Setup & Use Of
Carrier Performance Monitoring Archival (PMA)
For SPMs

Release: 1.0.5
Publication Date: October 19, 2001

Published in the United States

Document History

Issue No. Rel Date	Reason(s) for Reissue	Author and Department
1.0.0 September 21, 2001	Original	Eddie Bridges
1.0.1 September 24, 2001	Increased font size of document version/date/status fields. Other miscellaneous adjustments.	Eddie Bridges
1.0.2 September 24, 2001	Added additional text to section 2.4. Also bolded the word "fail" in section 2.5.	Eddie Bridges
1.0.3 September 26, 2001	Added specific commands to DISKADM in section 5. Changed 'P' setting in table DRMAPPL to Mandatory and added a warning message.	Allen Johnson
1.0.4 October 12, 2001	Changed wording of CLOSTATE warning, clarified volume configuration information.	Allen Johnson
1.0.5 October 19, 2001	Changed recommended alarm thresholds for table DRMAPPL.	Allen Johnson

Table Of Contents

1.0 Introduction

This document describes how to setup and use performance monitoring archival data collected by the carriers for SPMs. The data collected from carriers allows the telco to diagnose and correct problems which affect the health of the carriers of the peripheral. The data is a collection of parameters (i.e. CV, ES, SES, UAS, etc.) that start counting at zero and increment one at a time. Zero values mean there have been no problems. The optical parameters (OPT, OPR, LBC) are the only exception to this rule because they are calculated as percentages and stay at approximately 100%.

This step by step guide begins with the setting up of where the PMA data will be stored. When the disk volume storage mechanism, known as the Distributed Recording Manager (DRM), is setup and mounted, PMA collection begins. Data will be collected from all carriers on all SPMs every 15 minutes as well as every 24 hours. Data collection will continue indefinately until DRM volumes are unmounted.

The performance monitoring 15 minute data (PM15) and the performance monitoring 24 hour data (PM24) can be obtained and analyzed in one of two ways. The first way is to FTP the files to a machine and extract the information with a another program. Sample C code to perform this extraction is included at the end of this document. The second way to analyze the data is by using the ReachThrough feature of the SDM. The SDM responds to querys for performance monitoring information from a clients TCP/IP connection and asks the core for the data. The data is then returned to the user.

2.0 Setting Up PMA

The PMA system needs to know where to store the data it is going to collect. Tables DRMAPPL and DRMPOOL are already present in the load with default values. This step sets up the tables with the correct volume and naming information. This step only needs to be performed once. Subsequent restarts will maintain the settings that are selected in this step.

2.1 Ensure that table DRMAPPL is setup correctly for PM15 and PM24 like the following.

```
>table drmappl
JOURNAL FILE UNAVAILABLE - DMOS NOT ALLOWED
TABLE: DRMAPPL
>list all
TOP
APPLNAME  GROUPID APPLID POOLNAME ALARMMN ALARMMJ ALARMCR RETPD FILEDATE
SHEDDAYS SHEDBASE SHEDINCR CLOSTATE MAXFSIZE FORCBKUP
--------------------------------------------------------------------------
    PM15    SPMPM      0 PM15POOL      10      5       1    1   OPENED
 YYYYYYY        3      X24        P      100        N

    PM24    SPMPM      1 PM24POOL      5      3       1    7   OPENED
 YYYYYYY        3      X24        P      100        N

      OM      SOM      2 OMPOOL        50      20       1    30   OPENED
 YYYYYYY        3      X24        P      100        Y

BOTTOM
```

Field Recommendations and Explanations:

ALARMMN (Minor Alarm Threshold) **- Triggers a Minor alarm if volume free space drops below this value (in Mb). Recommended Setting: "10" for PM15, "5" for PM24.**

ALARMMJ (Major Alarm Threshold) **- Triggers a Major alarm if volume free space drops below this value (in Mb). Recommended Setting: "5" for PM15, "3" for PM24.**

ALARMCR (Critical Alarm Threshold) **- Triggers a Critical alarm if volume free space drops below this value (in Mb). Recommended Setting: "1" for PM15 and PM24.**

RETPD (Retention Period) - **Recommended Setting: "1" for PM15 and "7" for PM24**

CLOSTATE (Closed State) - **Mandatory Setting: "P"**

- "P" for Processed State: These files can have data extracted.
- "U" for Unprocessed State: These files can only be deleted with special commands and are unable to have the data extracted.

WARNING: If CLOSTATE is not set to "P", the data will not be able to be extracted, and a count of 0 records will be shown even if data has been collected.

FORCBKUP (Force Backup) - **Recommended Setting: "N"**
- "N" files can be deleted without being backed up
- "Y" files will not be deleted until they are backed up

2.2 Table DRMPOOL should already have the following setup for PM15 and PM24.

```
>table drmpool
JOURNAL FILE UNAVAILABLE - DMOS NOT ALLOWED
TABLE: DRMPOOL
>list all
TOP
POOLNAME VOLUME0 VOLUME1 VOLUME2 VOLUME3 VOLUME4 VOLUME5 VOLUME6 VOLUME7
VOLUME8 VOLUME9 VOLUME10 VOLUME11 VOLUME12 VOLUME13 VOLUME14 VOLUME15
VOLUME16 VOLUME17
--------------------------------------------------------------------------
PM15POOL       $       $       $       $       $       $       $       $
      $       $       $       $       $       $       $       $
       $       $

PM24POOL       $       $       $       $       $       $       $       $
      $       $       $       $       $       $       $       $
       $       $

BOTTOM
```

2.3 List the disk volumes using the diskut command.

If volumes for PM15 and PM24 data do not already exist, they will need to be created before continuing on (see diskut help and/or diskadm help for creating volumes). The volumes do not necessarily have to have the words PM15 or PM24 in their names, but it makes it easier to find where the PMA data resides later on.

The sample data volumes listed below use volumes F02LPM15, F02LPM24, F17LPM15, and F17LPM15. This is one of many configurations. Another configuration could be having one volume allocated for PM15 and another for PM24.

The volume sizes shown below are a default, and allow for a possible 84 SPMs in the office with 181 carriers each. PM15 also accounts for data which has to be preserved for 1 day at a minimum while PM24 preserves data for a minimum for 7 days.

```
>diskut
Disk utility is now active.
DISKUT:
>lv
Volumes found on the node CM:
-----------------------------

NAME                 TYPE      TOTAL       FREE TOTAL  OPEN  ITOC    LARGEST
                             BLOCKS     BLOCKS FILES FILES FILES  FREE SEGMENT
-------------------------------------------------------------------------------
F17LIMAGE            FTFS    4194304    3122176    10     0     2    1884160
F17LUSER             FTFS    3145728    1570480   118     1     0     182784
F17LTEST             FTFS    2097152    1960120    40     0     0    1914040
F17LDLOG             FTFS     204800     203960     0     0     0     203960
F17LPTCH_CM          FTFS     204800     198456    42     0     0     198200
F17LPTCH_ISN         FTFS     204800     203960     0     0     0     203960
F17LDATA             FTFS     102400      71856   149     0     0      71856
F17LPM15             FTFS      51200        296   144     0     0        296
F17LPM24             FTFS      20480      13240     9     0     0      13240
F17LSHRK             FTFS      20480      19640     0     0     0      19640
F17LHELP             FTFS      20480      15288    27     0     0       9912
F02LIMAGE            FTFS    4194304    3628032    11     0     1    1233920
-------------------------------------------------------------------------------
Do you wish to continue ??
Please confirm ("YES", "Y", "NO", or "N"):
>y
Volumes found on the node CM:
-----------------------------

NAME                 TYPE      TOTAL       FREE TOTAL  OPEN  ITOC    LARGEST
                             BLOCKS     BLOCKS FILES FILES FILES  FREE SEGMENT
-------------------------------------------------------------------------------
---
F02LUSER             FTFS    3145728    1438384   118     0     0     149688
F02LTEST             FTFS    2097152    1849528    33     0     0    1827000
F02LDLOG             FTFS     204800     203960     0     0     0     203960
F02LPTCH_CM          FTFS     204800     200248    29     0     0     200248
F02LPTCH_ISN         FTFS     204800     203576     3     0     0     203576
F02LDATA             FTFS     102400      79416    27     0     0      73016
F02LPM15             FTFS      51200        432   195     0     0        432
F02LPM24             FTFS      20480       6072    53     0     0       6072
F02LSHRK             FTFS      20480      19640     0     0     0      19640
F02LHELP             FTFS      20480      16824    16     0     0      15928

Total number of volumes found on node CM : 22
```

2.4 Using the information from the diskut command, the PM15 and PM24 volumes need to be mounted so the PMA system will begin using them. Start by going to the DRM map level.

```
CI:
>mapci;mtc;appl;oamap;drm
MAPCI:
MTC:
APPL:
OAMAP:
DRM:
```

2.5 Verify the DRM registration information.

The following commands should **fail** at this time.

```
>info pm15 vol
OPERATION FAILED
There are no volumes allocated to this stream.

>info pm24 vol
OPERATION FAILED
There are no volumes allocated to this stream.
```

2.6 Mount the PM15 and PM24 volumes.

By mounting these volumes, table DRMPOOL will be automatically updated. PMA will also be activated.

