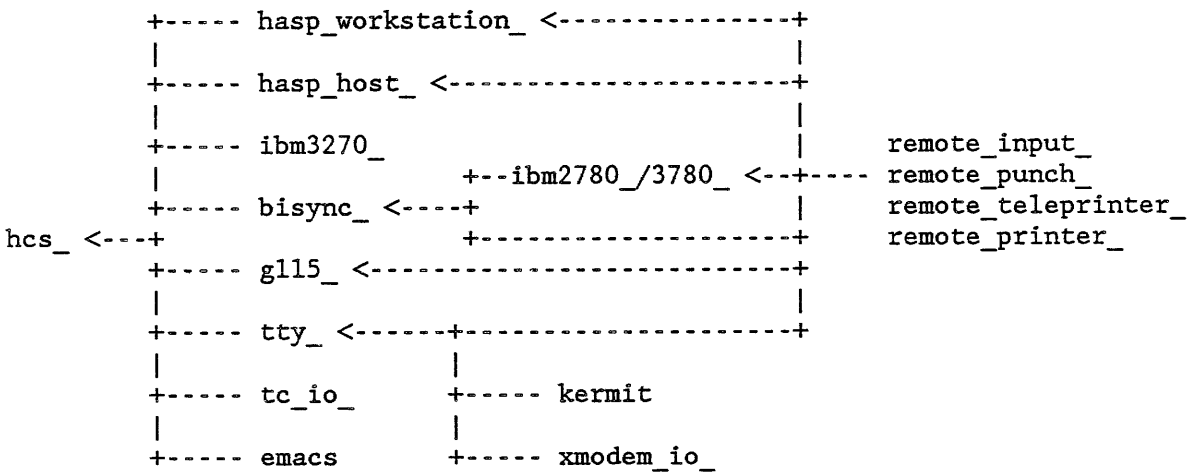
Direct Interface adapter

Transfer of data over DIA interrupts, Data Transfer
Buffer space management ~~TTY-BUF~~
Echoing
Padding sending nulls for delays, done in Ring 0 on MDT
Flow control for ASYNC Xon 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 MCS 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



---

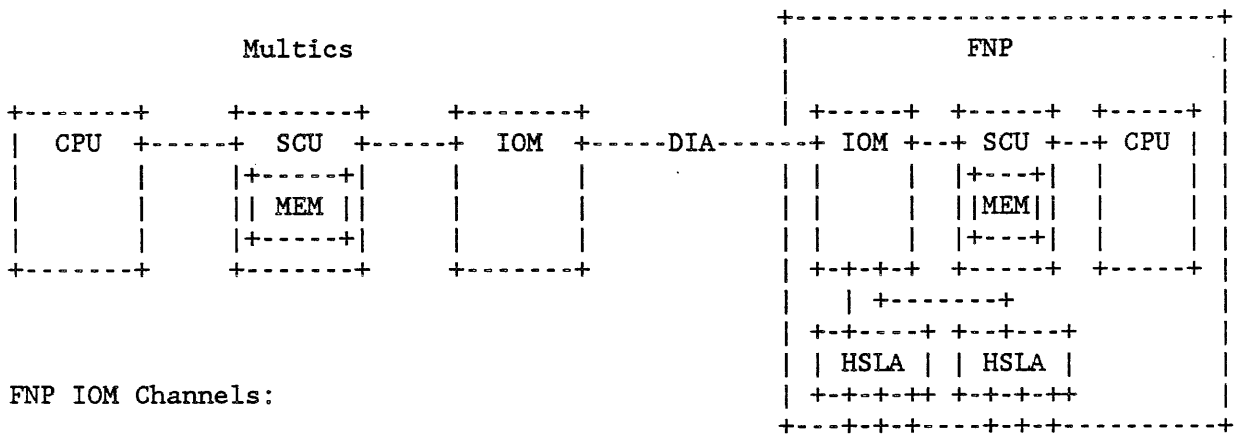
emacs - hcs directly instead of module tc_io video system

- 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
    - MPX
  - 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

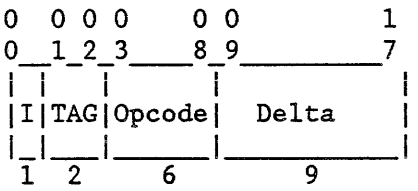


FNP IOM Channels:

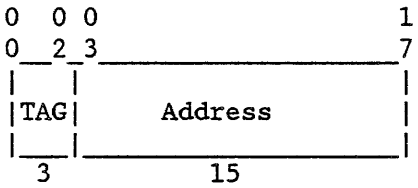
0	FNP Console
1	FNP Reader
2	FNP printer
3	
4	DIA
5	extra Dia
6	HSLA 0
7	HSLA 1
8	HSLA 2
9-14	LSLAs obsolete
15	Clock

- Data and Instruction Formats
  - 18-bit words
  - I/O commands also know about 36-bit words

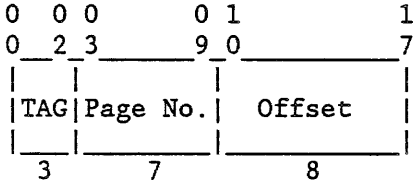
STORE REFERENCE INSTRUCTION FORMAT



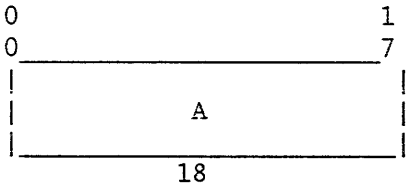
INDEX REGISTER (Xn) -- UNPAGED ADDRESS



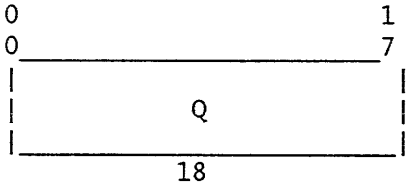
INDEX REGISTER (Xn) -- PAGED ADDRESS



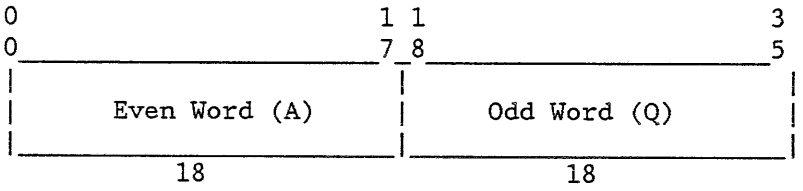
ACCUMULATOR REGISTER (A)



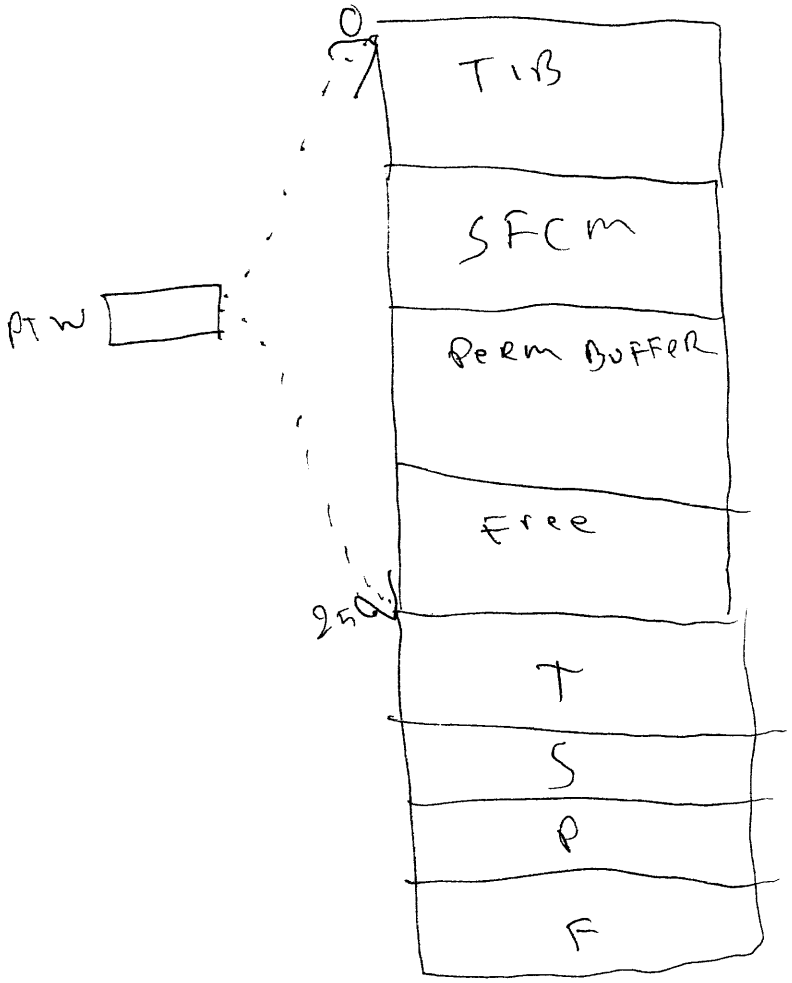
QUOTIENT REGISTER (Q)



ACCUMULATOR-QUOTIENT REGISTER (AQ)

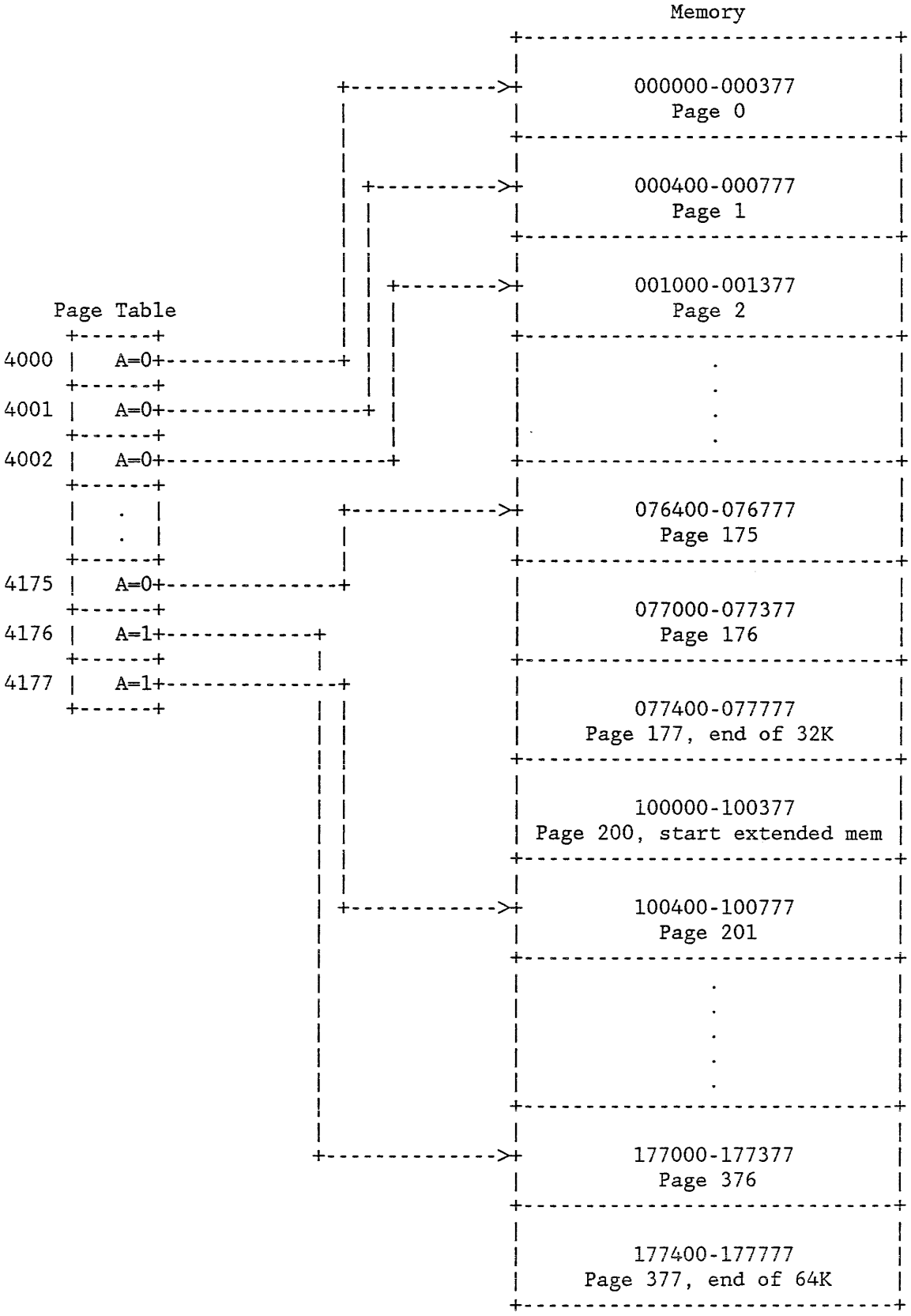


- Extended Memory Addressing
  - So-called paging is used to address beyond 32K
  - Difficult to program, so only certain types of data are stored in extended memory
  - No programs in extended memory



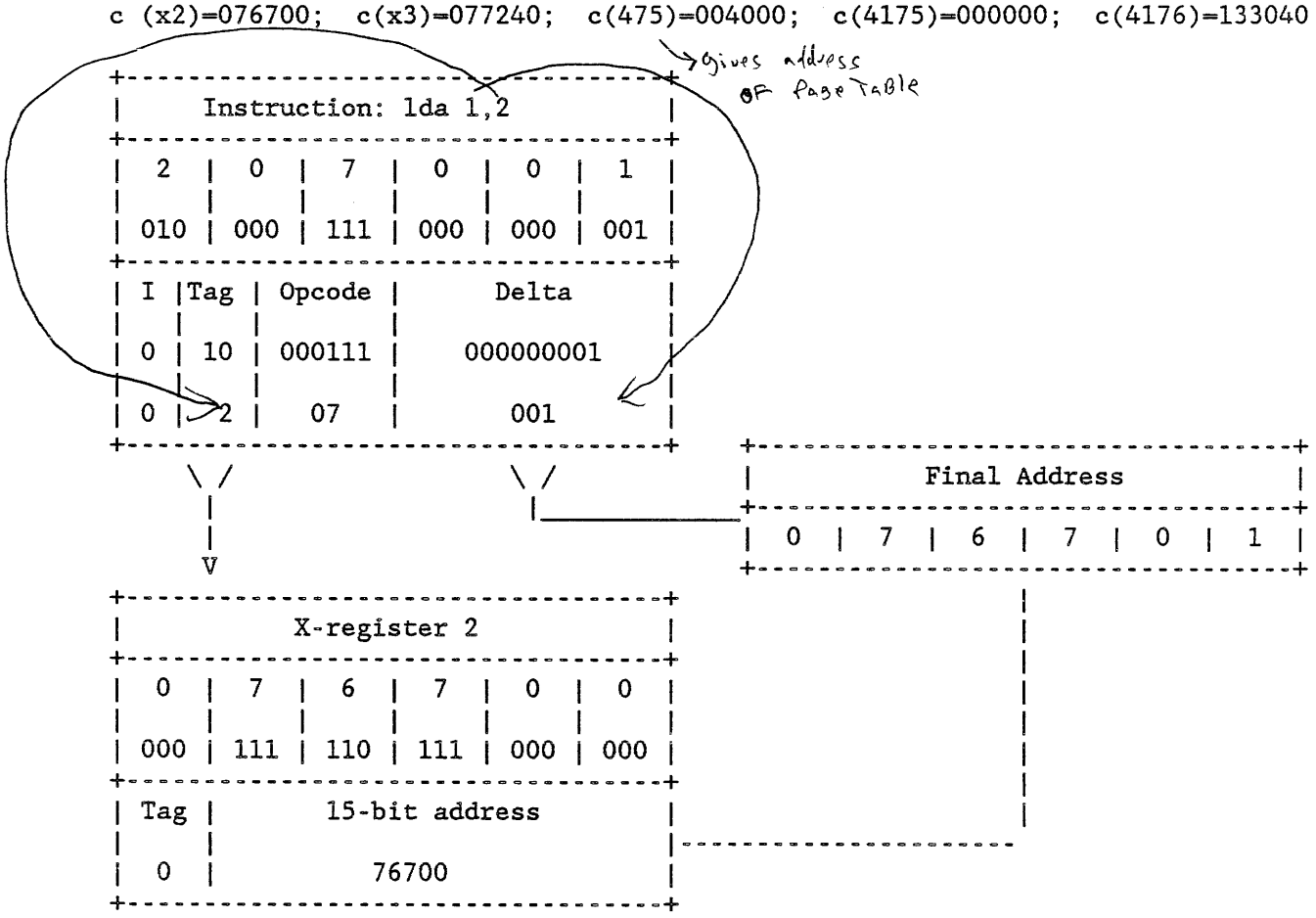
This is really how it is Done

FNP Extended Memory Addressing



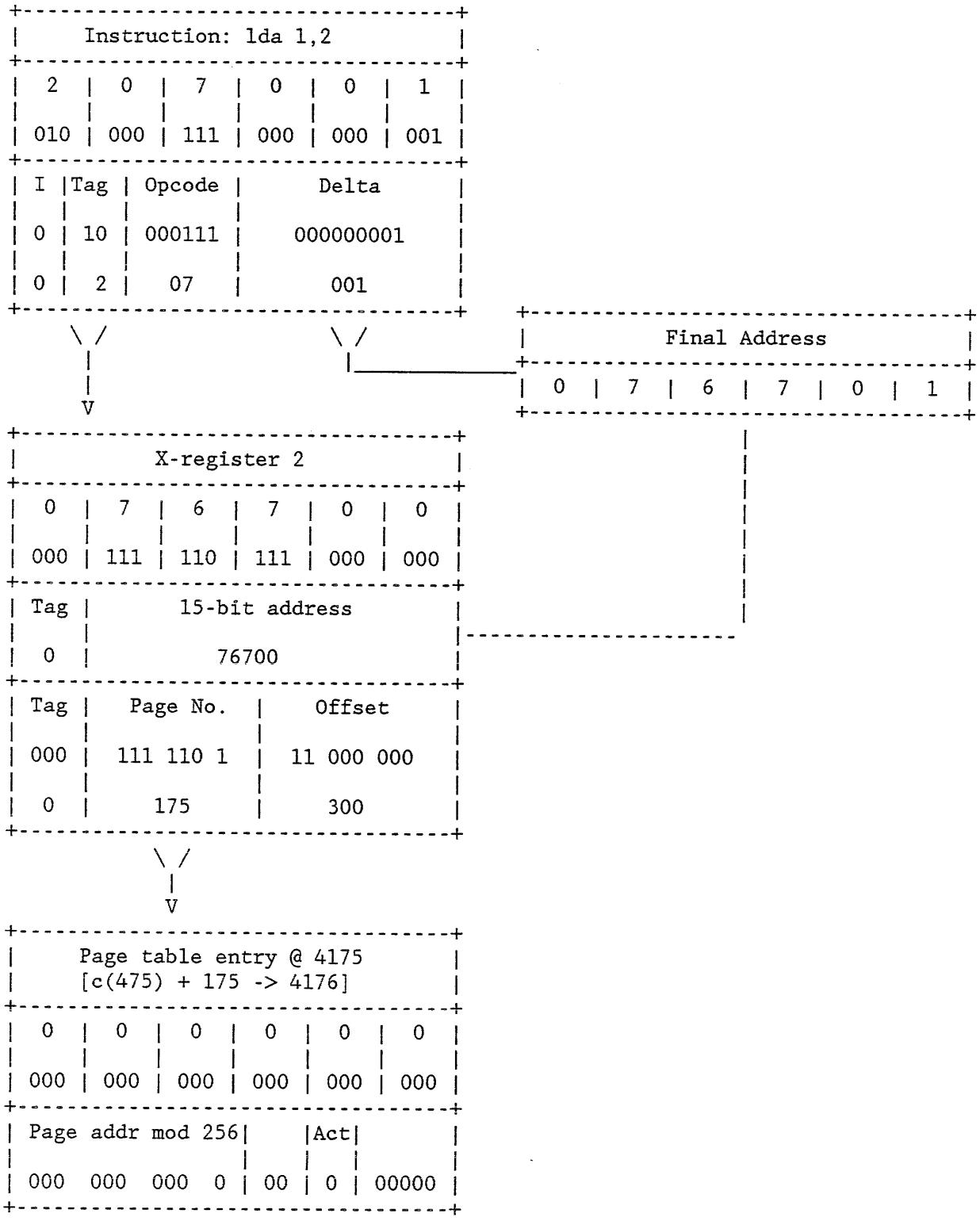
FNP Address Calculation Example 1
Reference To Address In Low-Order Memory Using Non-Paged Addressing

