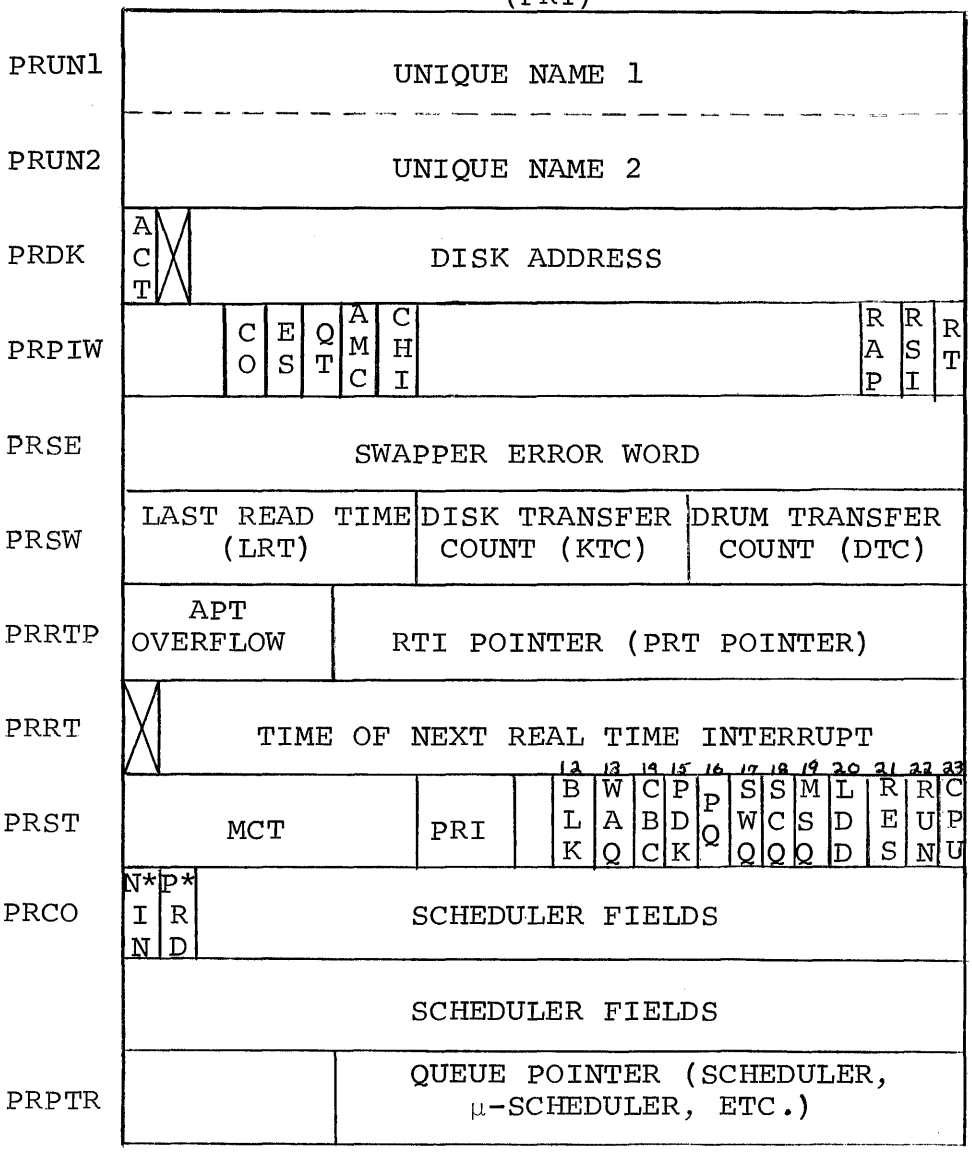
bcc	title	prefix/class-number.revision
	System Parameters	SYSP/W-15.1

checked	authors	approval date	revision date
checked	Rainer Schulz		9/20/69
approved Mel		classification Working Document	
		distribution Company Private	pages 12

ABSTRACT and CONTENTS

This document describes the PRT (process table) entries, fixed core allocation and the protect bus conventions.

FORMAT OF THE PROCESS TABLE
(PRT)



*Bits only looked at by software

DESCRIPTION OF PRT BITS

CO : Carrier Off Interrupt

ES : Escape Interrupt

QT : Quit Interrupt

CHI : CHIO Interrupt

AMC : AMC Interrupt

RAP : Reduce Active Page Set

RSI : Run Scheduler Initially

RT : Real Time Interrupt

MCT : Millisecond Compute Time

PRI : Micro Sched. Priority

BLK : Blocked

WAQ : Wake-up Queue

CBC : Context Block Considered

PDK : Process Delayed for Disk Transf.