The only way to de-activate PMA is to unmount these volumes (See the section titled "Stopping PMA").

```
>mount pm15 F02LPM15 cm
Sending request to CM...
Request sent.  (Please wait...)

     MOUNT OPERATION INITIATED.

     Volume F02LPM15 has been allocated to PM15POOL.
     The MOUNT may take a few seconds to complete.
```

```
>mount pm24 F02LPM24 cm
Sending request to CM...
Request sent.  (Please wait...)

     MOUNT OPERATION INITIATED.

     Volume F02LPM24 has been allocated to PM24POOL.
     The MOUNT may take a few seconds to complete.

>mount pm15 F17LPM15 cm
Sending request to CM...
Request sent.  (Please wait...)

     MOUNT OPERATION INITIATED.

     Volume F17LPM15 has been allocated to PM15POOL.
     The MOUNT may take a few seconds to complete.

>mount pm24 F17LPM24 cm
Sending request to CM...
Request sent.  (Please wait...)

     MOUNT OPERATION INITIATED.

     Volume F17LPM24 has been allocated to PM24POOL.
     The MOUNT may take a few seconds to complete.
```

2.7 Although not needed, additional verification of DRM can be performed.

The commands below will successfully return information relating to the newly mounted PM15 and PM24 volumes.

```
>mapci;mtc;appl;oamap;drm
>info pm15 reg
>info pm15 nodes
>info pm15 vol
>info pm24 reg
>info pm24 nodes
>info pm24 vol
```

2.8 Table DRMPOOL will show the updated mounting results.

```
>table drmpool
JOURNAL FILE UNAVAILABLE - DMOS NOT ALLOWED
TABLE: DRMPOOL
>list all
```

```
TOP
POOLNAME        VOLUME0        VOLUME1        VOLUME2        VOLUME3
       VOLUME4        VOLUME5        VOLUME6        VOLUME7        VOLUME8
       VOLUME9       VOLUME10       VOLUME11       VOLUME12       VOLUME13
      VOLUME14       VOLUME15       VOLUME16       VOLUME17
--------------------------------------------------------------------------
PM15POOL :CM/F02LPM15,1 :CM/F17LPM15,1             $              $
             $              $              $             $             $
             $              $              $             $             $
             $              $              $             $

PM24POOL :CM/F02LPM24,1 :CM/F17LPM24,1             $              $
             $              $              $             $             $
             $              $              $             $             $
             $              $              $             $
BOTTOM
```

3.0 Starting And Stopping PMA

PMA collects continuously after it is set up (after the mount commands have been issued).

To fully stop PMA demount all volumes:

```
>mapci;mtc;appl;oamap;drm
>demount pm15 F02LPM15
>demount pm15 F17LPM15
>demount pm24 F02LPM24
>demount pm24 F17LPM24
```

To start PMA again, repeat the mount process for DRM as shown in step 2.6

4.0 Accessing PMA Data One Carrier At A Time

With the successful mounting of the volumes in DRM, PMA will record data every 15 minutes. All nodes in Table MNNODE will record carrier performance monitoring information. Table MNHSCARR lists the carriers and their circuit numbers. Please note that only carriers that are provisioned will have information collected for them.

The PMA data can be retrieved one carrier at a time using the following method.

a. Go into the carrutil ci increment.

>**carrutil**

b. Set the carrier using the settcarr command.

>**setcarr spm <node number> <circuit number>**

c. Issue the gethist command to retrieve the PM information for that carrier. Parameters must be provided depending on whether you want near end (NE) or far end (FE) information for min (15 minute PM information) or day (24 hour PM information).

>**gethist <NE/FE> <MIN/DAY>**

d. Issue the listhist command to display the information.

>**listhist**

See the following example for a carrier which has collected 3 periods of data for 3 different 15 minute intervals.

WARNING!! The PM15 and PM24 volumes must be mounted for these commands to provide data to the gethist and listhist commands. If they are not mounted a count of "0" records will appear even if data has been collected.

```
>carrutil
SPM Carrier Utilities

CARRUTIL:
>setcarr spm 27 169   (Note: 27 is the node number & 169 is the circuit number)
SPM   27 RM     0 OC3S   0
SPM   27 169 : Name SPM_27_OC3S_1
```

```
>gethist ne min
SPM   27 RM     0 OC3S   0                    NE
 3 records collected

>listhist
SPM   27 RM     0 OC3S   0                    NE
Archived 15 Minute Performance Monitoring Data Report
Generated at 01/09/12 18:36.
Ended  secs I   LBC-N   OPT-N   OPR-N    CV-N    ES-N   SES-N  SEFS-N
18:30   902        90      98      97       0       0       0       0
18:15   902        90      98      97       0       0       0       0
18:00   893        90      98      97       0       0       0       0
End of report.

>gethist fe min
SPM   27 RM     0 OC3S   0                    FE
 3 records collected

>listhist
SPM   27 RM     0 OC3S   0                    FE
Archived 15 Minute Performance Monitoring Data Report
Generated at 01/09/12 18:36.
Ended  secs I
18:30   902
18:15   902
18:00   893
End of report.
```

To see another carrier's data simply type the setcarr command again and use the gethist and listhist commands to view the data.

5.0 Deleting PMA Files

The DRM system keeps the files cleaned up on the volumes. However, should something go wrong, the following commands specify how to delete the PMA files.

DRM FTFS files are written with write protection to ensure that they cannot be deleted outside of DRM.

There are two ways to remove these files: BFT or DISKADM.

5.1 Use DISKADM to reinitialize the volume.

This will delete all the files on the specified volume.

```
>diskadm f02l
>reinitvol pm15
>reinitvol pm24
>quit
>diskadm f17l
>reinitvol pm15
>reinitvol pm24
>quit
```

5.2 Use BFT to set yourself as DRM and delete the files.

Enter the BFT level from the CI prompt.

>**BFT**

Change to the CM directory and list the contents of the volume.

>**cd CM; ll <volume name>**

This will list out the files in the volume similar to UNIX format. Pay attention to the 2 fields UID and GID. These are the user id and group id corresponding to the owner, who is DRM in this case.

Login as DRM using the UID and GID listed above.

>**su UID# GID#**

Remove the offending file using the rm command with the full path.

>**rm ::CM/S00DGCDR1/UN990216122900GCDR**

```
Example:
>bft
BFT (Test)

>cd CM
Current working directory:  :CM

>ll ::CM/F17LPM15
Volume type: FTFS disk.
SEC MODES FT LNKS UID  GID  BYTES     LAST MODIF  NAME
--------- -- ---- ---- ---- --------- ----------- ----
rwxrwxr-- BD    1  128   28      8192 08/17 09:50 .
rwxr--r-- BG    1  514   28     14336 06/28 07:20 UN010628070600PM15
rwxr--r-- BG    1  514   28     16384 06/28 07:34 UN010628073401PM15
rwxr--r-- BG    1  514   28     16384 06/28 07:49 UN010628074902PM15

>su 514 28
User identity: Working user
UID =        514  GID =         28

>cd F17LPM15
Current working directory:  :CM/F17LPM15

>rm UN010628070600PM15
Removing  :CM/F17LPM15/UN010628070600PM15
RESULTING RETURN CODE: 0 : 0 ()

>rm ::CM/F17LPM15/*            (this command removes multiple files)
Removing  UN010628073401PM15
Remove  UN010628074902PM15  ?
Please confirm ("YES", "Y", "NO", or "N"):
>y
Removing  UN010628074902PM15
Remove  UN010628080503PM15  ?
Please confirm ("YES", "Y", "NO", or "N"):
>y

>ll
Volume type: FTFS disk.
SEC MODES FT LNKS UID  GID  BYTES     LAST MODIF  NAME
--------- -- ---- ---- ---- --------- ----------- ----
rwxrwxrwx BD    1    0    0      8192 08/17 09:50 .
```

6.0 Obtaining PMA Data By FTP

PMA data files can be retrieved by FTPing them from the switch. Only PMA files which being with "P" will be able to have data extracted from them. The naming convention of the files is as follows:

```
XBYYMMDDHHMNSQAPPL

X:    File Status: A(Active), U(Unprocessed), P(Processed), R(Remove)
B:    File Backup status: B(Backup), N(No Backup)
YY:   Year
MM:   Month
DD:   Day
HH:   Hour
MN:   Minute
SQ:   Sequence Number, used in maintaning chronological order
APPL: DRM Application Name
```

To FTP the files do the following:

a. FTP to the CM.