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



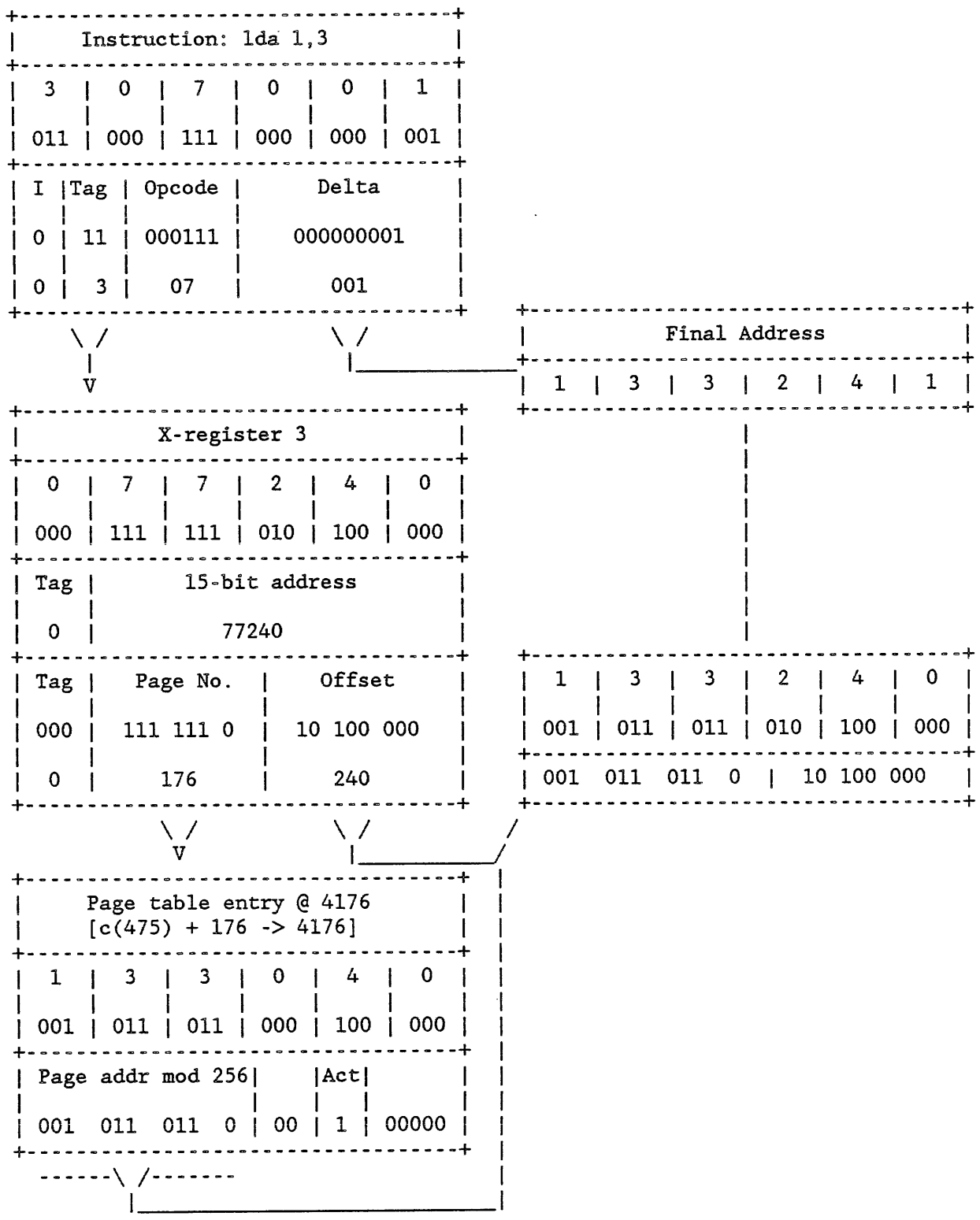
FNP Address Calculation Example 1
Reference To Address In Low-Order Memory Using Paged Addressing

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



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

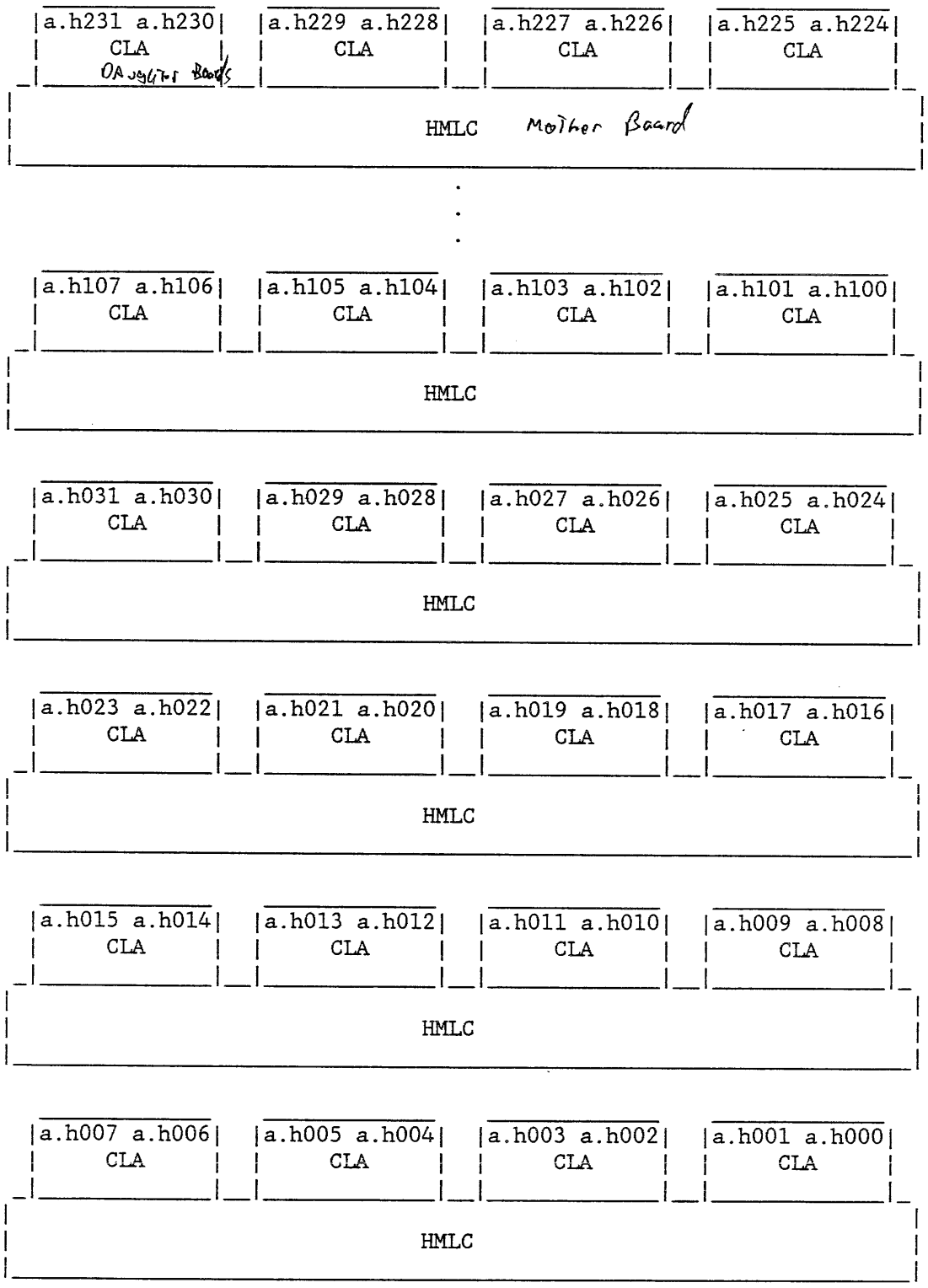


- HSLA
  - High-Speed Line Adapter
  - Emulated in 6670s
  - All communications lines are connected to HSLAs
  - Maximum of 32 channels per HSLA
  - Maximum of 3 HSLAs per FNP => maximum of 96 channels per FNP
  - HSLAs do much of the work of running a channel, freeing up the FNP's CPU
  - Set and detect 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.prt1
- 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 modems on Board To call out
    - BSC
    - G115
    - HDLC x.25 Connections only 1 channell Per Board
  - HDLC 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 => 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



- o HMLC = Mother Board
- o CLA = Daughter Board

- DIA
  - 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

FN01
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;

Check_acs:  all;

FNP:          a;
 type:        DN6670;
 lsla:        0;
 hsla:        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.h008;   comment:  "connexion stations HASP";
 baud:    4800;     line_type:  BSC;    terminal_type: HASP_HOST;
 multiplexer_type:  hasp;
 service: multiplexer;         attributes: ^hardwired;

          name: a.h008.opr;
            service:           slave;
            line_type:         BSC;
            terminal_type:     HASP_HOST;

          name: a.h008.rdr1;
            service:           slave;
            line_type:         BSC;
            terminal_type:     HASP_HOST;

          name: a.h008.prt1;
            service:           slave;
            line_type:         BSC;
            terminal_type:     HASP_HOST;

          name:  a.h008.pun1;
           service:            slave;
           line_type:          BSC;
           terminal_type:      HASP_HOST;

name:     a.h010;   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: BPS
    - 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:
    - Line protocol: ASCII (Async), G115, BSC (Bisync), VIP, POLLED_VIP, or X25LAP
    - Cannot be changed by user
  - 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.

now you can: io attach TTY_ h.hlll

    - mc
      - 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
    Also used over X.25 TO connect TO network instead of Phone #

- 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 >sc1>rcp

    - E.g. >sc1>rcp>a.h001.acs null Access *.SysDaemon

  - 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 Questionable
- 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 maybe run periodicly

```
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:                       77777|1
     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

Name	Type	No.	S	WP	A	Baud	User
a.c000	(NU)	0	5	18	2	9600	COLTS executive channel
a.h000	(NU)	0	5	0	1	1200	console pupitre
a.h001	ROSY	125	5	3	3	1200	console datanet
a.h002	AJ510	171	5	6	3	1200	AJ510 dans salle de consoles 125
a.h003	AJ860	155	5	17	1	1200	AJ860 dans salle de consoles 121
a.h006	(NU)	0	0	0	1	1200	X.25 major channel
a.h006.d01		1649	1	18	1	300	X.25 dial_out sub-channel
a.h006.d02		603	1	18	1	300	X.25 dial_out sub-channel
a.h006.d03		228	1	18	1	300	X.25 dial_out sub-channel
a.h006.d04		89	1	18	1	300	X.25 dial_out sub-channel
a.h006.d05		38	1	18	1	300	X.25 dial_out sub-channel
a.h006.001	ASCII_CAPS	2788	5	8	5	1200	Desgoutte CNIP2 (none) X.25 login sub-channel
a.h006.002	MINITEL	1790	2	1	2	1200	X.25 login sub-channel
a.h006.003	ASCII_CRT	1441	2	1	2	1200	X.25 login sub-channel
a.h006.004	ASCII_CRT	1144	2	1	2	1200	X.25 login sub-channel
a.h006.005	ASCII_CRT	910	2	1	2	1200	X.25 login sub-channel
a.h006.006	ASCII_CRT	719	2	1	2	1200	X.25 login sub-channel
a.h006.007	ASCII_CRT	587	2	1	2	1200	X.25 login sub-channel

tty_lines field	display_cdt field
Name	name:
Type	current_terminal_type:
No.	n_dialups:
S	state:
WP	tra_vec:
A	in_use:
Baud	baud:
User	comment:

FNP Images

- 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

FNP MODULES

Module Name	Needed if:	Which should happen:
dia_man	always	always
interpreter	always	always
scheduler	always	always
utilities	always	always
init	always	always deletes ITSPIF, SAVES GOES TO Buffers
control_tables	line_type: ascii;	always
hsla_man	hsla: >0;	always
trace	module: trace;	usually Slows up FNP, speed important on x.25 lines
meters	meter: yes;	usually
mclt	line_type: COLTS	usually keeps fe happy
g115_tables	line_type: g115	sometimes (Level-6 remotes)
bsc_tables	line_type: bsc;	sometimes (HASP, IBM remotes)
x25_tables	line_type: x25lap;	sometimes (real Multics x.25)
hasp_tables	multiplexer_type: hasp	sometimes (HASP)
vip_tables	line_type: vip	sometimes (Sync VIPs)
polled_vip_tables	line_type: polled_vip	sometimes (Polled sync VIPs)
ibm3270_tables	line_type: bsc; + ibm3270_	sometimes (3270 controller)
acu_tables	service: autocall;	sometimes (non-x.25 dialout)
autobaud_tables	baud: auto;	sometimes
ic_sampler	debug_fnp ic_sample	rare
breakpoint_man	debug_fnp breakpoints	rare
console_man	console: yes;	rare

multiplexing
Can be done in ring 0

or

1 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

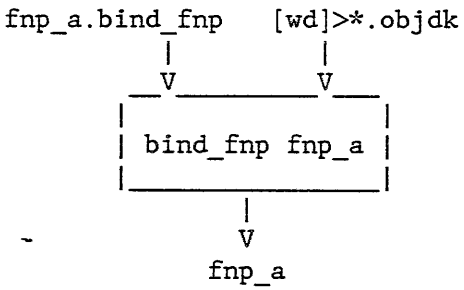
```
/* bindfile for MCS, Multics Communications System */

version:  6.5d;


lsla:     0;
hsla:     3;
memory:   64;
console:  no;
printer:  no;
meter:    yes;

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

order:    scheduler,
          interpreter,
          control_tables,
          dia_man,
          mclt,
          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;
```

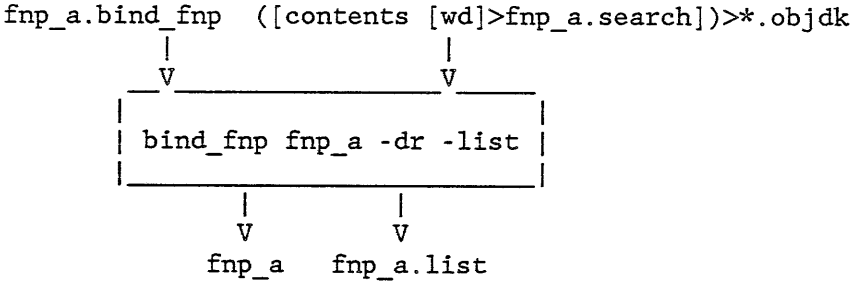
(correspond to cdt, otherwise lower Assumed.

doesn't change

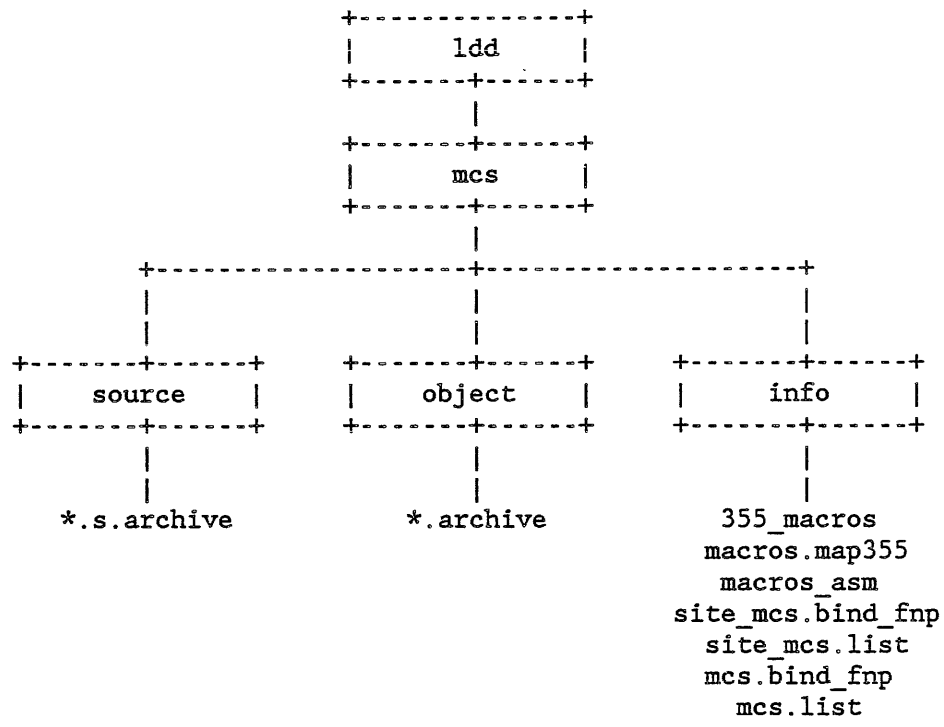
7 = OFF

bind_fnp

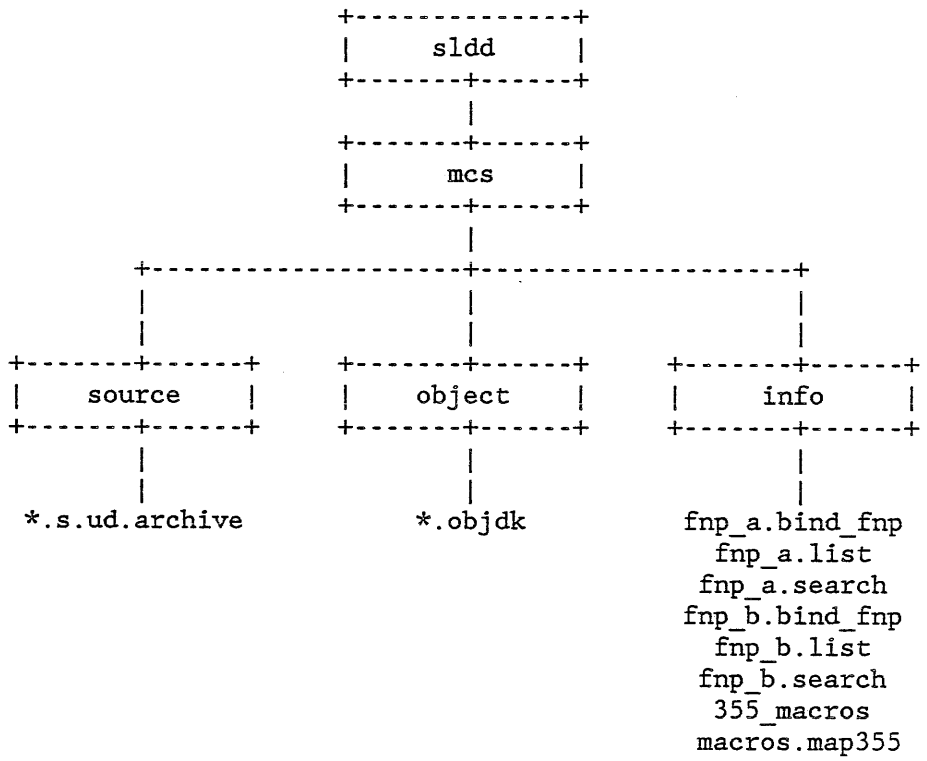

fnp_a.bind_fnp [wd]>*.objdk
bind_fnp fnp_a
fnp_a


fnp_a.bind_fnp ([contents [wd]>fnp_a.search])>*.objdk
bind_fnp fnp_a -dr -list
fnp_a
fnp_a.list

LDD Structure



SLDD Structure



| 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 -truncate {-list -lower_case}.

| macros.map355 -> 355_macros

⊠ Attributes in SAT/PDT

- dialok slave + dial lines
- save_on_disconnect
- disconnect_ok

◙ Config deck

| PRPH FNP A A 18. ON

| PARM TTYB 10240.

| Default = 6K

▣ 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 attibutes: hardwired

| device_prices, device_names

| Used in conjuction 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
    - stty
    - io_call control, io_call modes
    - iox_$control, iox_$modes
  - We will look at how to modify them in per terminal-type in TTF and dynamically using iox_
    - stty and io_call are just command-level interfaces to 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 RB 1, 5, 5
- 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