PQ : Process Queued on Sect. Rd. List

SWQ : On Swapper Request Queue

SCQ : On Scheduler Queue

MSQ : On Micro Scheduler Queue

LDD : Process Loaded

RUN : Process Running

CPU : CPU Number

RES : Resident Process

NIN : Non-Interactive

PRD : Process to be Destroyed

ACT: Active Process in PRT

PROTECT, Request Strobe

The value of the protect is taken from the X-bus in the μprocessor. Protect 1 is the least significant bit of the X-bus, etc.

PROTECT 1: PRT, Wake-up Queue, Drum Bit Table

2: Swapper Request Queues, CHT

3: CPU lock

4: CHIO Line Table, μ-Sched. Request Stack

REQUEST STROBE 1: AMC

2: μ-Sched

3: CHIO

4: CPUØ

5: CPU1

Octal Memory Location		Description
ø		
1		
2		
3		
4		
5	(SRMEM)	Switch register
6	(CPUø)	CPUø
7	(CPU1)	CPU1
10-11	(RTC)	Real Time Clock
12-13	(RTCL)	Relocation bias for Real Time Clock
14	(USIBASE)	Beginning of μ-scheduler stack pointer
15	(USIBTOP)	Request number for μ-scheduler stack pointer
16-17	(RTQ)	Real Time queue header
20-24		Break addresses
25-27		Break wait
30-37	(USRADR)	System Registers
40-77	(CPUIT)	CHIO - storage
100-101	(AMCA)	AMC activate cells
102-103	(AMCQ)	AMC activate queue header
104-105	(SWAPRQ)	AMC request queue for general requests
106-107	(SWAPIN)	Swap-in queue header (AMC)
110	(DHTS)	Base address of DHT1
111	(DHTS2)	Base address of DHT2
112	(DHTSI)	DHT1 size
113	(DRSCR)	Base address of drum sector read list
114	(FRDR)	Base address of free drum page table
115	(DKCQ)	Base address of disc cylinder queues
123	(NFSWQN)	Free node list count (AMC)
124-125	(SWFREL)	Free node list header (AMC)
126-277		General AMC storage
300-301	(WAKEUPQ)	Wake-up queue header
302	(PRTB)	PRT base address
303		
304-305	(SCHEDQ)	Scheduler queue
306	(ITPREL)	
307	(BND)	
310-327	(USCHQ)	μ-scheduler queues
330-377		Scheduler and μ-scheduler storage
400-777	(CHTS)	CHT1
1000-2377	(CHTS2)	CHT2
2420-2437	(USIB)	μ-scheduler request stack
2440-2463	(ERR)	
2500-2617		AMC state
2620-2737		μ-sched state
2740-3057		CHIO state

FIXED REAL MEMORY ALLOCATION

CB STORAGE ALLOCATION

SPT (42 words/entry)	336
OFT (5 words/entry)	50
PMT (4 words/entry) (128)	512
APT (1 word/entry)	66
SPCS (5 words/entry)	80
	1044
MAP	64
STATE, ETC.	25
	1197

CB CONTENTS

Ø : POP entry indirect address word
1 : POP entry indirect address word

2 : SP first unused stack address

3 : SL last word allocated for stack

4 : P for Trap (ring dependent)

5 : PAR for Trap (ring dependent)

6 : BRU for Trap (ring dependent)

7 : BRU for Trap (ring dependent)

FREE : 1Ø-177

MAP : 200-277

PMT : 300-1277

SPT : 1300-2117

SPCS : 2120-2237

ICT : 2240-2307

OFT : 2310-2371

STACK : 2372-2647

APT : 2650-2751

TRSTATE : 2752-2763

SWSTATE : 2764-2775

CTC : 2776

IT : 2777

RESIDENT TABLES

Teletypes	:	2 pages
PRT	:	5 pages
Drum & Disk		
bit table	:	2 pages
DHT	:	(7 pages)
SWAPPER free		
list	:	2 pages
		11 (18) pages
Monitor		10 pages
		21 (28) pages

Notes about unique names and access keys.

(See UNIQUE NAME TABLE)

All UN's from Ø to 2B5-1 are reserved for MIB unique names. These numbers correspond to the user number that the MIB is assigned to.

Since the first 64,000 numbers are used, file and CB unique names have to start after these numbers.

The unique name generator has to be initialized to 1ØØØB. The first unique name (48 bits) created from the UN generator will thus be 4B5.

The difference between 2B5 which is the next UN after the MIB's and 4B5, which is the first unique name of the UN generator is there for fixed small files which get loaded at system startup.

The tag of the unique names (bits Ø and 1) makes no distinction between small files and MIB's. The only way to distinguish these two objects is by comparing UN values.