>**ftp <IP address of CM>**

b. CD to the volume containing the desired files.

ftp>**cd <PMA volume name>**

c. Use the ls command to list the files in the volume.

d. Use the get command to retrieve the desired files in BIN (Binary) mode.

e. The files can be interpreted by using the source code for the PMparse program located at the end of this document.

f. To use PMparse:

>**PMparse <file name>**

e.g. >**PMparse PN010913190411PM15**

The results will be output to the screen.

Example of ftp session and PMparse:

```
unix-21> ftp 100.1.1.1
Connected to 100.1.1.1.
220 Welcome to Northern Telecom Supernode FTP BCS X
Name (100.1.1.1:b1b1):
331 Password required.
```

```
Password:
230 User logged in, proceed
Remote system type is SOS.
ftp> cd /F02LPM15
200 Directory changed.
ftp> ls
200 PORT Command okay
150 File status okay; about to open data connection.
PN010913175807PM15               FIXED    1024       12
PN010913183409PM15               FIXED    1024       12
226 Request successful. Closing data connection.
ftp> bin
200 Command okay.
ftp> get PN010913175807PM15
200 PORT Command okay
150 Opening BIN mode connection.
226 Request successful. Closing data connection.
12288 bytes received in 0.81 seconds (14.83 Kbytes/s)

unix>PMparse PN010913175807PM15

Period ending: 2001/09/21 15:45:00.000 MON.

================================================================================
  SPM  0 Interval type: 15 Minute Duration: 900 secs
    OC3S  0   NE
          LBC        OPT        OPR         CV         ES        SES       SEFS
            9         10         11         12         13         14         15

   STS3L  0   NE
             CV          ES         SES         UAS         PSC
              1           2           3           4           5

   STS1P  2   NE
             CV          ES         SES         UAS
          65280       65280       65280       65280

   STS1P  2 VT15P  1  DS1P  1   NE
             CV          ES         SES         UAS        AISS         CSS
              9          10          11          12          13          14
```

7.0 Obtaining PMA Data By Using The SDM

7.1 TL1 Command Overview

This section is intended to be used as a reference for TL1 commands.
The customer inputs the TL1 query commands from their workstation. The client process running on the customer node writes the TL1 input to server process on SDM via TCP/IP. The SDM server is recognized by the client via its IP address. The interface between customer node and SDM is implemented via TCP/IP client server model.

TL1 input syntax:
<cmd>:<tid>:<aid>:<ctag>::[<montype>,<monlev>,<locn>,<dirn>,<tmper>,<mondat>,<montm>];
In this release we support the following TL1 commands:

RTRV-PM-OC3,
RTRV-PM-STS3,
RTRV-PM-STS1,
RTRV-PM-DS3,
RTRV-PM-DS1,
RTRV-PM-VT15,

(for these command & parameter detail please refer to Please refer to “Table 1 TL1 Commands & Parameters Description” on page 22.)

TL1 Response Syntax:
<cr> <lf> <lf>
^^^ <tid> ^ <date> ^ <time> <cr> <lf>
M^^ <ctag> ^ COMPLD <cr> <lf>
^^^”<aid>:<num_records> <cr> <lf>
^^^ “text_result” <cr> <lf>
;
where:
<cr> : carriage return
<lf> : line feed
<tid> : target ID of NE (SPM-#)
<date> : the date (YYYY-MM-DD) when the command is executed
<time>: the time (HH-MM-SS) when the command is executed
<ctag>: correlated tag which specified the response and the input command
COMPLD : the result of the command execution (COMPLD for successful, DENY for failure, or PRTL for Partial successful)
<tex_result>: the result from the excution
; :the terminator of the output message

TL1 Command Example:
RTRV-PM-OC3:SPM-1:OC3-1:123: :CV,0-UP,NE,TRMT,MIN, , ;
(<mondate> and <montm> are left blank. That means this is current query request)

TL1 Normal Response Example:
<cr> > <lf> <lf>
^^^ CM-SPM-1^2000-02-21^10:15:25 <cr> <lf>
M^^123^COMPLD <cr> <lf>
^^^"OC3-1:CV,5, , NEND,RCV, 15-MIN, 02-21, 10-10-25" <cr> <lf>
;
TL1 Error Response Example:
<cr> <lf> <lf>
^^^CM-SPM-1^ 2000-02-21^10:25:15 <cr> <lf>
M^^ 123^ DENY <cr> <lf>
^^^IPNV <cr> <lf>
^^^" :PRMTR=AID" <cr> <lf>
^^^/* A parameter name appearing in an input command is not valid */

Table 1 TL1 Commands & Parameters Description

field	value	description
<cmd>	RTRV-PM-OC3 RTRV-PM-STS1 RTRV-PM-STS3 RTRV-PM-DS1 RTRV-PM-VT15 RTRV-PM-DS3	retrieve performance of OC3S carrier retrieve performance of STS1P retrieve performance of STS3L retrieve performance of DS1P retrieve performance of VT15P retrieve performance of DS3P carrier
<tid>	SPM-#	a target ID, which include SPM and spm number
<aid>	OC3-oc3_l-sts1_l-vt15_l-ds3_l-ds1_l	a target component, which includes OC3 and the hierarchy of location of the target component. See Please refer to "Figure 2 Hierarchy SPM Carriers" on page 26. for more detail
ctag	any char or number up to 9 character	A correlated tag to identify the response with the TL1 input command

Table 1 TL1 Commands & Parameters Description

field	value	description
<montype>	LBC OPT OPR CV ES SES SEFS UAS PSC AISS CSS	Laser Biase Current Optical Power Transmission levelOptical Power Receive level Coding Violation count Errored Second count -Line Severe Errored Second count - Severely Errored Framing Second Unavailable Second count Protection Switch Count Alarm Indication Signal Second- Control Slip Second count
<monlev>	0-UP 1-UP	all monitored parameter bins Non-zero bins only,default = 1-up
<locn>	NE FE	Near end, default = NE Far end
<dirn>	TRMT RCV	Transmit direction Receive direction
<tmper>	MIN DAY	15 Minute Interval of the last 8 hours, default=MIN Day Interval of the last 7 days
<mondat>	YYYY-MM-DD	is the date of the beginning of the historical requested PM period specified in <tmper>
<montm>	HH-MM-SS	is the beginning time of day of the historical requested PM period specified in <tmper>

Table 2 Parameter Definition

PARAMETER	VALUE	DEFINITION
CARRNAME	NAME	constant used to denote the desire to post a carrier by its name (OPTIONAL)
THE_NAME	string	the logical name of an SPM carrier; a string of up to 38 characters (OPTIONAL)
PM	(see below)	the peripheral type (OPTIONAL)
	SPM	an SPM

PARAMETER	VALUE	DEFINITION
NO	0 to 63	the peripheral number (REQUIRED) If a peripheral type is specified, the user MUST specify a peripheral number.
CARRIER	(see below)	the starting point from which the carrier's Carrier Payload Description position will be defined (REQUIRED)
	OC3RM	If a peripheral type and peripheral number are specified, the user MAY choose to specify a physical carrier starting with the OC3RM.
	STS1P	If a peripheral type and peripheral number are specified, the user MAY choose to specify a logical carrier, starting with the STS1P.
PACKNO	0 to 1	OC3RM number (REQUIRED) If OC3RM is specified, the user MUST specify the OC3RM number.
CKTNO	0 to 28 The minimum and maximum values depend on the parent carrier and the PCM type.	the circuit payload number (REQUIRED) If a PCM type is specified, the user MUST specify the circuit payload number for each PCM type specified.
PCMTYPE	(see below)	physical PCM types (OPTIONAL once the OC3S has been specified)
	OC3S	OC3-Section (carrier type) If an OC3RM is specified, the user MUST specify OC3S.
	STS3L	STS3-Line (carrier type) If an OC3S is specified, the user MAY specify STS3L.
PCMTYPE	(see below)	logical PCM types (OPTIONAL)
	DS3P	DS3-Path (carrier type) If an STS1P is specified, the user MAY specify DS3P.
	VT15P	VT15-Path (carrier type) If an STS1P is specified, the user MAY specify VT15P.

PARAMETER	VALUE	DEFINITION
	DS1P	DS1-Path (carrier type) If a DS3P or VT15P is specified, the user MAY specify DS1P.
CKTID	0 to 181	the carrier number (OPTIONAL) If a peripheral type and peripheral number are specified, the user MAY choose to specify the carrier via its identifier.

Table 3 Parameter Direction and Counts Definition

PARAMETER	VALUE	DEFINITION
DIRECTION	(see below)	identify the monitored entity of the carrier in context for which archived Performance Monitoring data is to be displayed (REQUIRED)
	NE	Near End Monitored Entity
	FE	Far End Monitored Entity
COUNTS	(see below)	identify which set of archived Performance Monitoring data is to be displayed (REQUIRED)
	MIN	display the archived 15 minute Performance Monitoring data Up to eight hours of data will be displayed.
	DAY	display the archived 24 hour Performance Monitoring data Up to seven days of data will be displayed.