```
Input_conversion: standard_input_conv;

/***** Typical ASCII teleprinter terminal *****/
terminal_type: ASCII;
 modes: default,1179,^p1,can_type=overstrike,hndlquit,
        fulldpx,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,^p1,can_type=overstrike,hndlquit,capo,
        fulldpx,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,
        fulldpx,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,
        fulldpx,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,
        fulldpx,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
    - 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;
 000 001 002 003 004 005 006 007 /*NUL SOH STX ETX EOT ENQ ACK BEL (000-007)*/
 010 011 012 013 014 015 016 017 /*BS  TAB LF  VT  FF  CR  SO  SI  (010-017)*/
 020 021 022 023 024 025 026 027 /*DLE DC1 DC2 DC3 DC4 NAK SYN ETB (020-027)*/
 030 031 032 033 034 035 036 037 /*CAN EM  SUB ESC FS  GS  RS  US  (030-037)*/
 040 041 042 043 044 045 046 047 /*SP  !   "   #   $   %   &   '   (040-047)*/
 050 051 052 053 054 055 056 057 /*(   )   *   +   ,   -   .   /   (050-057)*/
 060 061 062 063 064 065 066 067 /*0   1   2   3   4   5   6   7   (060-067)*/
 070 071 072 073 074 075 076 077 /*8   9   :   ;   <   =   >   ?   (070-077)*/
 100 141 142 143 144 145 146 147 /*@   A   B   C   D   E   F   G   (100-107)*/
 150 151 152 153 154 155 156 157 /*H   I   J   K   L   M   N   O   (110-117)*/
 160 161 162 163 164 165 166 167 /*P   Q   R   S   T   U   V   W   (120-127)*/
 170 171 172 133 134 135 136 137 /*X   Y   Z   [   \   ]   ^   _   (130-137)*/
 140 141 142 143 144 145 146 147 /*`   a   b   c   d   e   f   g   (140-147)*/
 150 151 152 153 154 155 156 157 /*h   i   j   k   l   m   n   o   (150-157)*/
 160 161 162 163 164 165 166 167 /*p   q   r   s   t   u   v   w   (160-167)*/
 170 171 172 173 174 175 176 177;/*x   y   z   {   |   }   ~   DEL (170-177)*/
```

Changes all characters To Lower case

Extract from tty_convert.incl.pl1
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 — what u see is what u get approach
  - Backspaces present a particular problem on input
  - When we see the string abc in a file we can't know if it was originally typed in
    - abc<BS><BS><BS>___
    - a<BS>_b<BS>_c<BS>_
    - Or even ab<BS>_<BS><BS>a<BS>___<SP><BS><BS>c
  - Similarly if we see the line abc on a CRT screen it may have been typed an infinite number of different ways
    - 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
  - 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 same column are put in sorted order, separated by backspaces
    - In the example above, canonical form is _<BS>a_<BS>b_<BS>c
  - 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
    - ab<SP><SP><BS>c and ab<SP><SP>#c give different results
  - 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.pl1
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 null & Pad chars

- 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
- esc mode
  - Can be turned off to inhibit escape sequence processing

4 steps
Translation
Canonicalize
handle erase & kill
Process escape chars

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 /* (   )   *   +   ,   -   .   /   (050-057) */
 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 03;/* x   y   z   {   |   }   ~   DEL (170-177) */
```

Extract from tty_convert.incl.pl1
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";
```

Translates \a into A

Extract from tty_convert.incl.pl1
(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
  - rawi mode video works that way
- 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~~ convert all alphabetics to uppercase
        Not tol translation
  - Formatting
    - ll mode
      - ^ll or llNN, e.g. ll79
      - 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 pl 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 desirable at low speeds
  - 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 RRS, 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 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 /* DLE DC1 DC2 DC3 DC4 NAK SYN ETB (020-027) */
 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 /* SP  !   "   #   $   %   &   '   (040-047) */
 00 00 00 00 00 00 00 00 /* (   )   *   +   ,   -   .   /   (050-057) */
 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 00 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 14;/* x   y   z   {   |   }   ~   DEL (170-177) */
```

Extract from tty_convert.incl.pl1
Used for get_output_conversion and set_output_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 (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:        ;
 printer_off:       ;
 red_shift:         ;
 black_shift:       ;
 end_of_page:       E O 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";
```

Password masking while not in echo mode if terminal supports local echoing

Extract from tty_convert.incl.pl1
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
      (special_chars_struc.escape_length)) like c_chars,
      3 edited_escapes (sc_escape_len refer
      (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
        (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;
```

- 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 (EMACS)

- 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 (ie: which data came from which terminal
    - 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 crecho
- tabecho mode
  - 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 crecho

- Quit Handling
  - There are a number of steps in handling break condition
    - Handled both in ring 0 and in FNP & Ring 4 in Process
  - 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 quits 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
    - IF quits enabled
    - 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

stty -modes ~~replay~~ force, replay, polite, prefixnl
on x25 ↑ ↑
non x25 sets all 3

| 2) PAD/X.25 network tell FNP (via control packet)

On network Break CR seems to Be faster

| 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
IF quit enable

| 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

- 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 not documented
  - 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.pl1
Used for set_wakeup_table control order

```
dcl  swt_info_version_1 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: ...
        .
        .
 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     ...;
```

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
  - oflow mode
    - Turns on either of the two types of output flow control
      - ETB/ACK output flow control
      - Suspend-Resume output flow control

End-of-text

ETB-ACK Output Flow Control Protocol

Sender	Characters sent	Meaning
Multics:	thisisETB	Here's a block of output for you.
	(Silence while Multics waits for OK to resume.)	
Terminal:	ACK	OK, I can take another block of up to 256 characters.
Multics:	theoutETB	Here's a block of output for you.
	(Silence while Multics waits for OK to resume.)	
Terminal:	ACK	OK, I can take another block of up to 256 characters.
Multics:	putforETB	Here's a block of output for you.
	(Silence while Multics waits for OK to resume.)	
Terminal:	ACK	OK, I can take another block of up to 256 characters.

.
.
.
.

Suspend/Resume Output Flow Control Protocol

Sender	Character sent	Meaning
Multics:	t	Here's a character for you.
Multics:	h	Here's a character for you.
Multics:	i	Here's a character for you.
Multics:	s	Here's a character for you.
Multics:	i	Here's a character for you.
Multics:	s	Here's a character for you.
Terminal:	X-OFF	Wait, I can't handle any more for the moment.
	(Silence while Multics waits for OK to resume.)	
Terminal:	X-ON	OK, I can handle some more now.
Multics:	t	Here's a character for you.
Multics:	h	Here's a character for you.
Multics:	e	Here's a character for you.
Multics:	o	Here's a character for you.
Multics:	u	Here's a character for you.
Terminal:	X-OFF	Wait, I can't handle any more for the moment.
Multics:	t	Here's a character for you.
	(Silence while Multics waits for OK to resume.)	
Terminal:	X-ON	OK, I can handle some more now.
Multics:	p	Here's a character for you.
Multics:	u	Here's a character for you.

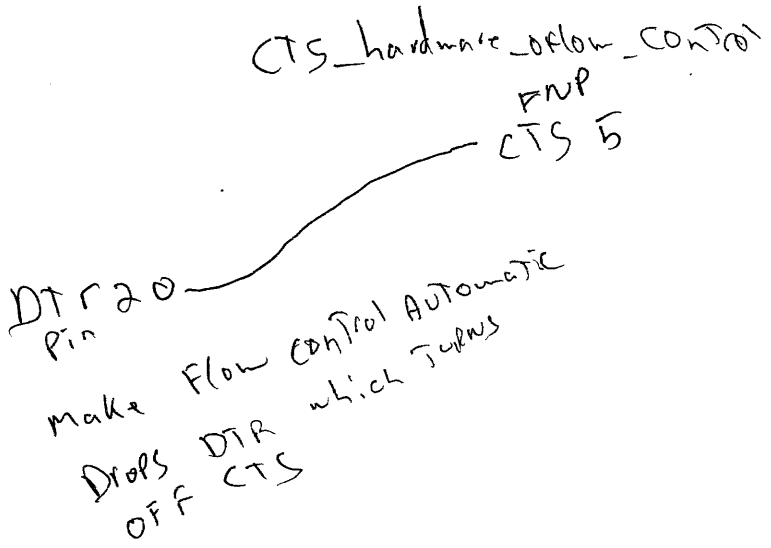
.
.
.
.

Output Flow Control
Extracts From Terminal Type File

```
output_suspend: ^S;                    output_end_of_block: ETX;
output_resume: ^Q;                     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);
```



- 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

Sender	Character sent	Meaning
Terminal:	t	Here's a character for you.
Terminal:	h	Here's a character for you.
Terminal:	i	Here's a character for you.
Terminal:	s	Here's a character for you.
Terminal:	i	Here's a character for you.
Terminal:	s	Here's a character for you.
Multics:	X-OFF	Wait, I can't handle any more for the moment.
	(Silence while Multics waits for OK to resume.)	
Multics:	X-ON	OK, I can handle some more now.
Terminal:	t	Here's a character for you.
Terminal:	h	Here's a character for you.
Terminal:	e	Here's a character for you.
Terminal:	i	Here's a character for you.
Terminal:	n	Here's a character for you.
Multics:	X-OFF	Wait, I can't handle any more for the moment.
Terminal:	p	Here's a character for you.
	(Silence while Multics waits for OK to resume.)	
Multics:	X-ON	OK, I can handle some more now.
	(Silence for one second)	
Multics:	X-ON	I repeat, I can handle some more now.
Terminal:	u	Here's a character for you.
Terminal:	t	Here's a character for you.

.
.
.
.

Input Flow Control
Extract From Terminal Type File

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

Extract from flow_control_info.incl.pl1
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 Change in MR12.2

| 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

device index

| Can get devx to use in direct hcs_ calls

ring 0 doesn't understand IOCB's, so you call get_chan_info to get index

| 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

login
to attach x tty_ -dial_id Foo
dial foo Homan.multics

⊠ 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 chan 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 more 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 (Single data stream)

| 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

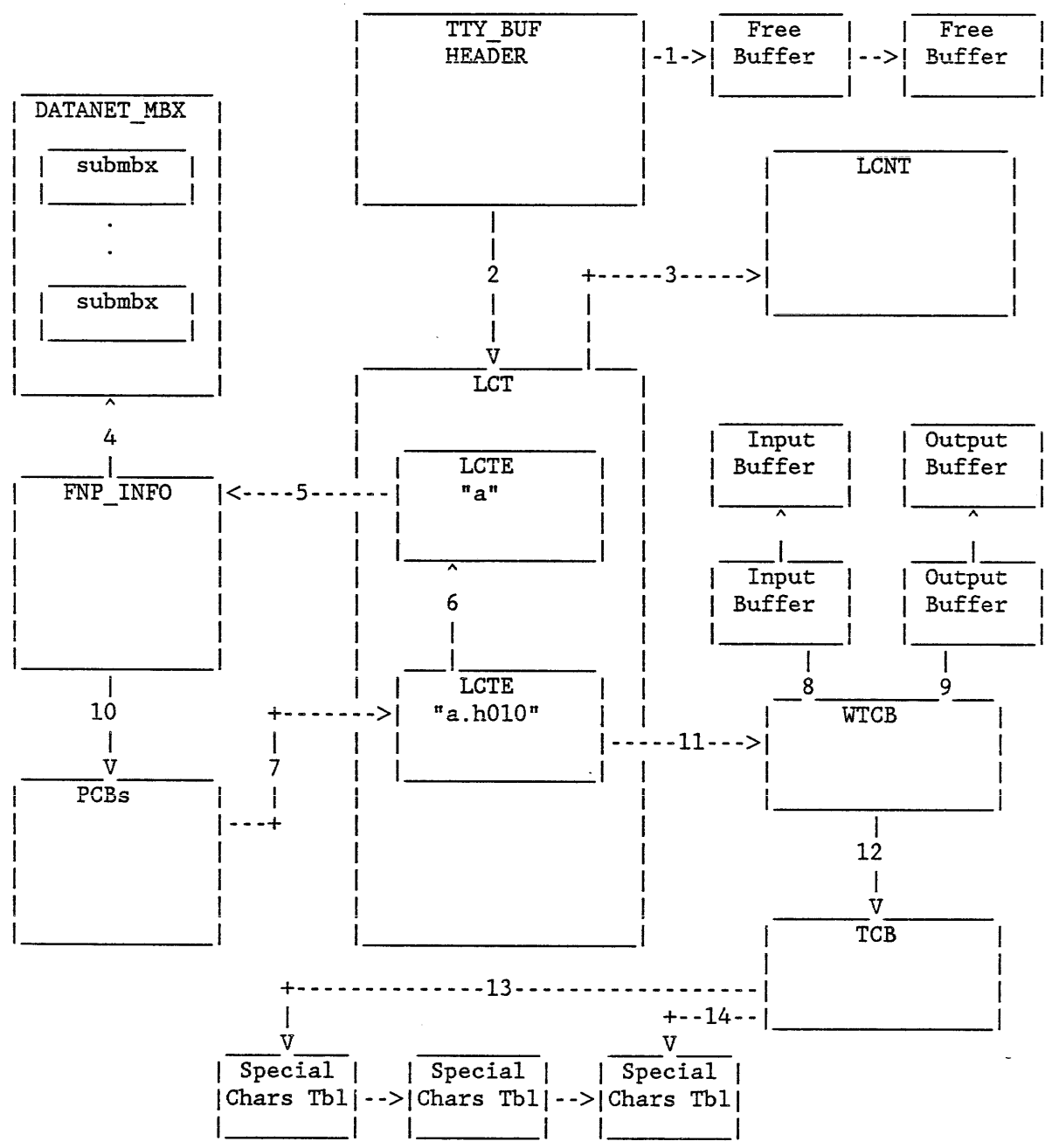
- Contains tables used during input/output conversion and translation
  - Input translation
  - Output translation
  - Input conversion
  - Output conversion
  - Special
  - Delay
- Tables are shared among lines (and hence among processes)
- Often many lines use same tables
- Reference count tells when 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

Ring Zero MCS Databases



1) tty_buf.free
2) tty_buf.lct_ptr
3) lct.lcnt_ptr
4) fnp_info.mbx_ptr
5) lcte.data_base_ptr
6) lcte.major_channel_devx
7) pcb_array(n).devx
8) wtcb.fblock
9) wtcb.write_first
10) fnp_info.pcb_array_ptr
11) lcte.data_base_ptr
12) wtcb.tcb_ptr
13) tcb.specialrp
14) tcb.df_specialrp

- 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

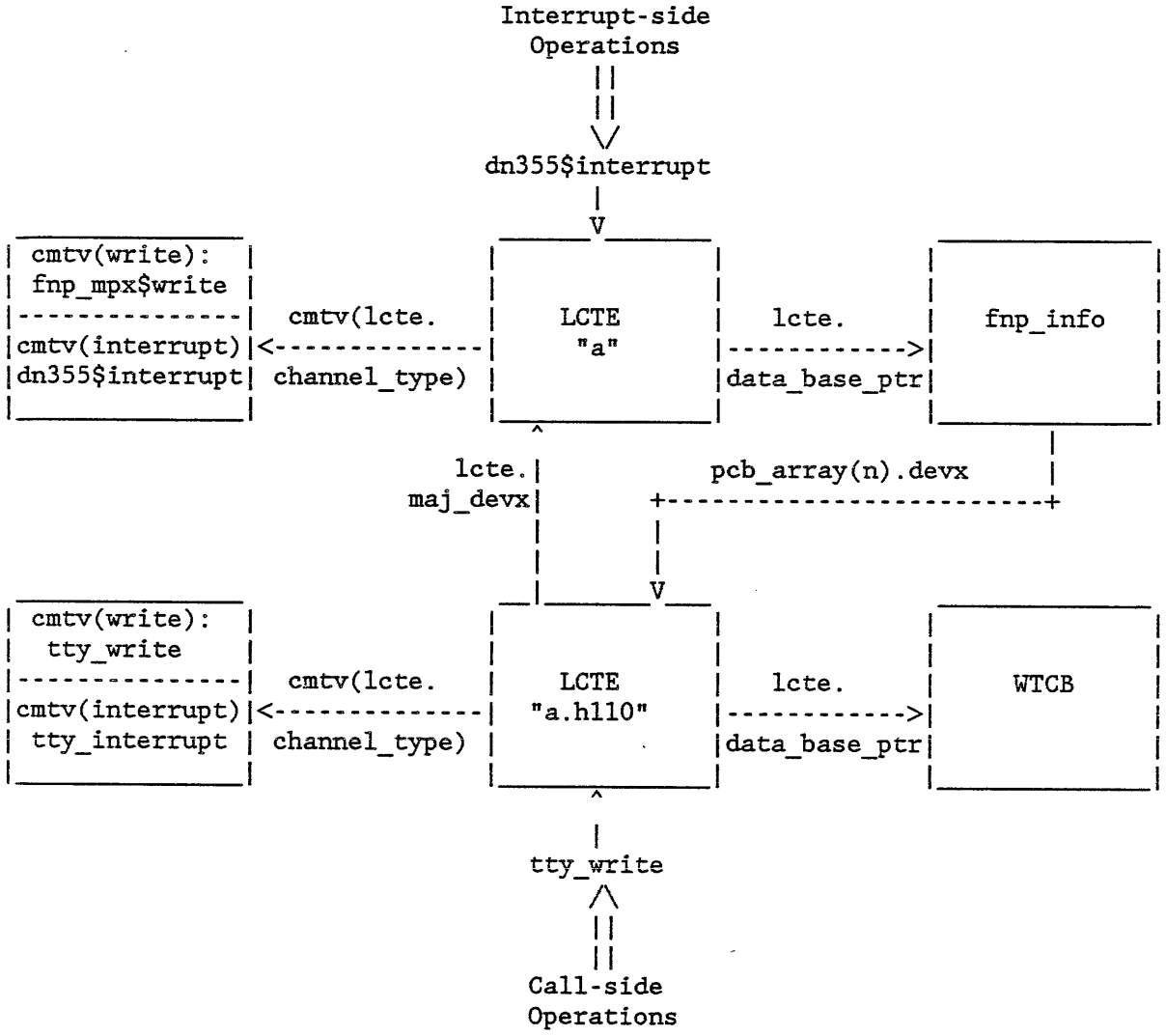
| Copies converted data from tty_buf into user ring buffer