UN's of MIB's $<$ 2B5, and

UN's of small files $\geq$ 2B5.

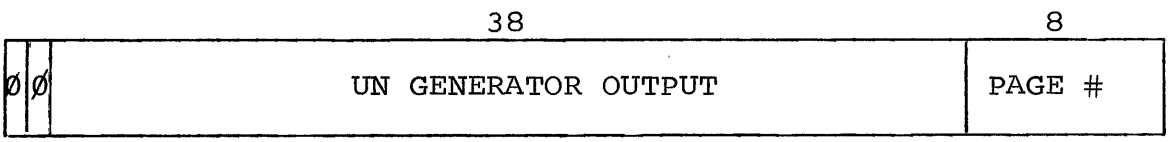
When the system creates large files it acquires values from the UN generator output until the least significant 3 bits of this output are zero. This is done to accomodate 2048 pages in large files.

Access keys are 38 bits long and are taken from the output of the unique name generator.

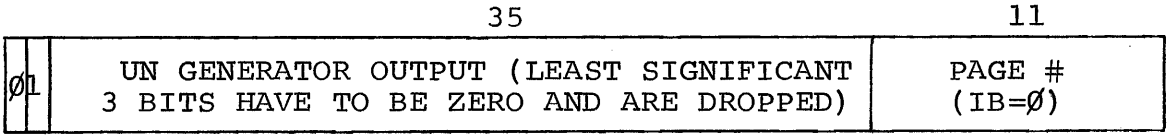
UNIQUE NAME TABLE

UN GENERATOR SENDS 38 BITS

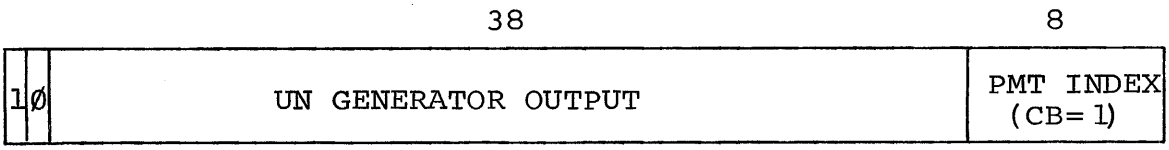
SMALL FILE



LARGE FILE



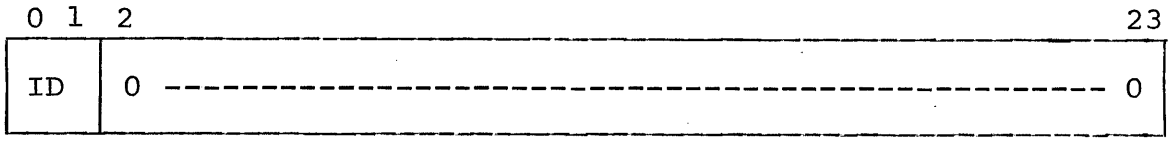
CONTEXT BLOCKS, PRIVATE MEMORY

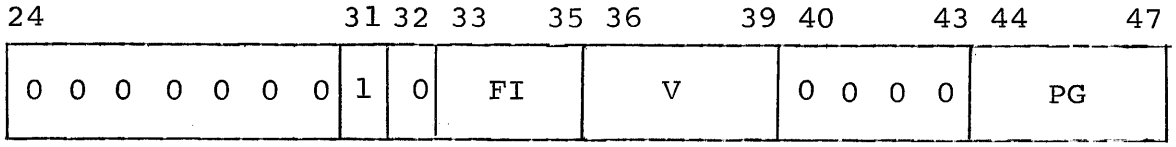


MIB's

Unique Names for System Initialization Files

Unique names between 2B5 and 4B5 are reserved for system initialization files (Monitor, Utility, CB, etc.). The format of these unique names is as follows:





ID : Unique name identification (file, MIB, CB, etc.)

FI : 001 for Monitor
010 for Utility
011 for CB
100 for Real Core

V : Version number (also specifies file locations on disk or drum)

PG : Page number in file

The structure of these unique names has several implications.

The number of objects loaded from tape or drum at system initialization is limited to 7 (as given by FI). FI = Ø is reserved for the MIB loaded with the system. The size of these files is limited to 16 pages. The files can actually contain more pages, as long as these pages are not loaded with the AMC or tape load procedure. Special provisions can be made to get around this problem if it becomes necessary to do so.

Accessing Protected Objects in the System

There are several tables in the system which are accessed by the CPU's while executing processes. Particularly CHT is used for loading the hardware map. If the monitor sets the protect signal for the CHT, then it has to be extremely careful not to cause the CPU to have to load the map, because the CPU would reset the protect signal for CHT after loading the map. This would leave CHT unprotected while the monitor references it.