Table 4 Monitor type & Carrier type

Carrier type	Direction	Monitor type
OC3	NE	LBC, OPT, OPR, CV, ES, SES, SEFS
STS3	NE	CV, ES, SES, UAS, PSC
STS1	NE	CV, ES,SES, UAS
STS1	FE	CV, ES, SES, UAS
DS3	NE	CV, ES, SES, UAS

Carrier type	Direction	Monitor type
DS1	NE	CV, ES, SES, UAS, AISS, CSS
VT15	NE	CV, ES, SES, UAS

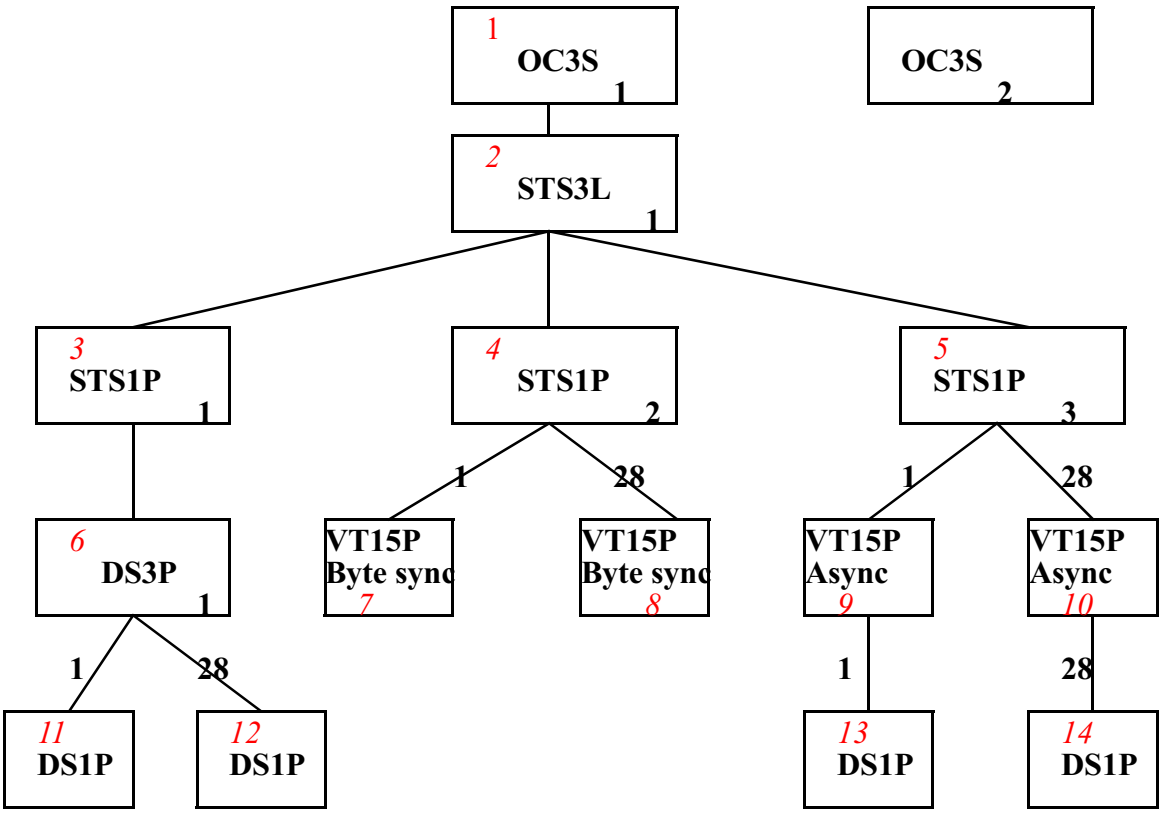
Currently, STS1P does not support FE data.

The SPM and carriers used in the tests must be datafilled.

To get current performance data, SPM and carriers must be in IDLE state.

7.1.1 TL1 Command Example Syntax

Figure 2 Hierarchy SPM Carriers



There are there types of STS1P:
STS1P-DS3P-DS1P, STS1P-VT15P, STS1P-VT15P-DS1P.
Bold numbers represent physical location and italic numbers are used TL1 examples.

7.1.2.1 Section/Path/Line Carrier type

Carrier type in the <cmd> is determined like:

OC3 = OC3S
STS3 = STS3L
STS1 = STS1P
DS3 = DS3P
VT15 = VT15P
DS1 = DS1P

in TL1 command, the carrier types needs to be entered like:
OC3 , STS3, STS1,DS3,VT15, DS1. TL1 Engine will convert them to:
OC3S, STS3L, STS1P, DS3P, VT15P, DS1P

7.1.2.2 Monitor Type

Monitor type is determined by carrier type as below:

CV = CVS if carrier type=OC3
CV= CVP if carrier type= STS1P/DS3P/VT15P/DS1P
CV= CVL if carrier type= STS3L

In TL1 commnad monitor type, CV needs to be entered as: CV. TL1 Engine will convert it to CVS or CVL depends on carrier type. Similar for other monitor types

7.0.0.0.1 Access IDentifier (AID)

The AID in TL1 command syntax that need to be entered exactly as followings: (see Please refer to “Figure 2 Hierarchy SPM Carriers” on page 26.)

for carrier number 1, OC3S:
RTRV-PM-OC3:SPM-1:OC3-1:123: :CVS,0-UP,NE,RCV,MIN;
/*<mondat> and <montm> is left blank means current query request */
/*carrier types: OC3 = OC3S
STS3 = STS3L
STS1 = STS1P
*/

For carrier number 2, STS3L:
RTRV-PM-STS3:SPM-1:OC3-1:124: :LBC,0-UP,NE,RCV,MIN;

For carrier number 3, STS1P:
RTRV-PM-STS1:SPM-1:OC3-1-1:125::LBC,0-UP,NE,RCV,MIN;

For carrier number 6, DS3P:
RTRV-PM-DS3:SPM-1:OC3-1-1-0-1:126::LBC,0-UP,NE,RCV,MIN;

For carrier number 8, VT15P:
RTRV-PM-VT15:SPM-1:OC3-1-2-28:127::LBC,0-UP,NE,RCV,MIN;

For carrier number 11, DS1P:
RTRV-PM-DS1:SPM-1:OC3-1-1-0-1-1:128::LBC,0-UP,NE,RCV,MIN;

For carrier number 14, DS1P:
RTRV-PM-DS1P:SPM-1:OC3-1-3-28-0-28:129::LBC,0-UP,NE, RCV, MIN;

7.1.2.3 TL1 Command Error Codes and Description

Table 1 TL1 Error Codes

Error Codes	Description
CMTL_EC	command too long
BEXT_EC	block extra
BMISS_EC	block missing
NNUP_EC	non-null unimplemented parameter
NCN_EC	null command name
NTID_EC	null tid is not allowed
NAID_EC	null ai is not allowed
NCTAG_E	null ctag is not allowd
ICM_EC	invalid command name
MONTYPE_EC	monitor type invalid
MONLEV_EC	monitor level invalid
LOCN_EC	location invalid
DIRN_EC	direction invalid
TMPER_EC	time period invalid

Error Codes	Description
MONDAT_EC	monitor date invalid
MONTM_EC	monitor time invalid
BL5TMF_EC	block5 has too many fields
CANC_EC	carrier type & aid not consistent
AID_LOC_EC	location in AID must be a number
AID_OC3_EC	OC3 in AID not valid
OC3_EC	OC3 or STS3 slot number invalid
AID_STS1_EC	sts1 in aid not valid
STS1_EC	sts1 number out of range
AID_VT15_EC	vt15 in aid not valid
VT15_EC	vt15 location out of range
AID_DS3_EC	ds3 in aid not valid
DS3_EC	ds3 location not valid
AID_DS1_EC	ds1 or vt15 in aid not valid
DS1_EC	ds1 or vt15 number invalid
AID_VT15_DS3_EC	vt15 location and ds3 location, one of them must be= zero
OFF_PARM_EC	office parameter is off
CNPV_EC	cicuit not provision
TID_EC	TID is invalidi, out of range, {0 to 63}"}
TMFDS_EC	too many fields
EXSMCL_EC	Extra semicolon in TL1 command inpu
MSMCL_EC	Missing semicolon at the end
TL1PMALLOC_EC	TL1 Parser Allocate Memory Error
SDM_RPC_EC	RPC in SDM failed
SWNMALLOC_EC	SWNMALLOC_EC
TL1APMALLOC_EC	TL1 Aid Parser Allocate Memory Error

Error Codes	Description
SPMINV_EC	SPM in tid is invalid

7.2 Startup the TCP/IP server on the SDM

Once logged on to the SDM change directories to reachthrough/TL1. From the TL1 directory on SDM start the TCP/IP server by typing ‘./server <CR>’. On the TCP/IP client workstation start the client by typing client IP address (client ctest1 <CR>). This will establish the TCP/IP server/client connection. After the TCP/IP connection has been established the user can enter TL1 commands at the client workstation. The command will be sent to the server on the SDM , parsed, and dumped to the terminal window where the server was started. This parsed information will indicate the success or failure of the TL1 command parser. For Tl1 command responses the server will send data in the TL1 response format to the TCP/IP client.