| tty_space_man

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

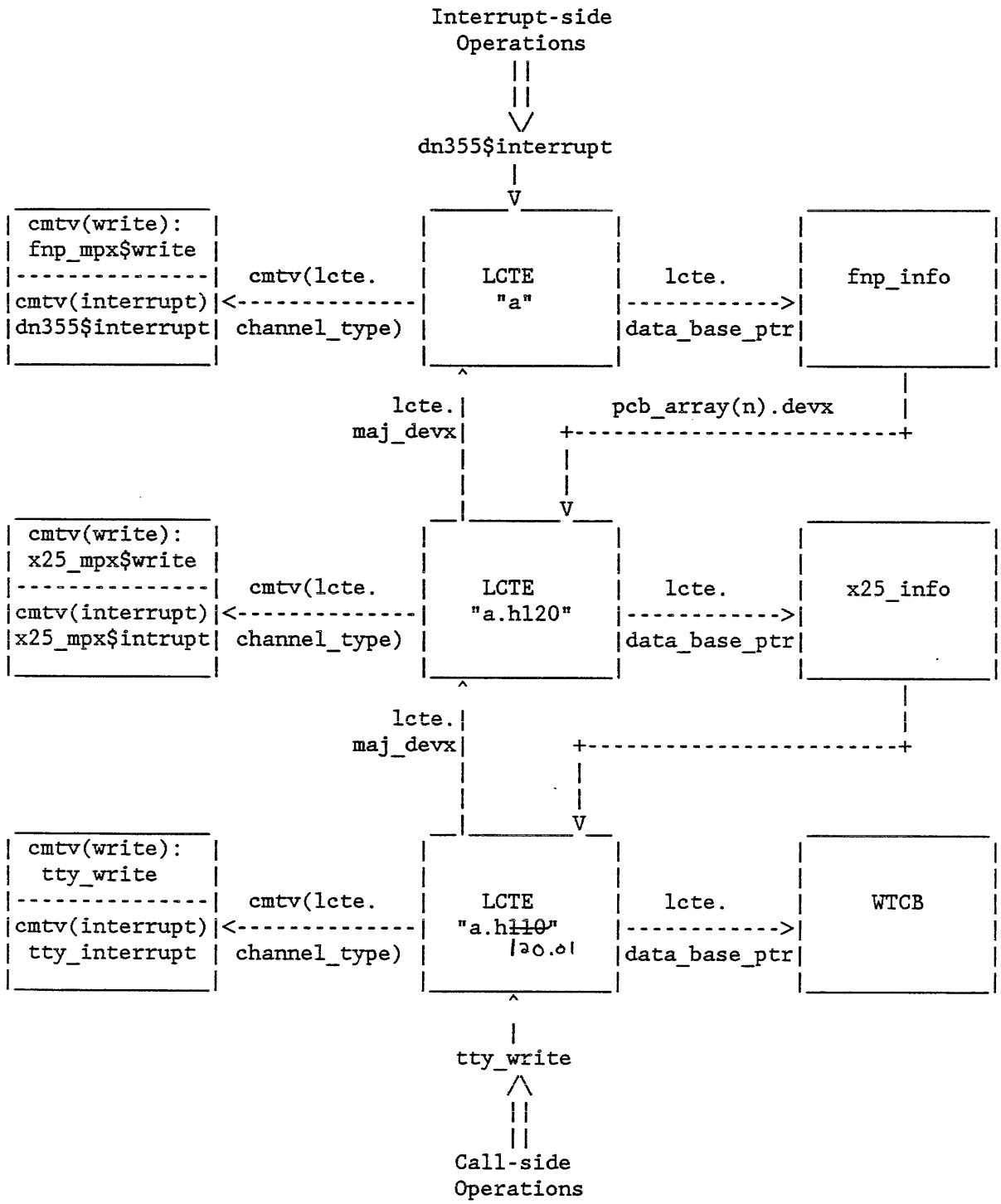
- Multiplexing
  - Connections just described between
    - dn355 and tty_interrupt on interrupt side
    - tty_read/write/index and fnp_multiplexer on call side
  - are not always direct
  - For multiplexed lines, some intermediate processing is needed
  - Adding control information to blocks of data 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

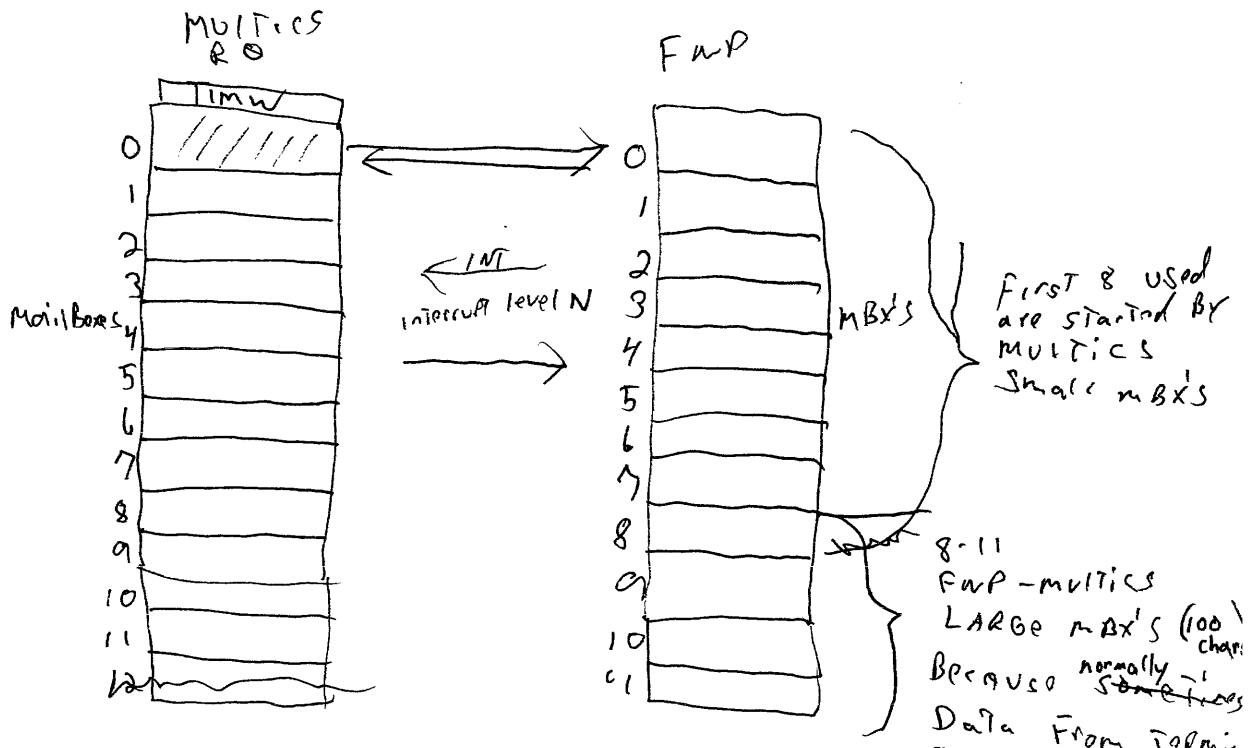


LCTE Chaining for Multiplexing

X.25 Subchannel



- DIA Interrupts
- Submailboxes
- TIMW

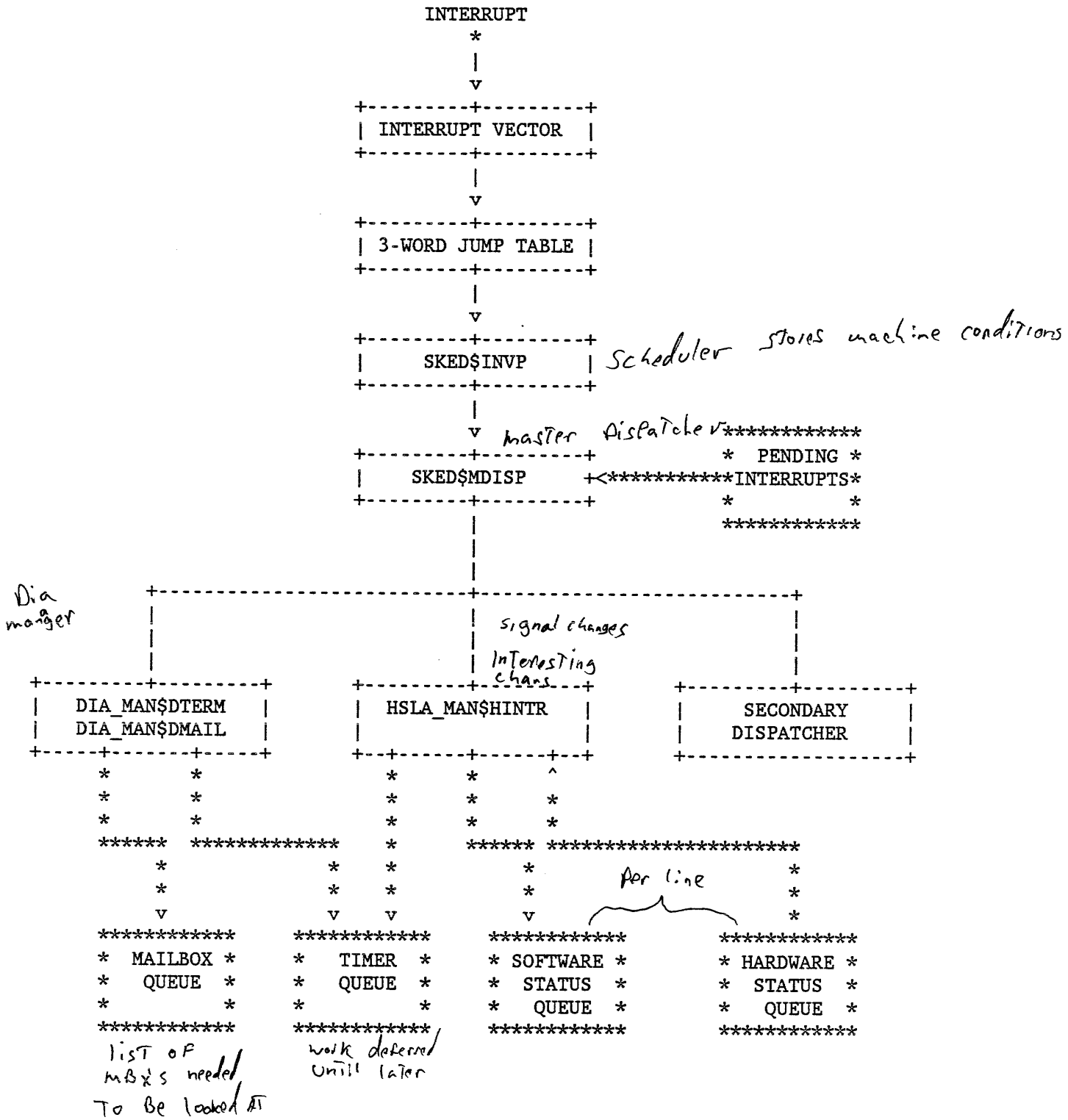


- Programs
  - All work in the FNP is driven by interrupts
    - HSLA
      - Input from a line
      - Status change for a line
    - DIA
      - Mailbox interrupt from Multics
      - Terminate interrupt from I/O started by FNP
    - Timer
  - No work ever seems to be done directly
    - Everything that needs to be done is put in a queue to be done later as time permits

FNP WORK SCHEDULING

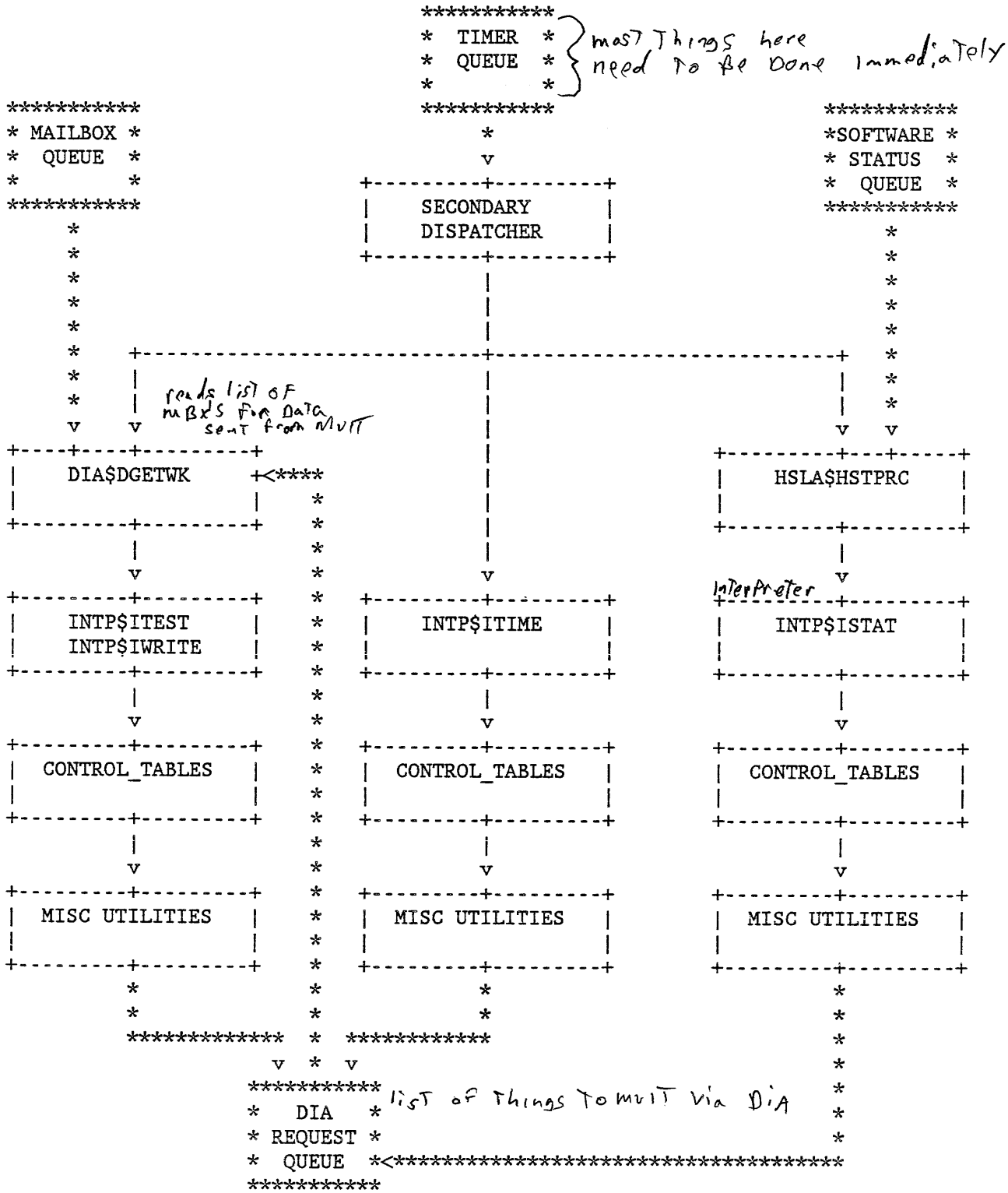
MASTER DISPATCHER--INTERRUPT-DRIVEN

names restricted To 4 chars



FNP WORK SCHEDULING

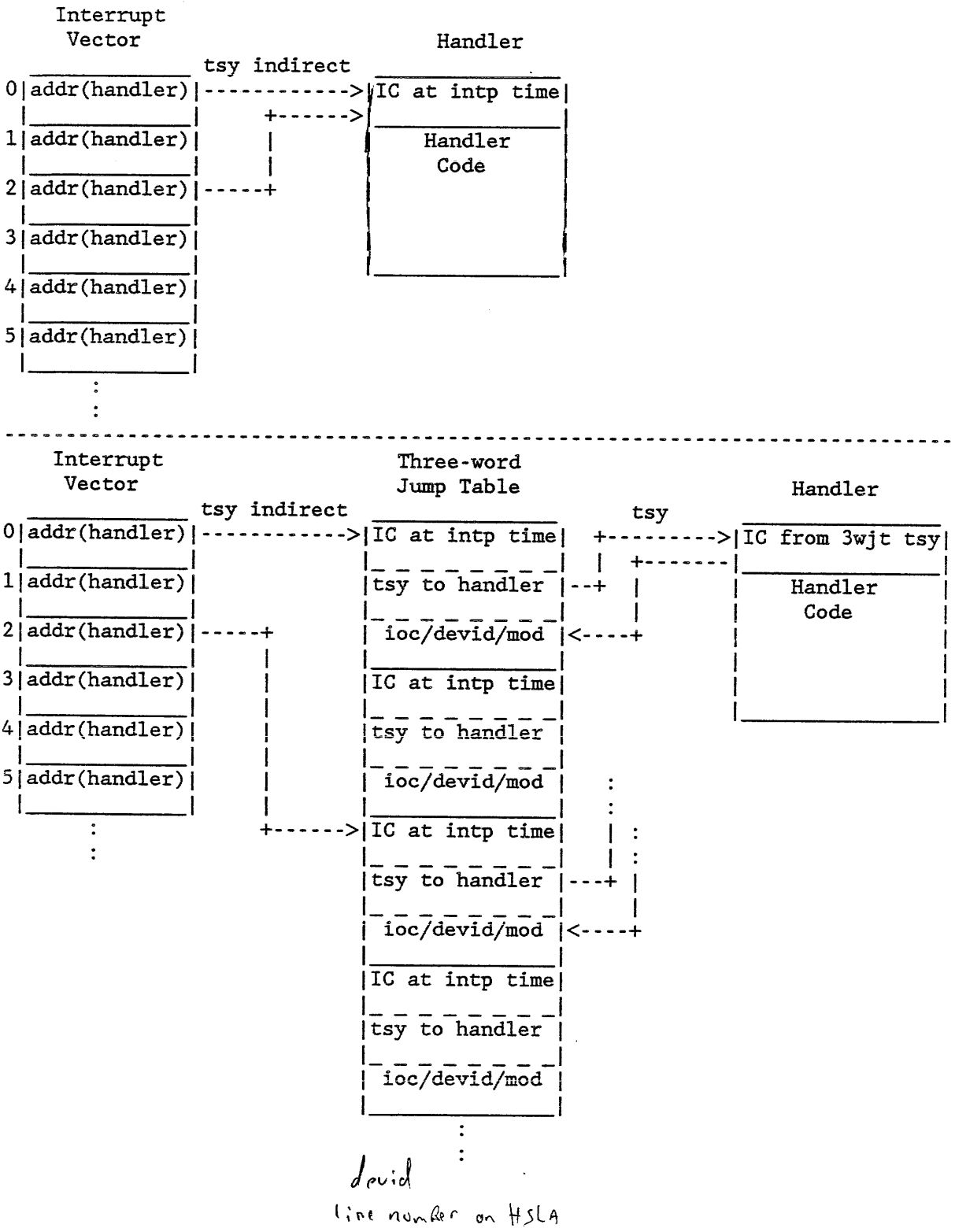
SECONDARY DISPATCHER--QUEUE DRIVEN



hardware software manual
DD 01 describes instruction set + interrupts

Three-Word Jump Table

sort of extended interrupt handler



| 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
```

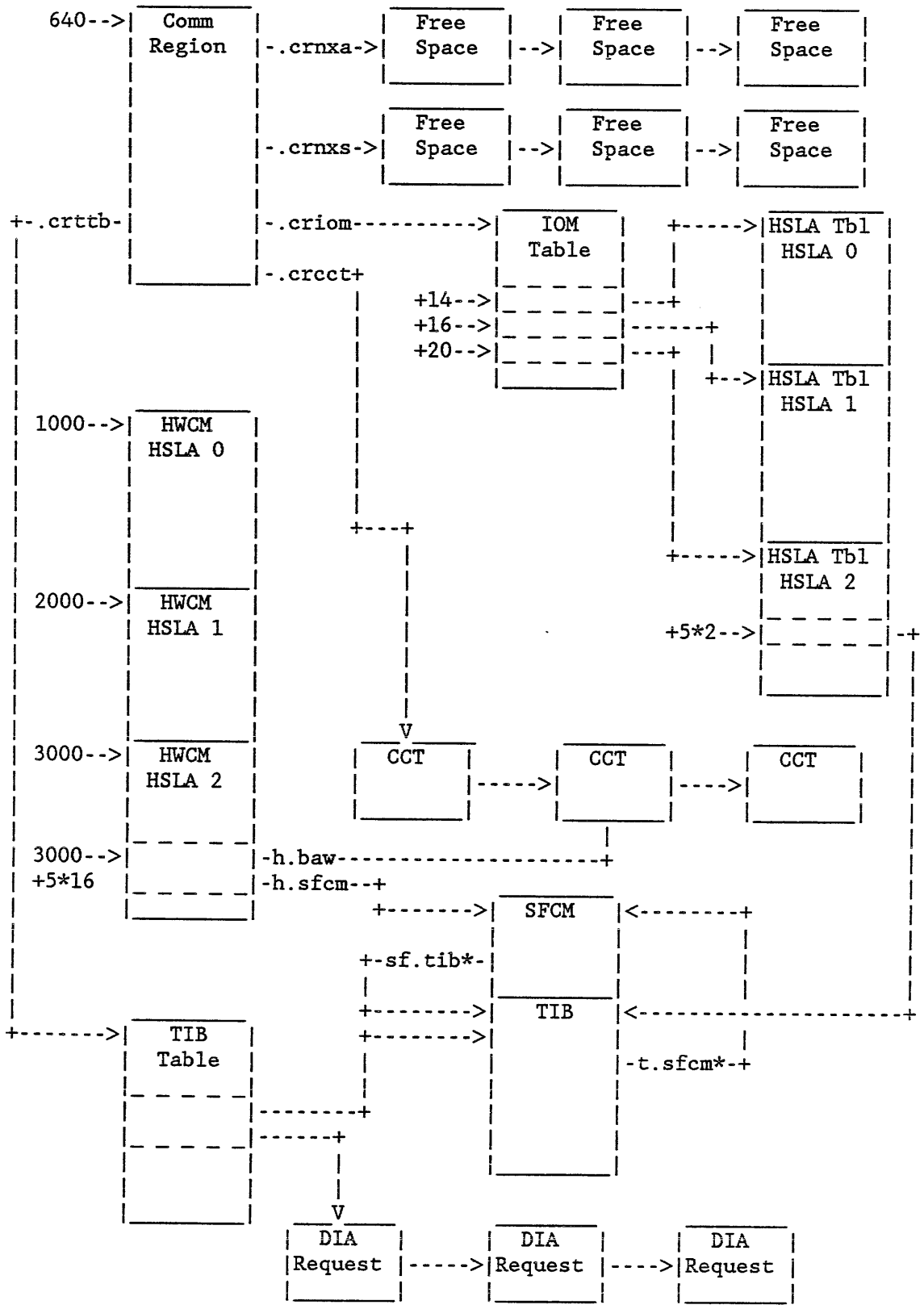
- Databases
  - Communications Region
    - Global database for FNP
    - Pointers to other databases in FNP
  - IOM Table
    - Used during initialization and to refind HSLA Table to find TIB
  - TIB
    - Terminal Information Block
    - Main per-channel database
    - Contains data used by many programs
    - Stored in extended memory
    - TIB, SFCM and two buffers permanently allocated to channel are stored in a 256-word page
    - Page is addressed using page table entry 4177 (addresses 77400-77777)
  - SFCM
    - Software Communications Region
    - Per-channel
    - Contains information for HSLA for channel
  - HWCM
    - Hardware Communications Region
    - Per-channel
    - Contains control words for HSLA
    - Pointer to CCT for channel
      - CCTs are shared, similar to tables in tty_tables
    - Pointer to SFCM for channel
    - Stored in low-order memory, address known by hardware
  - 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

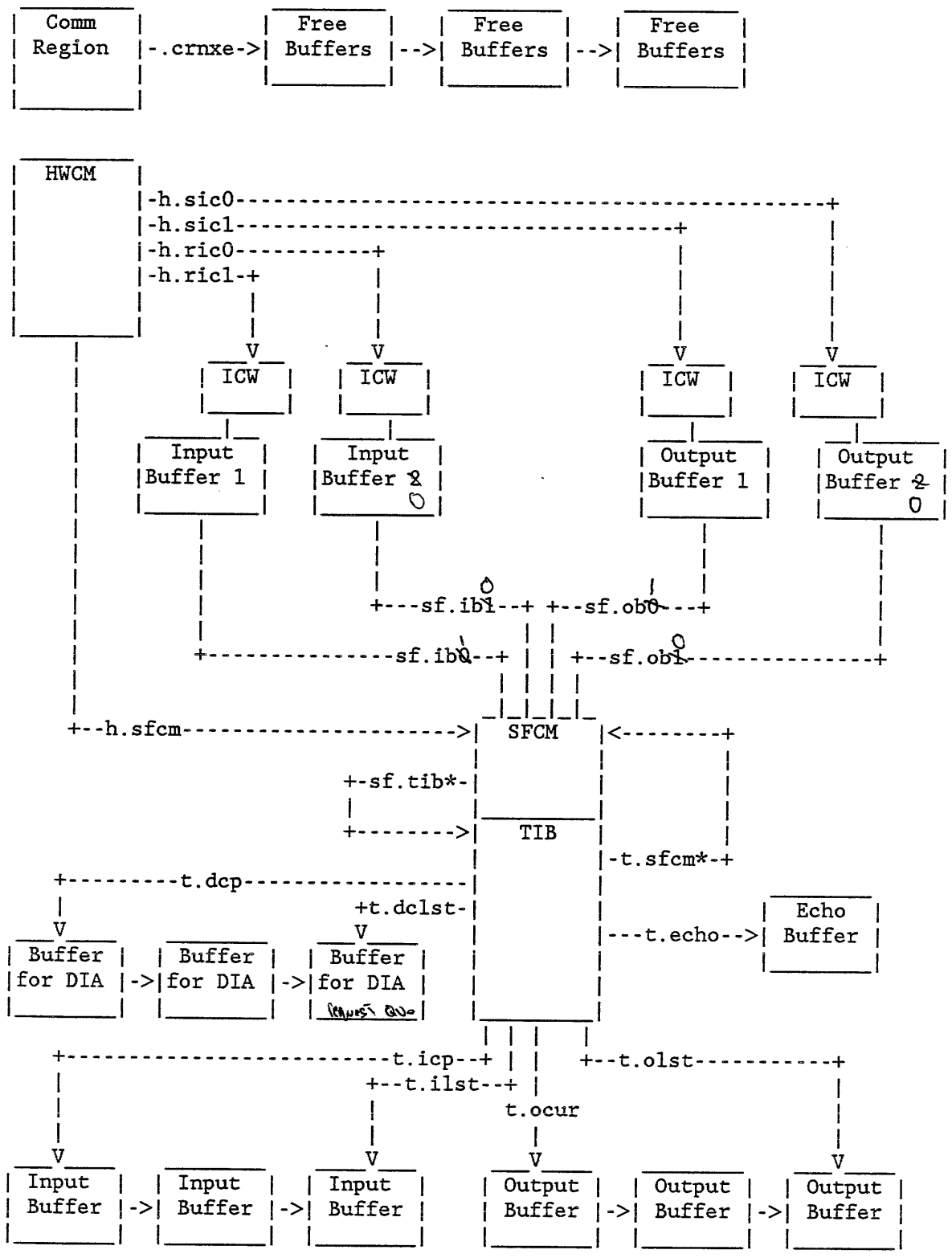
- Buffers
  - Are mainly allocated in extended memory
  - Addressed using page table entry 4176 (addresses 77000-77377)
  - Free space chained from .crnxe
  - If no free space in extended memory, can use free space in low-order memory
    - Normally used for CCTs, DIA requests, etc.
  - Free space in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



Entry

FNP I/O Buffers



higher line speed The Bigger The Buffer

FNP MEMORY LAYOUT

Start	Length	Explanation
0	420	Interrupt Vectors
420	20	Fault Status Words
440	20	Fault Vector
450	20	Mailboxes
475	1	Page Table Address
500	140	LSLA HWCMs
640	136	Communication Region
776	2	Missing Module Code
1000	#hslas*1000	HSLA HWCMs
.crcpt	200	CPU Page Table
.criom	40	IOM Table
.criom+15,*	#hslas*100	HSLA Tables
.crmod-22	variable	Program modules
.crttb	#lines*2	TIB Table
.crbuf (=.crtte)	variable	Low memory buffer pool (for TIB extensions, CCTs, DIA queue, delay tables)
77000	77777	Unused Pages
100000	#lines*400	SFCM/TIB pairs, I/O Buffer pool
100000+ #lines*400	variable	I/O Buffer pool
.crtrb	.crmem-.crtrb	Trace Buffer

```
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;
     exptextp = 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,
          exptext.data);
end;

end display_fnp_symbols;
```

Communications Region

Name	Address	Length	Explanation
.crldt	640	4	date and time of binding
.crbdt	644	4	date and time of bootloading
.crbuf	650	1	starting address of buffer area
.crmem	651	1	last location of memory
.crnbf	652	1	number of buffers available
.criom	653	1	start of iom table
.crnhs	654	1	number of hsla's configured
.crnls	655	1	number of lsla's configured
.crcon	656	1	console enabled flag
.crmod	657	1	starting address of module chain
.crnxa	660	1	ptr to next avaliable buffer
.crtra	661	1	trace entry enable mask
.crtrb	662	1	base address of trace table
.crtrc	663	1	next available location in trace table
.crreg	664	1	disaster fault register storage location
.crttb	665	1	location of tib table
.crtte	666	1	location of end of tib table
.crdly	667	1	head of delay table chain
.crver	670	2	mcs version number, 4 chars
.crbrk	672	1	addr of breakpoint control table
.crtsw	673	1	if non-zero, tracing will cease
.crnxs	674	1	next free small block
.crnbs	675	1	number of buffers devoted to small space
.crcct	676	1	address of first cct descriptor
.crskd	677	1	address of scheduler data block
.cretb	700	1	list of echo-negotiation bit tables
.crcpt	701	1	address of cpu page table
.crpte	702	1	address of variable cpu page table entry
.crtsz	703	1	size of trace data buffer
.crmet	704	1	non-zero if metering enabled
.crtdt	705	1	address of tib for t&d executive channel
.crbtm	706	1	address of time meters for getbuf/frebuf
.crnxe	707	1	next available space in extended memory
.crbpe	710	1	buffer paging window table entry
.crcpr	760	14	copyright notice

TIB

Name	Address	Length	Explanation
t.stat	0	1	holds current line status
t.flg	1	1	flag word
t.flg2	2	1	second word of flags
t.cur	3	1	current address in control table
t.line	4	1	10 bit line number
t.icp	5	1	first buffer in input chain
t.ilst	6	1	last buffer in input chain
t.icpl	7	1	count of buffers in icp chain
t.icch	10	1	address of next input character
t.elnk	11	1	link to tib extension
t.rcp	11	1	replay chain pointer (share t.elnk)
t.ocp	12	1	addr of output chain from cs
t.ocur	13	1	addr of current output buffer
t.olst	14	1	addr of last buffer in output chain
t.occh	15	1	addr of next output character
t.ocnt	16	1	count of buffers in t.ocur
t.type	17	1	line type
t.time	20	2	time at which next timeout will occur
t.reta	22	1	return address from calsub
t.dcwa	23	1	addr of dcw list to 'execute'
t.dcwl	24	1	length of dcw list
t.echo	25	1	echo buffer address
t.dcp	26	1	addr of message chain for cs
t.dlst	27	1	last buffer in messge chain for cs
t.ftse	30	1	first time slot entry in lsla table
t.sfcm	30	1	pointer to sfcm for hsla lines
t.bcnt	31	1	counting temporary
t.brkp	32	1	pointer to current break list
t.pos	33	1	current carriage position
t.char	34	1	pending line control char (lsla only)
t.ecch	34	1	address of current echo char (hsla only)
t.cntr	35	1	counter for control tables
t.flg3	36	1	third word of flags
t.dtp	37	1	pointer to delay table for this line
t.frmc	40	1	framing chars (2 chars)
t.dcpl	41	1	number of buffers in dcp chain
t.scll	42	1	screenline length left, for echo neg.
t.sncc	43	1	Echo negotiation sync ctr.
t.entp	44	1	Echo negotiation break table ptr
t.ifch	45	1	input flow control characters
t.ofch	46	1	output flow control characters
t.omct	47	1	output message count (for flow control)
t.itim	50	2	time of last call to inproc (2 words)
t.metr	52	1	address of metering area
t.abf0	53	1	absolute address of first permanent buffer
t.abf1	54	1	absolute address of first permanent buffer

Flags for tib.flg

Name	Address	Length	Explanation
tfosus	-400000	1	output suspended
tfecpx	200000	1	echoplex mode
tfisus	100000	1	input suspended
tfeotx	40000	1	eot expected from 2741
tfauto	20000	1	this is hsla autobaud line
tfwabt	10000	1	do write abort
tftbec	4000	1	do space echo on tab
tfcrec	2000	1	do cr echo on lf
tflfec	1000	1	do lf echo on cr
tfctrl	400	1	do kybd/printer addressing
tfquit	200	1	send nl on line break
tflisn	100	1	answer the phone
tfhang	40	1	hangup this line
tfprtr	20	1	printer addressed
tfkybd	10	1	keyboard addressed
tffdpx	4	1	full duplex mode
tfbel	2	1	send bell as next echoed character
tfwrit	1	1	output chain present
tfdlup	20	1	line is on dialup modem

Flags for tib.flg2

Name	Address	Length	Explanation
tfpfnl	-400000	1	line is in prefixnl mode
tfsftr	200000	1	terminal is shifting device
tffip	100000	1	frame in progress
tffrmi	40000	1	frame mode
tfmrcv	20000	1	'message' receive mode
tfcrcv	10000	1	'control' receive mode
tfblak	4000	1	asynchronous block acknowledgement
tfplit	2000	1	polite mode
tfrpon	1000	1	replay is occurring now
tfrply	400	1	replay mode
tfupsf	200	1	terminal is upshifted now
tfofc	100	1	output flow control mode
tfifc	40	1	input flow control mode
tfacu	20	1	do dial out
tfrabt	10	1	do read abort
tfercv	4	1	enter receive mode
tfdild	2	1	terminal is dialed up
tfxhld	1	1	hold transmit on

Flags for tib.flg3

Name	Address	Length	Explanation
tfmask	20000	1	channel has been masked for excessive interrupts
tfabf1	10000	1	t.abf1 available
tforp	4000	1	output resume pending (waiting until tro)
tfsoip	2000	1	suspension of output in progress
tfabf0	1000	1	t.afb0 available
tfoddp	400	1	odd parity
tf8out	200	1	don't strip output parity
tf8in	100	1	don't strip input parity
tfsked	40	1	input timeout routine scheduled
tfbral	20	1	break on all characters
tfecho	10	1	echoing has priority over output (lsla)
tfkpar	4	1	keep parity bits'
tfitim	2	1	timeout if input stops
tfbkpt	1	1	line is stopped at breakpoint

Flags for tib.stat

Name	Address	Length	Explanation
tsfcrq	20000	1	acu call request
tsfrts	10000	1	request to send
tsftre	4000	1	tally runout enable (hdlc)
tsfsxt	4000	1	supervisory transmit
tsfdtr	2000	1	data terminal ready
tsfbrk	1000	1	send line break
tsfxmt	400	1	transmit mode
tsfrcv	200	1	receive mode
tsftrm	100	1	send terminate status
tsfmrk	40	1	send marker status
tsfst	20	1	store status
tsfsrc	10	1	supervisory receive
tsfdsr	4	1	data set ready
tsfcts	2	1	clear to send
tsfcd	1	1	carrier detect

SFCM

Name	Address	Length	Explanation
sf.hcm	0	1	addr of hwcm
sf.nxa	1	1	addr of next available queue entry
sf.nxp	2	1	addr of next queue entry to process
sf.tly	3	1	tally of status queue
sf.tib	4	1	addr of tib for this line
sf.flg	5	1	flag word
sf.ib0	6	1	pointer to input buffer 1
sf.ib1	7	1	pointer to input buffer 2
sf.ob0	10	1	pointer to output buffer 1
sf.ob1	11	1	pointer to output buffer 2
sf.pcw	12	1	current pcw 2nd word
sf.cct	13	1	cct addr for this line, if non-zero
sf.rct	14	1	repeat count for status queue overflows
sf.hsl	15	1	address of hsla table entry for this line
sf.bsz	16	1	max buffer size
sf.fbs	17	1	buffer size to be used during frame input
sf.mms	20	1	maximum synchronous message size
sf.csz	20	1	current asynchronous buffer size
sf.rms	21	1	remaining unallocated message length
sf.nic	21	1	char. address of next asynchronous input char
sf.noc	22	1	char address of next asynchronous output char
sf.ssl	23	1	number of entries in software status queue
sf.cfg	24	2	2 words for config pcw
sf.sta	26	12	hardware status q, sicw here
sf.waq	42	40	wrap around queue, software status q

Flags for sf.flg

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

HWCM

Name	Address	Length	Explanation
h.ric0	0	2	primary receive icw
h.ric1	2	2	alternate receive icw
h.sic0	4	2	primary send icw
h.sic1	6	2	alternate send icw
h.baw	10	1	base address word
h.sfcm	11	1	software comm. region address
h.mask	12	2	mask register
h.aicw	14	2	active status icw
h.cnfg	16	2	configuration status

FNP Line Meters

Name	Address	Length	Explanation
m.dql	0	2	cumulative length of dia request queue
m.dqu	2	2	updates of m.dql
m.nst	4	2	cumulative no. of pending status
m.nsu	6	2	updates of m.nst
m.over	10	1	output overlaps
m.par	11	1	parity errors
m.ssqo	12	1	software status queue overflows
m.hsqo	13	1	hardware status queue overflows
m.inaf	14	1	input allocation failures
m.cql	15	1	current length of dia request queue
m.exh	16	2	exhaust status
m.xte	20	1	software xte status
m.leng	22	1	length of common meters (must be even)
m.prex	22	2	pre-exhaust status
m.ebof	24	2	echo buffer overflows
m.quit	26	1	bell-quits
m.asyl	30	1	total length of asynchronous meters
m.nim	22	2	number of input messages
m.iml	24	2	cumulative length of input
m.mini	26	1	minimum length of input message
m.maxi	27	1	maximum length of input message
m.nom	30	2	number of output messages
m.oml	32	2	cumulative length of output
m.mino	34	1	minimum length of output message
m.maxo	35	1	maximum length of output message
m.cnt1	36	2	first type of counter
m.cnt2	40	2	second type of counter
m.cnt3	42	2	third type of counter
m.cnt4	44	2	fourth type of counter
m.cnt5	46	2	fifth type of counter
m.cnt6	50	2	sixth type of counter
m.cnt7	52	2	seventh type of counter
m.cnt8	54	2	eighth type of counter
m.syn1	56	1	total length of synchronous meters

| display_cdt

| tty_lines

| system_comm_meters

| channel_comm_meters

| debug_fnp

    | bstat

    | sample_ic Developers Tool

    | fnp_idle

| interrupt_meters Ring 0 Global

| fnp_throughput not interesting

| meter_fnp_idle } Better Alternatives
| display_fnp_idle } to FNP idle from debug FNP

```
CDTE at 515|2460

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

```
                                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:
     cur_line_type:             0 (none)
     tty_id_code:               none
     process:                   77777|1
     next_channel:              0
     n_dialups:                 1649
     n_logins:                  1208  maynot have been reset, should reset_cdt_meters
                                      once week or month
     dialed_up_time:            210 hrs 58 mins 37 secs.
     dialup_time:               02/20/84  1623.0 hfh Mon
     recent_wakeup_count:       1
     recent_wakeup_time:        02/20/84  1730.2 hfh Mon
```

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

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

could this be modified for STATS for network, ↳ my channel com meters

system_comm_meters can do logical reset not of much use

Total metering time 0:10:03

THROUGHPUT

	before conversion	after conversion	ratio
Total characters input	5,041	4,705	0.93
Total characters output	153,567	123,256	0.80
Average length of input	5.1 characters		
Average length of output	73.3 characters		
Input characters preconverted	0 (0.0% of total)		

	read	write
Number of calls	1,436	2,495
Average time per call	2.13 msec.	5.09 msec.
Average chars. processed	3.5	61.6
Average chars. per msec.	1.7	12.1

CHANNEL INTERRUPTS not interesting

	input	output	other	total
software "interrupts"	3,243	3,155	113	6,511
average time (msec.)	2.45	1.02	2.36	1.76

TTY_BUF SPACE MANAGEMENT INTERESTING space in TTY_BUF then after other stuff.