7.3 Examples

7.3.1 Example 1 - Get Current Data

```
At TCP/IP client window enter > “RTRV-PM-OC3:SPM-0:OC3-1:123::;”
Expected Results:
current was called

 num_msg= 1

 resp_format, cust_outMsg->msg0=

CM-SPM-0 2000-03-30 09:44:38
M  123 COMPLD
   "OC3-1:11"
   "OC3-1:LBC,0,NE,RCV,15-MIN,,"
   "OC3-1:OPT,1,NE,RCV,15-MIN,,"
   "OC3-1:OPR,2,NE,RCV,15-MIN,,"
   "OC3-1:CV,3,NE,RCV,15-MIN,,"
   "OC3-1:ES,4,NE,RCV,15-MIN,,"
   "OC3-1:SES,5,NE,RCV,15-MIN,,"
   "OC3-1:SEFS,6,NE,RCV,15-MIN,,"
   "OC3-1:UAS,7,NE,RCV,15-MIN,,"
   "OC3-1:PSC,8,NE,RCV,15-MIN,,"
   "OC3-1:AISS,9,NE,RCV,15-MIN,,"
   "OC3-1:CSS,10,NE,RCV,15-MIN,,"
```

7.3.2 Example 2 - Get Historical Data

At TCP/IP client window enter > RTRV-PM-OC3:SPM-5:OC3-1:1234::,,,,,2000-03-30;

Results at TCP/IP client:

```
historical was called

 num_msg= 5

 resp_format, cust_outMsg->msg0=

   CM-SPM-5  2000-03-30 10:00:53
M  1234 COMPLD
   "OC3-1:352"
   "OC3-1:LBC,0,NE,RCV,15-MIN,,02:25"
   "OC3-1:LBC,1,NE,RCV,15-MIN,,02:10"
   "OC3-1:LBC,2,NE,RCV,15-MIN,,01:55"
   "OC3-1:LBC,3,NE,RCV,15-MIN,,01:40"
   "OC3-1:LBC,4,NE,RCV,15-MIN,,01:25"
   "OC3-1:LBC,5,NE,RCV,15-MIN,,01:10"
   "OC3-1:LBC,6,NE,RCV,15-MIN,,00:55"
   "OC3-1:LBC,7,NE,RCV,15-MIN,,00:40"
   "OC3-1:LBC,8,NE,RCV,15-MIN,,00:25"
   "OC3-1:LBC,9,NE,RCV,15-MIN,,00:10"
   "OC3-1:LBC,10,NE,RCV,15-MIN,,23:55"
   "OC3-1:LBC,11,NE,RCV,15-MIN,,23:40"
   "OC3-1:LBC,12,NE,RCV,15-MIN,,23:25"
   "OC3-1:LBC,13,NE,RCV,15-MIN,,23:10"
   "OC3-1:LBC,14,NE,RCV,15-MIN,,22:55"
   "OC3-1:LBC,15,NE,RCV,15-MIN,,22:40"
   "OC3-1:LBC,16,NE,RCV,15-MIN,,22:25"
   "OC3-1:LBC,17,NE,RCV,15-MIN,,22:10"
   "OC3-1:LBC,18,NE,RCV,15-MIN,,21:55"
   "OC3-1:LBC,19,NE,RCV,15-MIN,,21:40"
   "OC3-1:LBC,20,NE,RCV,15-MIN,,21:25"
   "OC3-1:LBC,21,NE,RCV,15-MIN,,21:10"
   "OC3-1:LBC,22,NE,RCV,15-MIN,,20:55"
   "OC3-1:LBC,23,NE,RCV,15-MIN,,20:40"

{Data omitted...Full output not shown}

   "OC3-1:CSS,347,NE,RCV,15-MIN,,19:40"
   "OC3-1:CSS,348,NE,RCV,15-MIN,,19:25"
   "OC3-1:CSS,349,NE,RCV,15-MIN,,19:10"
   "OC3-1:CSS,350,NE,RCV,15-MIN,,18:55"
   "OC3-1:CSS,351,NE,RCV,15-MIN,,18:40"

;
```

8.0 Frequently Asked Questions

The following information is given to enable the craftsperson to better understand what needs to be done based on the results of the PMA data.

8.1 Question: I'm getting lots of high counts in my data and I am getting CARR811 logs on the core. What do I need to check?

Depending on the performance parameters that are having a problem there will be different items to check.

- CSS, UAS: Most likely a clocking issue. For Succession nodes, check the ATM Passport switch and check table MNNODE for correct clocking (it should be set to loop timing).

- CV, ES, SES, UAS: Make sure your frame format (SF, ESF) and line encoding values (AMI, ZCS, B8ZS) are set to the same values on each end of the carrier.

- AISS: The equipment you are connected to is having a problem. Check the nodes on the far end of the carrier.

- Other Items To Check: On optical cards perform the following steps: loopback transmit to receive. If everything clears up, the problem is in the previously connected fiber or it is in the node it is connected to. If problem still exists, also reseat the card by physically unlatching it and pulling it out, and then re-inserting it (perform this action one card at a time).

-Check physical connections. Loose wires (LSA DS1 Succession node), poor optical connections, or poor backplane connections can contribute to error prone conditions.

8.2 Question: I need NTPs for PM and PMA. What should I look at?

For PM at MAPCI see NTP 297-1771-819. Search for "perfmon". This describes the commands available at the carrier performance monitoring semi-realtime monitoring level: mapci;mtc;trks;carrier;post spm X;17 (0 to 4)

For PMA see NTP 297-1771-819. PMA consists of the following commands: setcarr, gethist, listhist in the carrutil directory.

8.3 Question: What are the optical parameters?

There are 3 values that are present for reading on the map for optical section carriers (OC3S carriers). OPT, the optical power transmitted by the laser, OPR, the optical power received by the receiver, and LBC, the laser bias current. Of these, two of the readings (OPT and LBC) are actually initialized during manufacturing, when the laser is placed into the board. The OPR reading cannot be, since we don't know what the card is getting plugged into. That is why this

value must be initialized once the link is in service (you have to issue the recordOpr0 command at the perfmon level for the OC3S carrier to record OPR) With OPR recorded you can measure what the other end is sending.

8.4 Question: How do I analyze the optical parameters OPT and LBC?

Lasers last for a large number of years (10-15+ years). At the end of their life cycle, noticeable changes appear in OPT and LBC. OPT and LBC will not stay around the 100% scale factor. These changes will be evident about 3 months before complete failure.

9.0 Glossary

Item	Description/definition
AISS	Alarm Indication Seconds
CSS	Controlled Slip Seconds
CV	Code Violations
DAY	24 hour interval
ES	Errored Seconds
FE	Far End. This is the office that terminates the other end of the connection.
LBC	Laser Bias Current. OC3S Carrier Only. Used to indicate Failure for the laser through degradation of the laser performance.
MIN	15 minute interval
NE	Near End. This is the office the craftsperson is currently in.
OPT	Optical Power Transmitted OC3S Carrier Only. Used to indicate failure for the laser through degradation of laser performance.
PM15	Performance monitoring 15 minute interval
PM24	Performance monitoring 24 hour interval
SEFS	Severly Errored Frame Seconds
SES	Severly Errored Seconds
UAS	Unavailable Seconds

10.0 Sample Source Code For Interpreting FTPed PMA Files

Sample C source and header files are listed below.