Total size of buffer pool 19,368 words
Number of channels configured 191
Number of multiplexed channels 18

% of buffer pool in use	current	average
input	8.3	8.4
output	13.8	13.6
control structures	51.1	51.2
total	73.2	73.2

LCIE's WTCB's

should be less than 70% on normal system

Smallest amount nd free space ever 2,594 words (13% of buffer pool) should not be less than 10% of pool. reset at BOOT ↳ since BOOT

	allocate	free	total
Number of calls	6,755	5,894	12,649
Average time per call (msec.)	0.5	0.5	0.5
% of total CPU	0.6	0.5	1.0

Calls requiring loop on tty_buf lock 0 (0.00% of total)
average time spent looping on lock No data. msec. (0.00% of total CPU)
Number of allocation failures 0 (0.00% of attempts)

CHANNEL LOCK CONTENTION NOT interesting

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 for Video + EMACS

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

echoing of emacs & X.25 is the same. echoes of supervisor & xfers.

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, ie. TTY_BUF full, check meters on prev page.

```
Total metering time   78:15:27

a

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 believable, use FNP idle

                                  Input          Output
Characters transmitted        7,941,871      38,672,473
Characters per second                28             137

Abnormal DIA status                     0   IF
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 - indicates Bell quits } should be 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 TTY_buf
% of input transactions rejected      0.00
Total metering time   78:15:30 ---> goes with next Page
```

channel_comm_meters

a.h000

	before conversion	after conversion	ratio
Total characters input	3,471	3,274	0.94
Total characters output	874,287	1,243,897	1.42
Average length of input	5.0	4.7	
Average length of output	60.9	86.6	

	read	write	control	total
Number of calls	690	14,361	98	15,149
Average time per call (msec.)	3.2	8.6	36.6	8.6
Average chars. processed per call	5.0	60.9		
Number of software interrupts	10,891			
Average time per interrupt (msec.)	1.2			

	input	output
Effective speed (cps)	0.0	0.0

Output overlaps in FNP	2,875
Average length of DIA request queue	0.5
Exhaust status	0
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 non 0, not interesting usually good.
indication of line Business.
Avg Things waiting To go From mux over DIA
But not measured in time. >.5 is usually Busy.
BAD Thing. FNP could not keep up with input.
Possible errors.
Avg length of Software Status Queue.
Things Queued From Hardware Status Que, FNP can't keep up with time.
couldn't allocate Buffers, exhaust STAUS
Bell/Quits on ASync

channel_comm_meters

a.h006

	input	output
Messages transmitted	226,489	178,924
Minimum message length	2	2
Maximum message length	133	133
Average message length	5.19	25.25
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	

-INFO ignored by FNP. The value depends on X25 line

a.h008

Idle time 42.19%

	Input	Output
Blocks NAKed	2	68
Transmission timeouts	0	8
NAKs for DIA backlog	68	N/A
All transmission suspended	0	0
Blocks transmitted	4,812	7,042
Records transmitted	43,204	69
Average records per block	8.98	0.01

Duplicate input blocks	106
Output reprocessing	0
Blocks reprocessed	0

		input	output
Messages transmitted		4,812	7,042
Minimum message length		8	8
Maximum message length		397	398
Average message length	*B	344.85	8.09

Output overlaps in FNP	1,522
Average length of DIA request queue	0.5

Exhaust status	0
Software transfer timing error	2
Parity errors	0

Avg. number of pending status events	0.8
Software status queue overflows	0
Hardware status queue overflows	0
Input buffer allocation failures	0

nice for quick scan

channel_comm_meters -sum

Total metering time 78:15:36

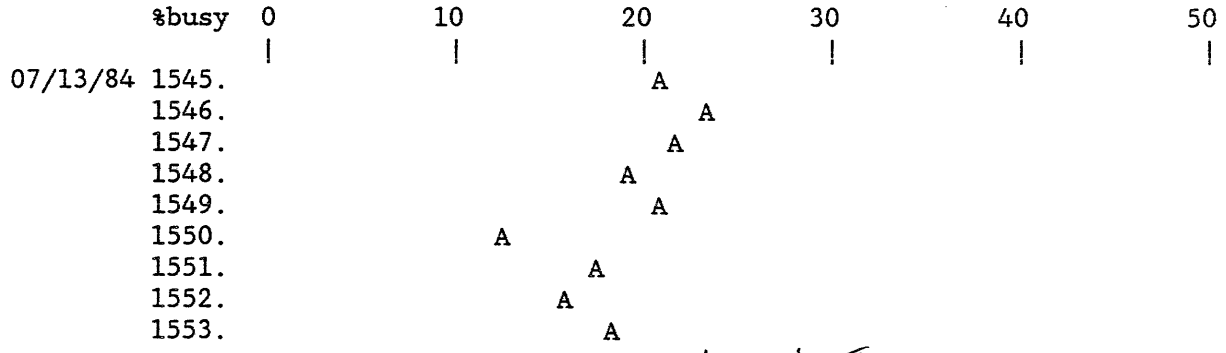
(characters per sec in out)

cps	cpsi	cpso	iotxXsbepQqa	err	ABE	name	user
120	No data.	No data.		0	a E	a.h000	Initializer
0	No data.	No data.		0	a	a.h001	
120	0.00	0.11		0	a E	a.h002	Initializer
120	0.00	0.02		0	a	a.h003	
0	No data.	No data.	i t X	0	s	a.h006	
0	No data.	No data.	io s	78	s	a.h008	
0	No data.	No data.		0	s	a.h009	
480	1.17	17.76		0	a E	a.h010	Weber
480	0.01	0.10		0	a E	a.h011	Initializer
480	0.09	3.70		0	a E	a.h012	Initializer
480	0.02	0.29		0	a E	a.h013	Initializer
480	0.04	2.24		0	a E	a.h014	Initializer
480	0.01	0.65		0	a E	a.h015	Initializer
0	No data.	No data.		0	s	a.h016	
0	No data.	No data.		0	s	a.h017	
480	0.39	0.03	xX e Qq	224	a E	a.h018	Initializer Sick line
480	0.18	2.06		0	a	a.h019	
480	0.66	11.58		0	a E	a.h020	Larmat
480	0.00	0.12		0	a	a.h021	
480	0.14	9.32		0	a E	a.h022	Initializer
480	0.05	1.34		0	a	a.h023	
0	No data.	No data.	iot p	130	s	a.h025	
480	0.04	0.84		0	a E	a.h026	Initializer
480	0.08	0.97		0	a	a.h027	
480	0.01	0.45		0	a E	a.h028	Initializer
480	0.21	7.07		0	a E	a.h029	Raich
480	0.01	1.56		0	a	a.h030	
480	0.03	1.53		0	a	a.h031	

(can be ignored.)

i: Invalid input messsage
o: Output message restransmitted
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

display_fnp_idle -histogram



first use meter fnp idle ~~first~~

display_fnp_idle -sum

```
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
```

5-10 minute intervals are TYPICAL

Busy Time.

You can Balance lines Between FNA's

for X.25. remove tracing module + or metering, or reduce lines,

~~Upgrade to~~ cache

or IF 6675 model upgrade, IT so it can use cache.

FNP

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

```
— db_fnp
—dump fnp.a.011084.1126
  No TIB available
—what
  Dump in >udd>SysAdmin>Homan>fnp.a.011084.1126, version 6.5d
—why
  illegal opcode fault at 46355 (util|6145)
  utilities: tried to free space already free

— map
  sked      4540  09/23/82
  intp      7176  11/18/82
  ctrl     13400  10/15/82
  dia      17060  01/11/83
  mclt     26342  12/01/82
  hsla     27250  07/07/83
  util     40210  12/01/82
  trac     46740  09/23/82
  bsc      47400  09/23/82
  hasp     54210  09/23/82
  x25tbs   56142  02/01/83
  meters   66112  09/23/82
  init     66304  09/29/83

—regs
  ic  046355 (util|6145)
  aq  137400 000040
  ir  520077 (zero,carry,interrupt inhibit)
  x1  077340 (init|11034)
  x2  004176
  x3  077000 (init|10474)
  er  777700
  et  776700

—bstat
  571 free
  12 small space

  LINE      INPUT BUFFR   DIA   OUTPUT   TOTAL
  a.h010          1         0       0        1
  a.h015          1         0       0        1
  a.h021          1         0       0        1
```

Terminal Information Block

dumps, list dumps in >dumps
dump_dir >dumps>old-dumps changes dir dumps looks in

INDex Reg #1 contains address of TIB. ADDress also goes with Page Table Entry

FNP memory address → utility program

Now it does it correctly

OFFSET in UTIL

Illegal opcode WAS intentional To Force FNP CRASH. ~~Called~~ IT executed AIP MACRO

SOFTware configured, address, date comp'ld.

← bind_FNP -list does some of this also, more detail

Values of FNP registers

instruction counter

indicator register

NORMAL here

buffer status looks AT TIB's in All Lines

FNP old default Buffer size (32 word BUFFers) 571 x 32 words = Free Space

(50-100) is risky.

low B-FFERS CAUSES Bell/Quit + ignores input. NOT ALWAYS

ASYNC → UNUSUAL AMOUNT INPUT BUFFERS OUT, 2 TYPICAL

OUTPUT BUFFERS NOT CRITICAL

```
a.h112      0          0          3          3
a.h114      4          0          0          4
a.h115      1          0          0          1

TOTAL       8          0          3          11
```

50 MOST RECENT OCTAL ENTRIES, 40 decimal

```
— print_trace -50
662566 op block at 53024 (bsc|3424), type = 060 (clrlcf)
662567 op block at 53027 (bsc|3427), type = 037 (retsub)
662567 op block at 54376 (hasp|166), type = 061 (tstlcf)
662570 op block at 54427 (hasp|217), type = 033 (dumpin)
662570 op block at 54430 (hasp|220), type = 056 (tstlcl)
662571 op block at 54370 (hasp|160), type = 072 (setimv)
662572 op block at 54372 (hasp|162), type = 005 (dcwlst)
662572 hsla dcw processor, tib 122562, list 54373 (hasp|163), len 01
662575 hsla pcw, tib 122562, pcw 211000 000460
662576 op block at 54374 (hasp|164), type = 036 (calsub)
662577 hsla interrupt, 3wjt = 362004 (ch=7 dv=2 subch=10 mod=4)
662600 op block at 52476 (bsc|3076), type = 060 (clrlcf)
662600 op block at 52501 (bsc|3101), type = 060 (clrlcf)
662601 op block at 52504 (bsc|3104), type = 061 (tstlcf)
662601 op block at 52510 (bsc|3110), type = 061 (tstlcf)
662602 op block at 52514 (bsc|3114), type = 003 (wait)
662602 hsla status, tib 122562, status 400000 404460
662721 hsla interrupt, 3wjt = 363004 (ch=7 dv=2 subch=14 mod=4)
662722 hsla status, tib 123512, status 420200 702060
662723 istat: tib at 123512, t.cur = 16367 (ctrl|2767), status 000407
662724 op block at 16413 (ctrl|3013), type = 004 (status)
662724 op block at 16036 (ctrl|2436), type = 036 (calsub)
662725 op block at 17034 (ctrl|3434), type = 005 (dcwlst)
662725 hsla dcw processor, tib 123512, list 17035 (ctrl|3435), len 01
662726 hsla pcw, tib 123512, pcw 211400 000060
662727 op block at 17036 (ctrl|3436), type = 006 (setime)
662727 hsla interrupt, 3wjt = 363004 (ch=7 dv=2 subch=14 mod=4)
662731 op block at 17040 (ctrl|3440), type = 003 (wait)
662732 hsla status, tib 123512, status 420000 700060
662732 istat: tib at 123512, t.cur = 17040 (ctrl|3440), status 000047
662733 op block at 17044 (ctrl|3444), type = 004 (status)
662733 op block at 17050 (ctrl|3450), type = 037 (retsub)
662734 op block at 16040 (ctrl|2440), type = 052 (setcct)
662735 op block at 16042 (ctrl|2442), type = 011 (clrflg)
662735 op block at 16045 (ctrl|2445), type = 017 (sendin)
662736 op block at 16046 (ctrl|2446), type = 076 (setfld)
662736 op block at 16051 (ctrl|2451), type = 014 (signal)
662740 new entry in dia i/o queue at 067176: opcode 113 (brkcon), line 1114
662740 op block at 16053 (ctrl|2453), type = 012 (tstflg)
662741 op block at 16064 (ctrl|2464), type = 013 (dmpout)

— explain .crldt;d .crldt
".crldt" = date and time of binding
00640  000000 112155 200407 431126  09/29/83  1746.7 hfe Thu
explain .crbdt;d .crbdt
```

clock value NOT IMPORT.

Can be used TO identify Line

Send dcw command TO HSLA. look in HSLA_MOR

C.H108

hsla #1 (2nd hsla chan)

SUBTRACT 1 from dv

3wjte

a.h112

octal

SUB CHAN # = a.h112 decimal

hsla bit

hsla #

display symbolic or can be absolute 640 for 4 words

```
".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  067440 000040 000000 000000 000000 007000 007000 000000 7 . .....
67310  000000 000000 000000 000000 000000 000000 000000 000000 ..............
67330  =

67440  067540 000040 000000 000000 000000 000000 000000 000000 7'. .....
67450  000000 000000 000000 000000 000000 000000 000000 000000 ..............
67470  =

67540  000000 007240 000000 000000 000000 000000 000000 000000 ... .....
67550  000000 000000 000000 000000 000000 000000 000000 000000 ..............
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
```

Asked for display com region + BUFFERS

577 (to decimal) = free BUFFERS from BSTAT

BUFFER chain

contains address of next BUFFER

so CRNXA contains 67300

next address — number of words

```
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;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 000020 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  sked|1702
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
```

con regions

dump 40

small Buffers not chained so can't use chain request

```
135440  136740 000040 ^'.

136740  137240 000040 _ .

137240  137340 000040 _'.

137340  137400 000040 _..

137400  137540 000040 _'.

137540  137700 000100 _@.@

137700  140000 000040 '..

140000  140100 000040 '@.
          :
          :
171400  172000 000400 z...

172000  172400 000400 z...

172400  173000 000400 {...

173000  173400 000400 {...

173400  000000 000400 ....
explain .crbpe;d .crbpe
".crbpe" = buffer paging window table entry
00710  004176
explain .crcpr;d .crcpr
".crcpr" = copyright notice
00760  103117 120122 056040 050103 051040 110056 111056 123056 COPR. (C) H.I.S.
00770  040111 116103 056040 061071 070061 040040                INC. 1981
q

dump fnp.a.011084.1126
No TIB available
line a.hl14
d tib
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
explain t.stat;flags t.stat
"t.stat" = holds current line status
124512  012207 tsfrts tsfdtr tsfrcv tsfdsr tsfcts tsfcd
explain t.flg;flags t.flg
"t.flg" = flag word
124513  207304 tfecpx tftbec tfcrec tflfec tfquit tflisn tffdpx
explain t.flg2;flags t.flg2
"t.flg2" = second word of flags
124514  400002 tfpfnl tfdild
```

```
explain t.cur;d t.cur,* opBlock Prints out opBlock
"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 075077 }=7ou=Uw?u{um?=?
123760  075077 077057 077077 067077 077077 153153 173173 157167 =??/???77?kk{{ow
123770  077155 155153 173077 173173 133173 057057 155153 173177 ?mmk{?{{[{//mk{.

134440  142040 000074 157177 177177 177172 177177 165135 177177 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 176177 177177 133173 137137 177177 077127 ?;..~...[{__..?W

142040  114340 000074 135177 073167 177177 177165 177165 176176 L'.<].;w...u.u~~
142050  073067 133057 157173 157155 135133 133133 135153 157065 ;7[/o{om][[[]ko5
142060  155177 177177 147073 177175 177157 177177 157133 157157 m...g;.}.o..o[oo
142070  153057 073177 155167 177177 173057 077077 153157 057155 k/;.mw..{/??ko/m

114340  000000 000021 133133 153177 173065 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 ................
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)
124523  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 175177 157177 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 messge chain for cs
124541  000000
explain t.ftse;d t.ftse
"t.ftse" = first time slot entry in lsla table
124542  077400
d t.sfcm
124542  077400
explain t.bcnt;d t.bcnt
"t.bcnt" = 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.dcpl;d t.dcpl
"t.dcpl" = 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.abf1;d t.abf1
"t.abf1" = absolute address of first permanent buffer
124566  124660
explain sfcm;d sfcm
Symbol "sfcm" has no explanation.
124400  002340 077472 077472 000012 077512 000204 105740 103340
124410  000000 000000 000460 067200 000000 004374 000040 000040
124420  000040 377347 000000 000012 571626 000004 000000 700660
124430  000000 700460 000000 700460 000000 000000 000000 000000
124440  000000 000000 600000 700460 000000 000000 032000 700660
124450  000000 000000 004000 700460 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
"sf.flg" = flag word
124405  000204 sffdct sffcoi
explain sf.ib0;d sf.ib0,* 40 -ch
"sf.ib0" = pointer to input buffer 1
105740  040377 373265 365277 325335 355257 335277 373377 377377  .{5u?U]m/]?{...
105750  377357 273173 377333 277376 377333 255255 277377 273133 .o;{.[?~.[--?.;[
105760  357277 273376 376377 257377 377153 377277 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 133357 157353 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  000040
explain sf.rms;d sf.rms
"sf.rms" = 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 600000 700460 000000 000000
124462  012000 700660 000000 000000 024000 700460 000000 000000
124472  004000 700460 000000 000000 600000 700460 000000 000000
124502  012000 700660 000000 000000 024000 700460 000000 000000
explain sf.sta;d sf.sta
"sf.sta" = hardware status q, sicw here
124426  000000 700660 000000 700460 000000 700460 000000 000000
124436  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.ric1;d h.ric1
"h.ric1" = alternate receive icw
02342  103340 460075
explain h.sic0;d h.sic0
"h.sic0" = primary send icw
02344  131754 470000
explain h.sic1;d h.sic1
"h.sic1" = 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.aicw;d h.aicw
"h.aicw" = active status icw
02354  124426 420005
explain h.cnfg;d h.cnfg
"h.cnfg" = configuration status
```

```
02356  443026 010004
explain m.dql;d m.dql
"m.dql" = cumulative length of dia request queue
124570  000000 000035
explain m.dqu;d m.dqu
"m.dqu" = updates of m.dql
124572  000000 000072
explain m.nst;d m.nst
"m.nst" = cumulative no. of pending status
124574  000000 006023
explain m.nsu;d m.nsu
"m.nsu" = updates of m.nst
124576  000000 006220
explain m.over;d m.over
"m.over" = output overlaps
124600  000000
explain m.par;d m.par
"m.par" = parity errors
124601  000000
explain m.ssqo;d m.ssqo
"m.ssqo" = software status queue overflows
124602  000000
explain m.hsqo;d m.hsqo
"m.hsqo" = hardware status queue overflows
124603  000000
explain m.inaf;d m.inaf
"m.inaf" = input allocation failures
124604  000000
explain m.cql;d m.cql
"m.cql" = current length of dia request queue
124605  000000
explain m.exh;d m.exh
"m.exh" = exhaust status
124606  000000 000000
explain m.xte;d m.xte
"m.xte" = software xte status
124610  000000
explain m.prex;d m.prex
"m.prex" = pre-exhaust status
124612  000000 000000
explain m.ebof;d m.ebof
"m.ebof" = echo buffer overflows
124614  000000 000000
explain m.quit;d m.quit
"m.quit" = bell-quits
124616  000000
q
```

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

```
debug_fnp {initial_command_line}
```

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, seperated 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 betweens 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 direcotry 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 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 "(", ")", "+", "-", "*", and "/" with their normal meanings and precedence. The symbol "|" is synnonomous 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 mnemmonic, the name of any FNP object module, or a machine instruction (specified by surrounding the instruction by apostrophes). In additon, user symbols can be defined. Examples of expressions:

```
hsla|500
t.icp,*
*+30
tib|14,*+10
goto
'lda 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 sepcified. 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 ommitted, character mode is assumed. If -breif is not specified, the entrie 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"
patch 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 equivilent 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 similiar 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 Analysys 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 infromation 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 occuring 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 traceing 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 excercised 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 equivilent to:

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

   followed by no status blocks. This means that timers can runout 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
```

will list all FNP breakpoints and the channels stopped at each.