10.1 Sample C Source (.cpp) Files

```
////////////////////////////////////////////////
                  cME.cpp
////////////////////////////////////////////////

#include
#include
#include
#include "spmtypes.h"
#include "bitUtil.h"
#include "cSPM.h"
#include "cME.h"
#include "cPP.h"


cME::cME(){
   payld.nlevels = 0;
   dir = (t_dir)0;
   nodePtr = NULL;
   for(int i=0;iinterval==PM15?1:3);
   for(i=0;ireadPP(fd):0;

return(1);
}

int cME::print(){
   int i;

   printf("   ");
   for(i=0;i
#include
#include
#include "spmtypes.h"
#include "bitUtil.h"
#include "cPMfile.h"
#include "cSPM.h"


cPMfile::cPMfile(){
   fd = 0;
```

```
   numspm = 0;
   for(int i=0;i1){
         offset=file_header.spm_start_pos;
         for(int j=0;j1)
              offset+=file_header.blk_size*(file_header.spm_size_tbl[j]>>1);
         lseek(fd,offset,SEEK_SET);
         pSPM=new cSPM(i);
         if(pSPM->readSPM(fd))
            this->addSPM(pSPM);  // Changed from . to ->
         else delete(pSPM);
      }
   }
return(1);
}

void cPMfile::print(){

   printf("Period ending: %s\n",timestamp);
   if(numspm){
      for(int i=0;iprint();
      }
   else printf("\n\nNo PM data collected.\n");
}


////////////////////////////////////////////////
                  cPP.cpp
////////////////////////////////////////////////

#include
#include
#include
#include "spmtypes.h"
#include "cPP.h"

cPP::cPP(){
   name = (t_perf_parm)0;
   size = 0;
   count=0;
}

cPP::cPP(t_perf_parm n, int s){
   name = n;
   size = s;
   count=0;
}

void cPP::setpp(long c){
   count = c;
}
```

```
int cPP::readPP(int fd){
   unsigned long temp=0;
   int times2shift;

   if(!read(fd,&temp,size))
      return(0);
   times2shift=((sizeof(long)-size)*CHAR_BIT);
   if(times2shift>=0)
      count=temp>>times2shift;
   else count=0;
return(1);
}

int cPP::printName(){
   printf("%12s",PERF_PARM[name]);
   return 1;  // Added as compiler kludge
}

int cPP::printCount(){
   printf("%12u",count);
   return 1;  // Added as compiler kludge
}


////////////////////////////////////////////////
                 cSPM.cpp
////////////////////////////////////////////////

#include
#include
#include "spmtypes.h"
#include "bitUtil.h"
#include "cSPM.h"
#include "cME.h"
#include "cPP.h"


#define STM1R_FIRST_OFFSET  0
#define STM1R_LAST_OFFSET   1
#define STM1M_FIRST_OFFSET  2
#define STM1M_LAST_OFFSET   3
#define VC4P_FIRST_OFFSET   4
#define VC4P_LAST_OFFSET    4
#define VC12P_FIRST_OFFSET  5
#define VC12P_LAST_OFFSET  67
#define E1P_FIRST_OFFSET   68
#define E1P_LAST_OFFSET   130

cSPM::cSPM(){
   num = 0;
   formatid = 0;
```

```
   idf = 0;
   duration = 0;
   mecount=0;
   interval=(t_pap_intvl)0;
   for(int i=0;iread_old_SPM(fd);  // Changed . to ->
   if (formatid == 1) return this->read_new_SPM(fd);  // Changed . to ->

   printf ("\nERROR: Invalid Format ID: %d\n", formatid);
   return(0);

} // cSPM::readSPM

static t_carr_type CONVERT_OLD_CARRTYPE [] =
{
   DS1L,
   OC3S,
   STS3L,
   STS1P,
   DS3P_M23,
   VT15P,
   DS1P_SF
};

int cSPM::read_old_SPM(int fd)
{
   PAYLOAD     pyld;
   cME        *pME;
   cPP        *pPP;
   char        carrv[C_CARR_VECTOR_SIZE_N_CHAR];
   char        dsxv[C_DSX_VECTOR_SIZE_N_CHAR];
   char        hold;
   char        parm_size[NUM_PERF_PARMS_V0][NUM_PAP_INTVLS];
   int         i,j,k,l;
   int         carrnum=0;
   int         numcarr=0;
   int         numpp=0;
   short       carr_format[NUM_CARR_TYPES];
   t_carr_type carrt;
   t_perf_parm pm_config[NUM_CARR_TYPES] [NUM_DIRS] [NUM_PERF_PARMS_V0];
   t_perf_parm ppid;

   for(i=0;iMAX_NUM_CARR_FMT))
      return(0);
   numcarr=(int)hold;

   for(i=0;i=NUM_CARR_TYPES))
         return(0);
      carrt=(t_carr_type)hold;

      for(j=0;jMAX_PP_PER_ME))
            return(0);
```

```
      numpp=(int)hold;

      for(k=0;k=NUM_PERF_PARMS_V0))
            return(0);
         pm_config[CONVERT_OLD_CARRTYPE[carrt] + carr_format[carrt]]
                  [j][k]=(t_perf_parm)hold;
      }
   }
   carr_format[carrt]++;
}

if(!read(fd,&hold,1))
   return(0);

if(((int)hold<0)||((int)hold>=NUM_POSSIBLE_INTVLS))
   return (0);
interval=(t_pap_intvl)hold==PM24?PM15:PM24;

if(!read(fd,&duration,sizeof(duration)))
   return(0);

for(i=0;i=C_OC3_CARRV_START&&i<=C_OC3_CARRV_STOP){
         carrt=OC3S;
         carrnum=i-C_OC3_CARRV_START;
         pyld.nlevels = 1;
         pyld.payload[0].carrType=OC3S;
         pyld.payload[0].pyldPos=carrnum;
      }
      else if(i>=C_STS3_CARRV_START&&i<=C_STS3_CARRV_STOP){
         carrt=STS3L;
         carrnum=i-C_STS3_CARRV_START;
         pyld.nlevels = 1;
         pyld.payload[0].carrType=STS3L;
         pyld.payload[0].pyldPos=carrnum;
      }
      else if(i>=C_STS1_CARRV_START&&i<=C_STS1_CARRV_STOP){
         carrt=STS1P;
         carrnum=i-C_STS1_CARRV_START+1;
         pyld.nlevels = 1;
         pyld.payload[0].carrType=STS1P;
         pyld.payload[0].pyldPos=carrnum;
      }
      else if(i>=C_DS3_CARRV_START&&i<=C_DS3_CARRV_STOP){
         if ((t_format)bitTest(dsxv,C_NUM_DS1_CARR+carrnum-1))
         {
            carrt=DS3P_CBIT;
         }
         {
            carrt=DS3P_M23;
         }
         carrnum=i-C_DS3_CARRV_START+1;
```

```
            pyld.nlevels = 2;
            pyld.payload[0].carrType=STS1P;
            pyld.payload[0].pyldPos=carrnum;
            pyld.payload[1].carrType=carrt;
            pyld.payload[1].pyldPos=1;
         }
         else if(i>=C_VT15_CARRV_START&&i<=C_VT15_CARRV_STOP){
            carrt=VT15P;
            carrnum=i-C_VT15_CARRV_START+1;
            pyld.nlevels = 2;
            pyld.payload[0].carrType=STS1P;
            pyld.payload[0].pyldPos=((carrnum-1)/28)+1;
            pyld.payload[1].carrType=VT15P;
            pyld.payload[1].pyldPos=((carrnum-1)%28)+1;
         }
         else if(i>=C_DS1_CARRV_START&&i<=C_DS1_CARRV_STOP){
            if ((t_format)bitTest(dsxv,C_NUM_DS1_CARR+carrnum-1))
            {
               carrt=DS1P_ESF;
            }
            {
               carrt=DS1P_SF;
            }
            carrnum=i-C_DS1_CARRV_START+1;
            pyld.nlevels = 3;
            pyld.payload[0].carrType=STS1P;
            pyld.payload[0].pyldPos=((carrnum-1)/28)+1;
            if(bitTest(carrv,(carrnum-1)/28+C_DS3_CARRV_START)){
               pyld.payload[1].carrType=DS3P_M23;
               pyld.payload[1].pyldPos=1;
               pyld.payload[2].carrType=carrt;
               pyld.payload[2].pyldPos=((carrnum-1)%28)+1;
            }
            else{
               pyld.payload[1].carrType=VT15P;
               pyld.payload[1].pyldPos=((carrnum-1)%28)+1;
               pyld.payload[2].carrType=carrt;
               pyld.payload[2].pyldPos=1;
            }
         }
         for(j=0;jaddPP(pPP,ppid);
               }
               if(pME->readME(fd))
                  this->addME(pME);  // Changed . to ->
               else delete(pME);
            }
      }
   }

return (1);
}
```

```
int cSPM::read_new_SPM(int fd)
{
   PAYLOAD pyld;
   cME *pME;
   cPP *pPP;
   char parm_size[NUM_PERF_PARMS][NUM_PAP_INTVLS];

   char hold;
   int  i,j,k,l;
   int   carrnum=0;
   int   numcarr=0;
   int   numpp=0;
   t_carr_type carrt;
   t_perf_parm pm_config[NUM_CARR_TYPES][NUM_DIRS][NUM_PERF_PARMS];

   for(i=0;i= NIL_PPID)
      {
        printf ("\nERROR: cSPM::read_new_SPM PPID(%d) out of range\n", ppid);
         return(0);
      }

      for (l_interval = 0; l_interval < NUM_PAP_INTVLS; l_interval++)
      {
         if(!read(fd, &interval_size, sizeof(interval_size)))
         {
            printf ("\nERROR: cSPM::read_new_SPM reading interval size\n");
            return(0);
         }
         if ((interval_size!=1) && (interval_size!=2) && (interval_size!=4))
         {
            printf ("\nERROR: cSPM::read_new_SPM interval size(%d) is
invalid\n", interval_size);
            return(0);
         }
         parm_size[ppid][l_interval] = interval_size;
//         printf ("  %4d", interval_size);
      }
//      printf ("\n");