Peformance 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:

```
idle_time {-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:

```
sample_time {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, whih 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:

```
ic_sample start
```

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

```
ic_sample stop
```

stops ic_sampling.

```
ic_sample reset
```

zeroes all the sampling buckets.

The following options are used to display the information collected:

```
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 referrenced buckets. This option is useful in deleting "noise" from the histogram.

Other commands

To select a specfic 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 explaination of any FNP symbol (usally the output of a flags or convert_address command):

```
explain sym1 {sym2} ... {symn}
```

where sym$_i$ 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 processer.

To exit from debug_fnp,

```
quit
q
```

Multics Problems/Dumps

- tty_analyze
- tty_dump
- trace_mcs

```
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 407777000311, event = 007777000001 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: uproc_attached
shift state = 00 (none) l1 = 0, p1 = 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:
000000000061 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:  ...........
```

Logical Channel Table Entry 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,lfecho,hndlquit,echoplex,polite,scroll,can_type=replace
flags: uproc_attached
shift state = 00 (none) ll = 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
```

tty_analyze

USES MULTICS System Dump + USES IT'S TTY_BUF instead of FNP Dump.

NOT ALWAYS Trustworthy. For use with TL systems crashes involving MCS

```
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 lfech
 LC 16: state=p41(FLOW CONTROL READY),max_packet_size=128,window used=0/2
          iti baud=4800
           address=31063050005411
         .
         .
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:
000000000061 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)
```

RING0 Tracing setup in TTY_BUF

→ Table size — words or entries

```
trace_mcs ts 100
r 08:22 0.199 94

trace_mcs modes all
New MCS trace modes: on,^default,read,write,data,control,modes,interrupt,
init_mpx,start_mpx,stop_mpx,space_man
r 08:23 0.199 89

trace_mcs channel ** -on
r 08:24 0.435 91

trace_mcs print          Prints out events
08:24:55.331876  a.h214: int: proc accept_input 024130024130000007604103
08:24:55.334231  a.h214:  16 words at 024130; 7 chars; flags:
08:24:55.336228  a.h214:   0: 000000000007 003144020000 .....d..
08:24:55.337438  a.h214:   8: 010161215000 000000000000 .q......
08:24:55.339879  a.h214.001: int: proc accept_input 024130024130000002404103
08:24:55.342078  a.h214.001:  16 words at 024130; 2 chars; flags: break
08:24:55.345253  a.h214.001:   8: 010161215000 000000000000 .q......
08:24:55.348781  a.h214: write: 216|41214:  6 bytes
08:24:55.349907  a.h214:  16 words at 041214; 6 chars; flags: break
08:24:55.353048  a.h214:   8: 240412000000 000000000000 ........
08:24:55.390663  a.h214.001: control: reset_more 77777|1
08:24:55.398741  a.h214.001: write: 216|24130:  19 bytes
08:24:55.400093  a.h214.001:  16 words at 024130; 19 chars; flags: break
08:24:55.403383  a.h214.001:   8: 072062064040 060056063071 :24 0.39
08:24:55.404609  a.h214.001:  16: 060040070015 012015012000 0 8.....
08:24:55.408170  a.h214.001: int: queued send_output 000000000000000000000000
08:24:55.411969  a.h214.001: int: proc queued send_output 000000000000000000000000
08:24:55.423650  a.h214: int: proc send_output 000000000000000000000000
08:24:55.426293  a.h214: write: 216|24130:  24 bytes
08:24:55.427510  a.h214:  16 words at 024130; 24 chars; flags: break
08:24:55.430770  a.h214:   8: 242162240060 270072262264 .r.0.:..
08:24:55.431998  a.h214:  16: 240060056063 071060240270 .0.390..
08:24:55.433199  a.h214:  24: 215012215412 000000000000 ........
08:24:55.629519  a.h214: int: proc accept_input 024130024130000005604103
08:24:55.631780  a.h214:  16 words at 024130; 5 chars; flags:
08:24:55.633764  a.h214:   0: 000000000005 003206020000 ........
08:24:55.634967  a.h214:   8: 041000000000 000000000000 !.......
08:24:55.645811  a.h214: int: proc send_output 000000000000000000000000
08:24:55.786612  a.h214: int: proc accept_input 024130024130000005604103
08:24:55.788824  a.h214:  16 words at 024130; 5 chars; flags:
08:24:55.790779  a.h214:   0: 000000000005 003250020000 ........
08:24:55.791978  a.h214:   8: 101000000000 000000000000 A.......
08:24:55.969268  a.h027: int: proc accept_input 024130024130000011404103
08:24:55.971547  a.h027:  16 words at 024130; 9 chars; flags:
08:24:55.973504  a.h027:   0: 000000000011 156145167137 ....new_
08:24:55.974704  a.h027:   8: 160162157143 012000000000 proc....
08:24:55.987471  a.h027: control: set_required_access_class 366|5160
08:24:56.002503  a.h027: write: 216|24130:  60 bytes etc.
08:24:56.003797  a.h027:  16 words at 024130; 60 chars; flags:
08:24:56.007063  a.h027:   8: 040144151163 143157156156  disconn
08:24:56.008293  a.h027:  16: 145143164145 144040160162 ected pr
08:24:56.009492  a.h027:  24: 157143145163 163040167151 ocess wi
```

```
08:24:56.010691  a.h027:  32: 154154040142 145040143157 ll be co
08:24:56.011892  a.h027:  40: 156156145143 164145144040 nnected
08:24:56.013123  a.h027:  48: 164157040164 150151163040 to this
08:24:56.014348  a.h027:  56: 164145162155 151156141154 terminal
08:24:56.015472  a.h027:  16 words at 041214; 18 chars; flags: break
08:24:56.018778  a.h027:   8: 145162040156 145167137160 er new_p
08:24:56.020008  a.h027:  16: 162157143056 015012000000 roc.....
08:24:56.059137  a.h027: int: proc send_output 000000000000000000000000
08:24:56.169153  a.h000: write: 216|24130:  60 bytes etc.
08:24:56.170605  a.h000:  16 words at 024130; 60 chars; flags:
08:24:56.172639  a.h000:   0: 041214000074 040060070062 !..< 082
08:24:56.173865  a.h000:   8: 064040040141 1630040040 4  as
08:24:56.175127  a.h000:  16: 103117116116 105103124040 CONNECT
08:24:56.176354  a.h000:  24: 126111120067 062060065040 VIP7205
08:24:56.177579  a.h000:  32: 040156157156 145040141056  none a.
08:24:56.178779  a.h000:  40: 150060062067 040115157162 h027 Mor
08:24:56.179977  a.h000:  48: 145154154157 056101111130 ello.AIX
08:24:56.181180  a.h000:  56: 137101124015 012000000000 _AT.....
08:24:56.182305  a.h000:  16 words at 041214; 21 chars; flags: break
08:24:56.238164  a.h000: int: proc send_output 000000000000000000000000
08:24:56.823444  a.h027: check_modes: 230|1460
08:24:56.826460  a.h027: set_mor?s: 230|1460
08:24:58.363862  a.h006: int: proc accept_input 024130024130000025604103
08:24:58.366380  a.h006:  16 words at 024130; 21 chars; flags:
08:24:58.368450  a.h006:   0: 000000000025 003106020000 .....F..
08:24:58.369678  a.h006:   8: 346164162141 143145137155 .trace_m
08:24:58.370917  a.h006:  16: 143163040160 162151156164 cs print
08:24:58.372126  a.h006:  24: 015000000000 000000000000 ........
08:24:58.374735  a.h006.001: int: proc accept_input 024130024130000020404103
08:24:58.376945  a.h006.001:  16 words at 024130; 16 chars; flags: break
08:24:58.380116  a.h006.001:   8: 145137155143 163040160162 e_mcs pr
08:24:58.381505  a.h006.001:  16: 151156164012 162151156164 int.rint
08:24:58.382702  a.h006.001:  24: 015000000000 000000000000 ........
08:24:58.386288  a.h006: write: 216|41214:  6 bytes
08:24:58.387422  a.h006:  16 words at 041214; 6 chars; flags: break
08:24:58.390562  a.h006:   8: 216412000000 000000000000 ........
08:24:58.462541  a.h006: int: proc send_output 000000000000000000000000
08:24:58.533918  a.h027: control: copy_meters 77777|1
r 08:25 0.929 89
```

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 reset
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

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 argumentds (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" fag, 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

```
/*  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 Beattie 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 FNPs 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 iom/channel fb's.
   Modified 1984-07-26 BIM for paged iom.
*/

/* 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 t&d 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,                         /* watch events on console */
       2 debug_stop bit (1) aligned,                    /* crash on errors */
       2 pad (2) bit (36) aligned,                      /* to get alignment for dump reading */
       2 per_datanet (max_no_355s) aligned like fnp_info;   /* data per datanet */
```

```
dcl  1 fnp_info aligned based (fnpp),                    /* structure for each FNP */
       2 mbx_pt pointer,                                  /* pointer to mailbox NULL if not in config */
       2 pcb_array_ptr pointer,                           /* pointer to array of physical channel blocks */
       2 dcw_list_array_ptr pointer,                      /* pointer to array of space reserved for output DCW lists */
       2 no_of_channels fixed bin,                        /* number of channels on this FNP */
       2 fnp_id,
         3 fnp_tag char (1) unaligned,                    /* letter identifying FNP */
         3 fnp_number fixed bin (9) unsigned unaligned,   /* sequence number of FNP */
         3 padc bit (18) unaligned,
       2 io_chanid char (8) aligned,
       2 io_manager_chx fixed bin (35),                   /* devx for DIA on iom */
       2 lsla_idx (0:5) fixed bin aligned,                /* index into PCB array for lsla lines */
       2 hsla_idx (0:2) fixed bin aligned,                /* index into PCB array for hsla lines */
       2 count fixed bin,                                 /* number of items in delay queue */
       2 cur_ptr fixed bin,                               /* offset in tty_buf of next delay queue element */
       2 last_ptr fixed bin,                              /* offset in tty_buf of last delay queue element */
       2 bleft_355 fixed bin,                             /* number of free buffers in this 355 */
       2 flags,
         3 work_reqd bit (1) unaligned,                   /* mailbox messages queued up */
         3 bootloading bit (1) unaligned,                 /* currently being bootloaded */
         3 running bit (1) unaligned,                     /* this FNP is running */
         3 wired bit (1) unaligned,                       /* bootload buffer is wired */
         3 dump_patch_in_progress bit (1) unaligned,      /* a dump or patch order is in progress */
         3 level_3_pending bit (1) unaligned,             /* level 3 interrupt pending */
         3 level_7_pending bit (1) unaligned,             /* level 7 interrupt pending */
         3 dump_patch_disabled bit (1) unaligned,         /* dump & patch orders disabled because of timeout */
         3 t_and_d_in_progress bit (1) unaligned,         /* T & D using FNP */
         3 t_and_d_lev_3_occurred bit (1) unaligned,      /* A level 3 occurred */
         3 t_and_d_lev_7_occurred bit (1) unaligned,
         3 t_and_d_notify_requested bit (1) unaligned,
         3 t_and_d_assigned bit (1) unaligned,            /* AS has given fnp to process */
         3 get_meters_waiting bit (1) unaligned,          /* waiting for meter copy to complete */
         3 padb bit (22) unaligned,
       2 lcte_ptr ptr,                                    /* pointer to this FNP's LCT entry */
       2 astep ptr,                                       /* pointer to aste of wired bootload buffer */
       2 boot_ev_chan fixed bin (71),                     /* event channel over which to signal bootload completion */
       2 boot_process_id bit (36),                        /* process that initiated bootload */
       2 version char (4),                                /* version id of core image */
       2 fnp_mem_size fixed bin (18) unsigned,            /* memory size of this FNP in 18-bit words */
       2 queue_lock bit (36) aligned,                     /* lock for interrupt queue */
       2 dump_patch_lock bit (36),                        /* lock for fnp_dump or _patch operation */
       2 q_entries_made fixed bin (35),                   /* count of delay queue entries made */
       2 input_reject_count fixed bin,                    /* number of times input rejected */
       2 processed_from_q fixed bin (35),                 /* number of interrupts processed from queue */
       2 fnp_channel_locked fixed bin (35),               /* number of times dn355 found per-FNP lock locked */
       2 input_data_transactions fixed bin (35),          /* number of mailbox transactions for input */
       2 output_data_transactions fixed bin (35),         /* number of mailbox transactions for output */
       2 input_control_transactions fixed bin (35),       /* number of mailbox transactions for inbound control info */
       2 output_control_transactions fixed bin (35),      /* number of mailbox transactions for outbound control info */
       2 cumulative_mbx_in_use fixed bin (35),            /* cumulative count of number of outbound mailboxes in use */
       2 max_mbx_in_use fixed bin,                        /* maximum number of mailboxes in use at any given time */
       2 mbx_in_use_updated fixed bin (35),               /* number of increments to cumulative_mbx_in_use */
       2 mbx_unavailable fixed bin (35),                  /* number of times had to queue mailbox transaction because none available */
       2 free_size fixed bin (35),                        /* cumulative amount of bleft_355 */
       2 free_count fixed bin,                            /* number of adds to above */
```

```
      2 fnp_space_restricted_output fixed bin (35),       /* number of times available FNP space restricted amount of output sent */
      2 tandd_pcbx fixed bin,                              /* index of PCB for COLTS channel */
      2 n_pages_wired fixed bin,                           /* pages wired for loading */
      2 config_flags aligned,
        3 available bit (1) unaligned,                     /* reconfig says "yes" */
        3 io_manager_assigned bit (1) unaligned,           /* We have channel assigned to us */
        3 pad bit (34) aligned,
      2 ptx fixed bin,                                     /* page table index, used only at bootload */
      2 ptp pointer unaligned;                             /* page table for this FNP */

/**** The following named constants are used to lay out 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

      Page   I/O address   Memory address     Comments
      ----      ------     --------------     --------
      0            0       xxxxxx           invalid PTW
      1         2000         2000           write-enabled (mailbox)
      2         4000         4000           write-enabled (mailbox)
      3         6000         6000           write-enabled (mailbox)
      4        10000       as needed        bootload image segment page 0
      ....       ....         ....           ....
      35      110000       as needed        bootload image segment page 31
      36      112000       xxxxxx           invalid PTW
      ...       ....         ....           ....
      63      160000         ....           invalid PTW
      64      200000       as needed        tty_buf page 0
      ...       ....         ....           ....
      127     260000       as needed        tty_buf page 63
      255     ......        .....           tty_buf page 191
*/

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

declare  FIRST_BOOTLOAD_PAGEX fixed bin init (4) int static options (constant);
declare  FIRST_TTY_BUF_PAGEX fixed bin init (64) int static options (constant);

/* End include file dn355_data.incl.pl1 */
```

```
/* BEGIN INCLUDE FILE ... lct.incl.pl1 */

/* 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;                                          /* ptr to logical channel table */
dcl  lctep ptr;                                         /* ptr to logical channel table entry */
dcl  lct_size fixed bin;                                /* size of lcte_array when allocated */

dcl 1 lct aligned based (lctp),                         /* logical channel table */
    2 max_no_lctes fixed bin,                           /* maximum number of lct entries */
    2 cur_no_lctes fixed bin,                           /* current number of lct entries used */
    2 lcnt_ptr ptr,                                     /* ptr to logical channel name table */
    2 queue_lock bit (36),                              /* lock used to serialize queueing operations */
    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),                       /* logical channel table entry */
    2 lock bit (36),                                    /* channel lock */
    2 data_base_ptr ptr unal,                           /* ptr to channel data base */
    2 channel_type fixed bin (8) unal,                  /* identifies channel manager program */
    2 flags unal,
      3 entry_in_use bit (1) unal,                      /* ON if this entry in use */
      3 initialized bit (1) unal,                       /* ON if this channel initialized */
      3 notify_reqd bit (1) unal,                       /* ON if must notify when unlocking this channel */
      3 locked_for_interrupt bit (1) unal,              /* ON if lock set by interrupt handler */
      3 space_needed bit (1) unal,                      /* ON if this channel needs buffer space */
      3 special_lock bit (1) unal,                      /* ON if lock is managed by multiplexer */
      3 trace_force bit (1) unal,                       /* ON to trace based on next bit only */
                                                        /* OFF to XOR next bit with tty_buf.default_tracing */
      3 trace bit (1) unal,                             /* ON to trace this channel */
      3 unused bit (1) unal,
    2 physical_channel_devx fixed bin (17) unal,        /* devx of physical chan from which logical chan is derived */
    2 major_channel_info,
      3 major_channel_devx fixed bin unal,              /* major channel device index */
      3 subchannel fixed bin (17) unal,                 /* subchannel id (or data ptr) wrt major channel */
    2 queue_entries,
      3 queue_head bit (18) unal,                       /* ptr to first queue entry for this channel */
      3 queue_tail bit (18) unal,                       /* ptr to last queue entry for this channel */
    2 word_counts,
      3 input_words fixed bin (17) unal,                /* number of input words charged to this channel */
      3 output_words fixed bin (17) unal,               /* number of output words charged to this channel */

    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 lcte.meters.in,
      3 control like lcte.meters.in,
    2 saved_meters_ptr ptr,                     /* pointer to unwired copy of meters saved at last dialup */

    2 timer_offset bit (18) aligned,            /* Head of list of timers for this channel */

    2 pad (3) fixed bin (35);


dcl  lcntp ptr;                                 /* ptr to logical channel name table */

dcl 1 lcnt aligned based (lcntp),               /* logical channel name table */
    2 names (lct.max_no_lctes) char (32) unal;  /* channel names */

dcl 1 saved_meters aligned based like lcte.meters;  /* meters saved at dialup, allocated in tty_area */


/* END INCLUDE FILE ... lct.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),                          /* physical channel block declaration */
    2 channel_desc unaligned,
      3 devx fixed bin (17),                             /* index of LCT entry */
      3 subchan fixed bin (7) unaligned,                 /* logical subchannel/lsla slot # correspondence */
      3 line_number unal,                                /* regular line number */
        4 is_hsla bit (1) unaligned,                     /* on if hsla, off if lsla */
        4 la_no bit (3) unaligned,                       /* line adapter (high or low speed) number */
        4 slot_no bit (6) unaligned,                     /* physical slot or subchannel number */
    2 write_first fixed bin (17) unaligned,              /* offset of first buffer in output chain */
    2 write_last fixed bin (17) unaligned,               /* offset of last buffer in output chain */
    2 baud_rate fixed bin (17) unaligned,                /* baud rate of channel */
    2 line_type fixed bin (17) unaligned,                /* line type */
    2 max_buf_size fixed bin (17) unaligned,             /* largest buffer to be allocated for output */
    2 write_cnt fixed bin (17) unaligned,                /* number of characters in write chain */
    2 flags unaligned,
      3 listen bit (1),                                  /* channel is ready for dialup */
      3 dialed bit (1),                                  /* channel is dialed up or connected */
      3 send_output bit (1),                             /* channel is ready for output */
      3 high_speed bit (1),                              /* needs large send_out threshold */
      3 sync_line bit (1),                               /* synchronous channel */
      3 end_frame bit (1),                               /* channel is waiting for formfeed */
      3 hndlquit bit (1),                                /* channel in hndlquit mode */
      3 breakall_enabled bit (1),                        /* breakall mode allowed for this channel */
      3 output_mbx_pending bit (1),                      /* A wtx mbx has been sent, but not relinquished */
      3 copied_meters_ready bit (1),                     /* copy_meters operation has completed */
      3 get_meters_waiting bit (1),                      /* waiting for get_meters operation to complete */
      3 tandd_attached bit (1),                          /* this channel is in use by T & D */
      3 padb bit (24),
    2 read_first fixed bin (18) unsigned unaligned,      /* head of read chain (while reading from FNP) */
    2 read_last fixed bin (18) unsigned unaligned,       /* tail of read chain (likewise) */
    2 saved_meters_ptr pointer unaligned,                /* pointer to (unwired) copy of meters at last dialup */
    2 copied_meters_offset fixed bin (18) unsigned;      /* 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 no_outp, oddp, & eight_bit modes
   Modified 10/08/80 by Robert Coren to add meters for tty_read & tty_write
   Modified: 10 November 1980 by G. Palter 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 */

    2 terminal_type char (32) unaligned,           /* terminal type name */
    2 tables,
      3 input_mvtrp bit (18) unaligned,            /* rel pointer to current input mvt table */
      3 output_mvtrp bit (18) unaligned,           /* rel pointer to current output mvt table */
      3 input_tctrp bit (18) unaligned,            /* rel pointer to current input tct table */
      3 output_tctrp bit (18) unaligned,           /* rel pointer to current output tct table */
      3 specialrp bit (18) unaligned,              /* rel pointer to current special chars table */
      3 delayrp bit (18) unaligned,                /* rel pointer to current delay table */
    2 default_tables,
      3 df_input_mvtrp bit (18) unaligned,         /* rel pointer to default input mvt table */
      3 df_output_mvtrp bit (18) unaligned,        /* rel pointer to default output mvt table */
      3 df_input_tctrp bit (18) unaligned,         /* rel pointer to default input tct table */
      3 df_output_tctrp bit (18) unaligned,        /* rel pointer to default output tct table */
      3 df_specialrp bit (18) unaligned,           /* rel pointer to default special chars table */
      3 df_delayrp bit (18) unaligned,             /* rel pointer to default delay table */
    2 special_input_chars unaligned,
      3 erase char (1) unaligned,                  /* erase character */
      3 kill char (1) unaligned,                   /* kill character */
    2 old_type fixed bin (17) unaligned,           /* old terminal type number */

    2 modes unaligned,                             /* modes set by order call */
      3 edited bit (1) unaligned,                  /* edited output mode */
      3 tabm bit (1) unaligned,                    /* insert output tabs mode */
      3 canm bit (1) unaligned,                    /* do canonical form conversion */

      3 escm bit (1) unaligned,                    /* do input escape conversions */
      3 erklm bit (1) unaligned,                   /* do erase kill processing */
      3 rawim bit (1) unaligned,                   /* don't convert input */
```

```
    3 rawom bit (1) unaligned,                      /* don't convert output */
    3 redm bit (1) unaligned,                       /* has red-shift function */
    3 vertsp bit (1) unaligned,                     /* send real ff's and vt's if on, else escape them */

    3 echo_cr bit (1) unaligned,                    /* echo carriage returns */
    3 echo_lf bit (1) unaligned,                    /* echo line feeds */
    3 echo_tab bit (1) unaligned,                   /* echo tabs */

    3 hndlquit bit (1) unaligned,                   /* cr's on quit */
    3 full_duplex bit (1) unaligned,                /* xmit and receive simultaneously */
    3 echoplex bit (1) unaligned,                   /* echo input characters on terminal */

    3 upper_case bit (1) unaligned,                 /* map lower-case output into upper-case */
    3 replay bit (1) unaligned,                     /* replay interrupted input */
    3 polite bit (1) unaligned,                     /* output must start at left margin */

    3 control bit (1) unaligned,                    /* accept control characters */
    3 blk_xfer bit (1) unaligned,                   /* block transfer or "frame" mode */
    3 breakall bit (1) unaligned,                   /* break on all characters */

    3 scroll bit (1) unaligned,                     /* scroll mode for crt terminals */
    3 prefixnl bit (1) unaligned,                   /* prefix output iwth nl when input interrupted */
    3 wake_tbl bit (1) unaligned,                   /* input wakeups determined by wakeup table */

    3 iflow bit (1) unaligned,                      /* input flow control */
    3 oflow bit (1) unaligned,                      /* output flow control */
    3 no_outp bit (1) unaligned,                    /* don't generate output parity */

    3 eight_bit bit (1) unaligned,                  /* don't strip input parity */
    3 odd_parity bit (1) unaligned,                 /* generate odd parity (if any) */

    3 modes_pad bit (7) unaligned,

  2 id char (4) unaligned,                          /* terminal id */

  2 colmax fixed bin (8) unaligned,                 /* current maximum number of columns */
  2 linemax fixed bin (8) unaligned,                /* current maximum number of lines/frame */
  2 wrt_lchar fixed bin (17) unaligned,             /* char within last write block */

  2 input_msg_size fixed bin,                       /* maximum input message size in chars */
  2 framing_chars unaligned,
    3 frame_begin char (1) unaligned,               /* frame-begin character */
    3 frame_end char (1) unaligned,                 /* frame-end character */
  2 max_output_block fixed bin (18) unsigned unaligned,  /* maximum size of output block in block_acknowledge */

  2 input_suspend_seq unaligned,                    /* sequence for input suspension */
    3 count fixed bin (9) unsigned,
    3 chars char (3),
  2 input_resume_seq unaligned,                     /* likewise for input resumption */
    3 count fixed bin (9) unsigned,
    3 chars char (3),

  2 output_suspend_etb_seq unaligned,               /* sequence for output suspension or end_of_block */
    3 count fixed bin (9) unsigned,
    3 chars char (3),
  2 output_resume_ack_seq unaligned,                /* likewise for resumption or ack */
```

```
      3 count fixed bin (9) unsigned,
      3 chars char (3),

    2 flags unaligned,                                  /* tty dim flag bits */
      3 breakall_enabled bit (1) unaligned,             /* channel is permitted to use breakall mode */
      3 dont_count_next bit (1) unaligned,              /* next output character is escaped */
      3 keyboard_locking bit (1) unaligned,             /* ON if doing keybd locking for ASCIi line type */
      3 no_printer_off bit (1) unaligned,               /* reject printer_off/printer_on orders */
      3 break_char_pending bit (1) unaligned,           /* break character is in preconverted buffer */
      3 uproc_attached bit (1) unaligned,               /* user process has attached device */
      3 block_acknowledge bit (1) unaligned,            /* block acknowledgement output protocol */
      3 flags_pad bit (27) unaligned,

    2 actshift bit (2) unaligned,                       /* tty shift, 00 none, 01 lower, 10 upper, 11 unknown */

    2 cumulative_meters,                                /* continuously running meters */
      3 read_calls fixed bin (35),                      /* number of calls to tty_read */
      3 write_calls fixed bin (35),                     /* number of calls to tty_write */
      3 read_chars fixed bin (35),                      /* after conversion */
      3 write_chars fixed bin (35),                     /* before conversion */
      3 read_time fixed bin (71),                       /* total time spent in tty_read */
      3 write_time fixed bin (71),                      /* total time spent in tty_write */
    2 saved_meters like tcb.cumulative_meters,          /* meters saved at last dialup */

    2 can_type fixed binary (9) unaligned unsigned,     /* type of canonicalization to use on this channel */
    2 pad1 bit (27) unaligned,                          /* to word boundary */
    2 time_dialed fixed bin (71);                       /* clock time of last copy_meters order */

/* END INCLUDE FILE ... tcb.incl.pl1 */
```

```
/* 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 wtcb 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 mcs_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_buf_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);

dcl 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) unal,                   /* flag(i) ON if fnp(i) configured */
    2 padb1 bit (28) unaligned,
    2 lct_ptr ptr,                                         /* pointer to logical channel table */

    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 nblocked fixed bin (35),                             /* number of times process output blocked */
```

```
2 minbuf fixed bin (18),                    /* min output buffer size */
2 totbuf fixed bin (35),                    /* divide by nblocked to get ave buffer size */

2 preconverted fixed bin (35),              /* number of converted chars held in tty_buf */
2 input_restart fixed bin,                  /* number of times tty_read had to start over */
2 output_restart fixed bin,                 /* number of times tty_write has had to start over */
2 output_buffer_overflow fixed bin,         /* number of times tty_write has run out of buffers */
2 read_time fixed bin (71),                 /* total time spent in tty_read */
2 write_time fixed bin (71),                /* total time spent in tty_write */

2 read_calls fixed bin (35),                /* number of calls to tty_read */
2 write_calls fixed bin (35),               /* number of calls to tty_write */
2 bfx fixed bin,                            /* used in calls to iobm */
2 nquits fixed bin (35),                    /* number of quits */
2 space_needed_data,
  3 space_needed bit (1) unal,              /* space_needed bit on in at least 1 lcte */
  3 space_needed_calls fixed bin (34) unal, /* meter of uses of this facility */
2 space_lock_count fixed bin (35),          /* count of times tty_buf.slock locked */
2 space_lock_wait_count fixed bin (35),     /* count of times necessary to loop to lock it */
2 space_lock_wait_time fixed bin (35),      /* total time looped trying to lock it */

2 alloc_calls fixed bin (35),               /* total number of allocations performed in tty_buf */
2 free_calls fixed bin (35),                /* total number of freeings in tty_buf */
2 alloc_time fixed bin (35),                /* time spent masked in tty_space_man$get entries */
2 free_time fixed bin (35),                 /* time spent masked in tty_space_man$free entries */
2 total_alloc_steps fixed bin (35),         /* number of steps thru free chain while doing above */
2 alloc_failures fixed bin (35),            /* number of unsuccessful attempts to allocate space */
2 cumulative_input_space fixed bin (71),    /* cumulative amount of space allocated for input */

2 cumulative_output_space fixed bin (71),   /* cumulative amount of space allocated for output */
2 cumulative_control_space fixed bin (71),  /* cumulative amount of space allocated by tty_space_man$get_space */
2 input_space_updates fixed bin (35),       /* number of increments to cumulative_input_space */
2 output_space_updates fixed bin (35),      /* number of increments to cumulative_output_space */
2 control_space_updates fixed bin (35),     /* number of increments to cumulative_control_space */
2 minimum_free_space fixed bin (18),        /* smallest amount of free space ever available */

2 current_input_space fixed bin (18),       /* amount of space currently allocated for input */
2 current_output_space fixed bin (18),      /* amount of space currently allocated for output */
2 current_control_space fixed bin (18),     /* amount of space currently allocated by get_space */
2 tty_lock_calls fixed bin (35),            /* number of calls to tty_lock$lock entries */
2 found_channel_locked fixed bin (35),      /* number of times tty_lock found channel already locked */
2 max_wait_time fixed bin (35),             /* longest time waited for any channel lock */
2 total_wait_time fixed bin (71),           /* total amount of time spent waiting for channel locks */

2 echo_neg_time fixed bin (71),             /* cumulative time spent doing echo negotiation */
2 echo_neg_interrupts fixed bin (35),       /* Echo-negotiated shipments */
2 echo_neg_r0_chars fixed bin (35),         /* Chars echoed by ring 0 */
2 echo_neg_mux_chars fixed bin (35),        /* Chars echoed by mux */
2 echo_neg_sndopt_restarts fixed bin (35),  /* Echo reinits */
2 echo_neg_mux_nonecho fixed bin (35),
2 echo_neg_entries fixed bin (35),          /* Entries into negotiate */

2 echo_neg_mux_inhibit bit (1) aligned,     /* For testing */
2 n_queued_interrupts fixed bin (35),       /* number of interrupts queued by tty_lock */
2 trace unaligned,                          /* tracing information */
  3 flags,
```

```
      4 enable bit,                                  /* global tracing control */
      4 default_mode bit,                            /* whether to trace channels by default */
      4 read bit,                                    /* read */
      4 write bit,                                   /* write */
      4 data bit,                                    /* buffers on reads and writes */
      4 control bit,                                 /* control, priv_control, and hpriv_control */
      4 modes bit,                                   /* (get set check)_modes */
      4 interrupt bit,                               /* interrupt, interrupt_later */
      4 init bit,                                    /* init_multiplexer, terminate_multiplexer */
      4 start bit,                                   /* start, stop */
      4 shutdown bit,                                /* shutdown */
      4 space_man bit,                               /* tty_space_man$* */
      4 pad_flags bit (6),
      3 data_offset bit (18),                        /* offset of tracing data */

    2 recoverable_error_severity fixed bin,          /* Syserr severity for recoverable MCS errors */

    2 timer_lock bit (36) aligned,                   /* Lock owned by mcs_timer */
    2 next_timer_offset bit (18) aligned,            /* Offset of next timer to come due */
    2 timer_count fixed bin,                         /* Number of timers outstanding */
    2 timer_process bit (36) aligned,                /* Who is doing timers? */

    2 timer_ev_chn fixed bin (71),                   /* How get get him */
    2 timer_lock_wait_time fixed bin (71),           /* CPU time spent spinning on timer lock */

    2 timer_lock_count fixed bin (35),               /* Number of times timer lock locked */
    2 timer_lock_wait_count fixed bin (35),          /* Number of times imer lock waited on */
    2 timer_call_time fixed bin (71),                /* CPU time spent in call side timer operations */

    2 timer_polling_time fixed bin (71),             /* CPU time spent polling (including channel_manager) */
    2 timer_set_calls fixed bin (35),                /* Number of calls to mcs_timer$set, set_wired */
    2 timer_reset_calls fixed bin (35),              /* Number of calls to mcs_timer$reset, reset_wired */

    2 timer_change_calls fixed bin (35),             /* Number of calls to mcs_timer$change, change_wired */
    2 timer_poll_calls fixed bin (35),               /* Number of calls to mcs_timer$poll */
    2 timer_error_calls fixed bin (35),              /* Number of mcs_timer calls ending with recoverable errors */
    2 timer_duplicate_pollings fixed bin (35),       /* Number of timer polling found in progress on other CPU */

    2 tty_area_lock like hc_fast_lock,               /* to prevent contention in allocating/freeing in tty_area */

    2 pad2 (13) fixed bin (35),

    2 free_space fixed bin;                          /* start of free space region */


%include hc_fast_lock;

/* END INCLUDE FILE ... tty_buf.incl.pl1 */
```

```
/* 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;                                    /* pointer which block entry is based on */
dcl  free_blockp ptr;                               /* pointer to head of free space chain */


dcl 1 free_block aligned based (free_blockp),       /* format of start of free block */
    2 next bit (18),                                /* foward pointer to next free block */
    2 size fixed bin;                               /* number of words in this block */


dcl 1 buffer based (blockp) aligned,                /* buffer definition */
    2 next fixed bin (18) unal uns,                 /* addr of next buffer */
    2 flags unaligned,
      3 end_of_page bit (1) unaligned,              /* buffer contains end of page */
      3 converted bit (1) unaligned,                /* buffer contains converted input */
      3 break bit (1) unaligned,                    /* buffer contains break character */
      3 mark bit (1) unaligned,                     /* buffer contains first character after "mark" */
      3 pad bit (2) unaligned,
    2 size_code fixed bin (3) unal uns,             /* (nwords/16) - 1 */
    2 tally fixed bin (9) unal uns,                 /* number of characters in buffer */
    2 chars (0:59) char (1) unaligned;              /* 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 hevent 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 rflag bit (1) unaligned,                   /* process blocked on input */
      3 wru bit (1) unaligned,                     /* reading answerback */

      3 hndlquit bit (1) unaligned,                /* on if in hndlquit mode */
      3 count_lines bit (1) unaligned,             /* on if tcb.linemax > 0 */
      3 line_status_present bit (1) unaligned,     /* fnp has sent line status */
```

```
      3 sync_line bit (1) unaligned,          /* channel is synchronous line type */
      3 breakall bit (1) unaligned,           /* channel is in breakall mode */
      3 scroll bit (1) unaligned,             /* channel is in scroll mode */

      3 negotiating_echo bit (1) unaligned,   /* ring zero to echo chars. */
      3 wake_tbl bit (1) unaligned,           /* on if in wake_tbl mode */
      3 allow_wakeup bit (1) unaligned,       /* on to allow input wakeup in wake_tbl mode */

      3 receive_mode_device bit (1) unaligned, /* device must be told to enter receive mode */
      3 mark_set bit (1) unal,                /* write_with_mark call outstanding */
      3 masked bit (1) unal,                  /* channel masked by FNP */

    2 dial_status_code fixed bin (8) unaligned, /* code returned by 355 after dialing a phone number */

    2 fblock fixed bin (17) unaligned,        /* oldest read pointer, block */
    2 lblock fixed bin (17) unaligned,        /* newest read block */

    2 fchar fixed bin (9) unsigned unaligned, /* first block char index */
    2 actline fixed bin (9) unsigned unaligned, /* line number of current line */
    2 actcol fixed bin (9) unsigned unaligned, /* tty column position */
    2 nramsgs fixed bin (9) unsigned unaligned, /* current number of read-ahead msgs */

    2 write_first fixed bin (17) unaligned,   /* first write block */
    2 write_last fixed bin (17) unaligned,    /* last write block */
    2 write_cnt fixed bin (17) unaligned,     /* count of chars in write chain */
    2 white_col fixed bin (17) unaligned,     /* column position resulting from trailing white space */

    2 max_buf_size fixed bin (9) unal uns,    /* maximum-size buffer to be allocated for this channel */
    2 buffer_pad fixed bin (9) unal uns,      /* amount of pad to be left in output buffers */
    2 devx fixed bin (17) unaligned,          /* index into lct of channel's entry */
    2 echdp bit (18) unaligned,               /* echo negotiation data ptr */
    2 waketp bit (18) unaligned,              /* wakeup table offset */
    2 prompt_len fixed bin (8) unaligned,     /* number of chars in prompt string */
    2 prompt char (3) unaligned,              /* text of prompt message */
    2 line_delimiter char (1) unal,           /* line delimiter for tty_read parse */
    2 more_flags unaligned,                   /* in addition to flags (above) */
      3 line_status_disabled bit (1) unaligned, /* "1"b => don't relay line_status interrupts */
      3 pad bit (26) unal,
    2 error_code fixed bin (35),              /* error code returned by channel_manager to tty_interrupt */
    2 pad2 bit (36);

/* Ends on doubleword boundary */

/* END INCLUDE FILE ... wtcb.incl.pl1 */
```