   }

   //-------------------------------------------------------------------------
   // Carrier Config Block
   //-------------------------------------------------------------------------
   int   number_of_carriers;
   char  type_of_carrier   [MAX_NUM_CARRIERS+1];
   int   carriers_position [MAX_NUM_CARRIERS+1];

   if(!read(fd, &hold, 1))
   {
      printf ("\nERROR: cSPM::read_new_SPM reading # of carriers\n");
```

```
      return(0);
   }
   number_of_carriers = (int)hold &0x00FF;
//printf ("INFO: number_of_carriers = %d\n", number_of_carriers);

   if (number_of_carriers > MAX_NUM_CARRIERS)
   {
      printf ("\nERROR: cSPM::read_new_SPM # of carriers(%d)\n",
number_of_carriers);
      return(0);
   }

   for (i = 0; i < number_of_carriers; i++)
   {
      if(!read(fd, &type_of_carrier[i], sizeof(char)))
      {
         printf ("\nERROR: cSPM::read_new_SPM reading carrier type\n");
         return(0);
      }
//printf ("INFO: i = %3d  type_of_carrier = %2d", i, type_of_carrier[i]);
      if(!read(fd, &hold, sizeof(char)))
      {
         printf ("\nERROR: cSPM::read_new_SPM reading position\n");
         return(0);
      }
   carriers_position[i] = (int)hold &0x00FF;
//printf ("  carriers_position = %3d\n", carriers_position[i]);
   }

   //-------------------------------------------------------------------------
----
   // Parameter Lists
   //-------------------------------------------------------------------------
----
   char  number_of_carrier_types;
   char  carrier_type;
   char  number_of_directions;
   char  direction;
   char  pp_vector_byte;
   int   curr_pp;

   if(!read(fd, &number_of_carrier_types, sizeof(number_of_carrier_types)))
   {
      printf ("\nERROR: cSPM::read_new_SPM reading # of carrier types\n");
      return(0);
   }
   if (number_of_carrier_types > NUM_CARR_TYPES)
   {
      printf ("\nERROR: cSPM::read_new_SPM # of carrier types(%d)\n",
number_of_carrier_types);
      return(0);
```

```
   }
//printf ("INFO: number_of_carrier_types = %d\n", number_of_carrier_types);

   for (i = 0; i < number_of_carrier_types; i++)
   {
      if(!read(fd, &carrier_type, sizeof(carrier_type)))
      {
         printf ("\nERROR: cSPM::read_new_SPM reading carrier types\n");
         return(0);
      }
//printf ("INFO: i = %d  carrier_type = %d\n", i, carrier_type);
      if (carrier_type >= NUM_CARR_TYPES)
      {
         printf ("\nERROR: cSPM::read_new_SPM carrier type(%d)\n",
carrier_type);
         return(0);
      }

      if(!read(fd, &number_of_directions, sizeof(number_of_directions)))
      {
       printf ("\nERROR: cSPM::read_new_SPM reading number_of_directions\n");
         return(0);
      }
      if (number_of_directions > NUM_DIRS)
      {
         printf ("\nERROR: cSPM::read_new_SPM number_of_directions(%d)\n",
number_of_directions);
         return(0);
      }

      for (j = 0; j < number_of_directions; j++)
      {
         if(!read(fd, &direction, sizeof(direction)))
         {
            printf ("\nERROR: cSPM::read_new_SPM reading direction\n");
            return(0);
         }
         if (direction >= NUM_DIRS)
         {
            printf ("\nERROR: cSPM::read_new_SPM carrier type(%d)
direction(%d)\n",
                     carrier_type, direction);
            return(0);
         }

         curr_pp = 0;

         for (k = 0; k < NUM_PERF_PARMS; k+=8)
         {
            if(!read(fd, &pp_vector_byte, sizeof(pp_vector_byte)))
            {
```

```
                printf ("\nERROR: cSPM::read_new_SPM reading pp_vector_byte
(%d)\n", (k/8));
                return(0);
             }

             for (l = 0; l < 8; l++)
             {
                if (pp_vector_byte & (1 << l))
                {
                   (int)ppid = k + l;  // Changed to satisfy new compiler
                   if (ppid >= NIL_PPID)
                   {
                    printf ("\nERROR: cSPM::read_new_SPM pp_vector_byte (ppid
= %d)\n", ppid);
                      return(0);
                   }

                   pm_config[carrier_type][direction][curr_pp++] = ppid;
                }
             }
          }
       }
    }

    //------------------------------------------------------------------------
    //------------------------------------------------------------------------
    //------------------------------------------------------------------------
    if(!read(fd,&hold,1))
       return(0);

    if(((int)hold<0)||((int)hold>=NUM_POSSIBLE_INTVLS))
       return (0);
    interval=(t_pap_intvl)hold==PM24?PM15:PM24;

    if(!read(fd,&duration,sizeof(duration)))
       return(0);

    //------------------------------------------------------------------------
    //------------------------------------------------------------------------
    //------------------------------------------------------------------------
    int dir;
    for (i = 0; i < number_of_carriers; i++)
    {
       switch (type_of_carrier[i])
       {
       case STM1R:
            pyld.nlevels = 1;
            pyld.payload[0].carrType=STM1R;
            pyld.payload[0].pyldPos =carriers_position[i];
            break;
```

```
      case STM1M:
           pyld.nlevels = 2;
           pyld.payload[0].carrType=STM1R;
           pyld.payload[0].pyldPos =carriers_position[i] - 2;
           pyld.payload[1].carrType=STM1M;
           pyld.payload[1].pyldPos =0;
           break;

      case VC4P:
           pyld.nlevels = 1;
           pyld.payload[0].carrType=VC4P;
           pyld.payload[0].pyldPos =1;
           break;

      case VC12P:
           pyld.nlevels = 2;
           pyld.payload[0].carrType=VC4P;
           pyld.payload[0].pyldPos =1;
           pyld.payload[1].carrType=VC12P;
           pyld.payload[1].pyldPos =carriers_position[i] - 4;
           break;

      case E1P_CRC:
      case E1P_NO_CRC:
           pyld.nlevels = 3;
           pyld.payload[0].carrType=VC4P;
           pyld.payload[0].pyldPos =1;
           pyld.payload[1].carrType=VC12P;
           pyld.payload[1].pyldPos =carriers_position[i] - 67;
          (int)(pyld.payload[2].carrType)=type_of_carrier[i];  // Changed to
satisfy new compiler
           pyld.payload[2].pyldPos =1;
           break;

      default:
           pyld.nlevels = 1;
          (int)(pyld.payload[0].carrType)=type_of_carrier[i];  // Changed to
satisfy new compiler
           pyld.payload[0].pyldPos=carriers_position[i];
           break;
      }

      for (dir = 0; dir < NUM_DIRS; dir++)
      {
         if (pm_config[type_of_carrier[i]][dir][0] != NUM_PERF_PARMS)
         {
            pME = new cME (pyld, (t_dir) dir, this);

            curr_pp = 0;
            while (pm_config[type_of_carrier[i]][dir][curr_pp] !=
NUM_PERF_PARMS)
```

```
            {
               ppid = pm_config[type_of_carrier[i]][dir][curr_pp++];
               pPP  = new cPP(ppid, parm_size[ppid][interval]);
               pME->addPP(pPP, ppid);
            }

            if (pME->readME(fd)) this->addME(pME);  // Changed . to ->
            else delete(pME);
         }
      }
   }

   return (1);
} // cSPM::read_new_SPM

void cSPM::print()
{
   printf("\nNumber of Carriers: %d", mecount);

printf("\n=============================================================================
======================");
   if(idf) printf("\n* ");
   else printf("\n  ");
   printf("SPM %2dInterval type: %s",num, PAP_INTVL[interval]);
   printf("Duration: %d secs\n",duration);
   for(int i=0; i < mecount; i++)
      melist[i]?melist[i]->print():0;
}


////////////////////////////////////////////////
                 main.cpp
////////////////////////////////////////////////

#include
#include
#include
#include "cPMfile.h"

int main(int argc, char *argv[])
{
int fd;
cPMfile *pFile;

   if(argc>1) fd=open(argv[1], O_RDONLY);
   else fd=STDIN_FILENO;
   if(fd>=0){
      pFile=new cPMfile(fd);
      pFile->readFile();
      pFile->print();
   }
```

```
   else perror(argv[1]);
   close(fd);
return (EXIT_SUCCESS);
}
```

10.2 Sample Header (.h) files

```
////////////////////////////////////////////////
                bitUtil.h
////////////////////////////////////////////////

int bitTest(char *c, int index);


////////////////////////////////////////////////
                cME.h
////////////////////////////////////////////////

#ifndef _cME_H
#define _cME_H

class cSPM;
class cPP;

class cME{
protected:
   PAYLOAD       payld;
   char          idf;
   t_dir         dir;
   cPP           *pplist[NUM_PERF_PARMS];
   cSPM          *nodePtr;

public:
cME();
cME(PAYLOAD p, t_dir d, cSPM *n);
void addPP(cPP *c, t_perf_parm pos);
int readME(int fd);
int print();
};

#endif


////////////////////////////////////////////////
                cPMfile.h
```

```
////////////////////////////////////////////////

#ifndef _cPMfile_H
#define _cPMfile_H

#include
#include "spmtypes.h"
class cSPM;

#define MAX_SPMS 64

class cPMfile{
protected:
   int            fd,
                  numspm;
   char           timestamp[SIZE_CHARTOD];
   cSPM           *spmlist[MAX_SPMS];

public:
cPMfile();
cPMfile(int f);
void addSPM(cSPM *c);
int readFile();
void print();
};

#endif


////////////////////////////////////////////////
                  cPP.h
////////////////////////////////////////////////

#ifndef _cPP_H
#define _cPP_H

class cPP{
protected:
   t_perf_parm   name;
   int           size;
   unsigned long count;

public:
cPP();
cPP(t_perf_parm n, int s);
void setpp(long c);
int readPP(int fd);
int printName();
int printCount();
};
```

```
#endif


////////////////////////////////////////////////
                cSPM.h
////////////////////////////////////////////////

#ifndef _cSPM_H
#define _cSPM_H

#include "spmtypes.h"
class cME;

#define MAX_MES 200

class cSPM{
friend cME;
protected:
   int           num,
                 mecount;
   unsigned long duration;
   short         formatid,
                 idf;
   t_pap_intvl   interval;
   cME           *melist[MAX_MES];

public:
cSPM();
cSPM(int n);
void addME(cME *m);
int readSPM(int fd);
int read_old_SPM(int fd);
int read_new_SPM(int fd);
void print();
};

#endif


////////////////////////////////////////////////
                spmtypes.h
////////////////////////////////////////////////

#ifndef _SPMTYPES_H
#define _SPMTYPES_H

#include

#define PM_FILE_FRMT_ID  (0)

#define MAX_SPM_PER_FILE (64)
```

```
#define MAX_PP_PER_ME    (8)
#define MAX_NUM_CARR_FMT (8)
#define MAX_NUM_CARRIERS (181)

#define SIZE_CHARTOD     (28)
#define SIZE_INTTOD      (6)

#define C_NUM_OC3_CARR   (2)
#define C_NUM_STS3_CARR  (2)
#define C_NUM_STS1_CARR  (3)
#define C_NUM_DS3_CARR   (3)
#define C_NUM_VT15_CARR  (84)
#define C_NUM_DS1_CARR   (84)

#define C_NUM_CARR_IN_VECTOR      (C_NUM_OC3_CARR + C_NUM_STS3_CARR +
C_NUM_STS1_CARR + C_NUM_DS3_CARR +
C_NUM_VT15_CARR + C_NUM_DS1_CARR)
#define C_CARR_VECTOR_SIZE_N_CHAR ((C_NUM_CARR_IN_VECTOR / CHAR_BIT) + 1)

#define C_OC3_CARRV_START   (0)
#define C_OC3_CARRV_STOP    (C_OC3_CARRV_START+(C_NUM_OC3_CARR-1))
#define C_STS3_CARRV_START  (C_OC3_CARRV_STOP+1)
#define C_STS3_CARRV_STOP   (C_STS3_CARRV_START+(C_NUM_STS3_CARR-1))
#define C_STS1_CARRV_START  (C_STS3_CARRV_STOP+1)
#define C_STS1_CARRV_STOP   (C_STS1_CARRV_START+(C_NUM_STS1_CARR-1))
#define C_DS3_CARRV_START   (C_STS1_CARRV_STOP+1)
#define C_DS3_CARRV_STOP    (C_DS3_CARRV_START+(C_NUM_DS3_CARR-1))
#define C_VT15_CARRV_START  (C_DS3_CARRV_STOP+1)
#define C_VT15_CARRV_STOP   (C_VT15_CARRV_START+(C_NUM_VT15_CARR-1))
#define C_DS1_CARRV_START   (C_VT15_CARRV_STOP+1)
#define C_DS1_CARRV_STOP    (C_DS1_CARRV_START+(C_NUM_DS1_CARR-1))

#define C_DSX_NUM_CARR_IN_VECTOR  (C_NUM_DS1_CARR + C_NUM_DS3_CARR)
#define C_DSX_VECTOR_SIZE_N_CHAR  ((C_DSX_NUM_CARR_IN_VECTOR / CHAR_BIT) + 1)

#define C_DS1_DSXV_START   (0)
#define C_DS1_DSXV_STOP    (C_DS1_DSXV_START+(C_NUM_DS1_CARR-1))
#define C_DS3_DSXV_START   (C_DS1_DSXV_STOP+1)
#define C_DS3_DSXV_STOP    (C_DS3_DSXV_START+(C_NUM_DS3_CARR-1))

enum t_carr_type{
   DS1L,
   OC3S,
   STS3L,
   STS3CP,
   STS1P,
   DS3P_M23,
   DS3P_CBIT,
   VT15P,
   DS1P_SF,
   DS1P_ESF,
```

```
   STM1R,
   STM1M,
   VC4P,
   VC12P,
   E1P_CRC,
   E1P_NO_CRC
};
#define NUM_CARR_TYPES (int)E1P_NO_CRC +1

static char *CARR_TYPE[] = {
   "DS1L",
   "OC3S",
   "STS3L",
   "STS3CP",
   "STS1P",
   "DS3P",
   "DS3P",
   "VT15P",
   "DS1P",
   "DS1P",
   "STM1R",
   "STM1M",
   "VC4P",
   "VC12P",
   "E1P",
   "E1P",
""
};

enum t_perf_parm{
   LBC,          /* 00 laser bias current               */
   OPT,          /* 01 optical power transmitter        */
   OPR,          /* 02 optical power received           */
   CV,           /* 03 coding violations                */
   ES,           /* 04 errored seconds                  */
   SES,          /* 05 severely errored seconds         */
   SEFS,         /* 06 severely errored framing seconds */
   UAS,          /* 07 unavailable seconds              */
   PSC,          /* 08 protection switch count          */
   AISS,         /* 09 alarm indication signal seconds  */
   CSS,          /* 0A controlled slip seconds          */
   BBE,          /* 0B background block errors          */
   CRC4,         /* 0C crc4 violations                  */
   NIL_PPID
};
#define NUM_PERF_PARMS (int) NIL_PPID
#define NUM_PERF_PARMS_V0 (int) BBE

#define SIZE_OF_PPVECTOR ((NUM_PERF_PARMS+7)/8);

static char *PERF_PARM[] = {
```

```
   "LBC",
   "OPT",
   "OPR",
   "CV",
   "ES",
   "SES",
   "SEFS",
   "UAS",
   "PSC",
   "AISS",
   "CSS",
   "BBE",
   "CRC4",
""
};

#define NUM_FORMATS (2)
enum t_format{
   F_ONE,
   F_TWO
};

#define NUM_DIRS (2)
enum t_dir{
   NE,
   FE
};

static char *DIR[] = {
   "NE",
   "FE",
""
};

#define NUM_PAP_INTVLS      (2)
#define NUM_POSSIBLE_INTVLS (4)
enum t_pap_intvl{
   PM15,      /* previous 15-min            */
   PM24       /* previous 24-hour (daily)   */
};

static char *PAP_INTVL[] = {
   "15 Minute",      /* previous 15-min            */
   "24 Hour",        /* previous 24-hour (daily)   */
""
};

typedef struct{
   short hour,
         min,
         sec,
```

```
         frac;
} CLOCK_TIME;

typedef struct{
   short year,
         dayinyear;
} YEAR_DAY;

typedef struct{
   YEAR_DAY    yr_day;
   CLOCK_TIME  ct;
} FULLTOD;

union INTTOD{
   FULLTOD     ltime;
   short       tod[6];
};

#define PM_FILE_HEADER_SIZE_N_CHAR
((2*3)+SIZE_CHARTOD+(2*6)+(2*1)+MAX_SPM_PER_FILE)
typedef struct{
   short       spm_start_pos,
               blk_size,
               pm_file_version;
   char        timestamp_str[SIZE_CHARTOD];
   INTTOD      timestamp;
   short       spm_num_w_pmdata;
   char        spm_size_tbl[MAX_SPM_PER_FILE];
} PM_FILE_HEADER;

typedef struct{
  t_carr_type     carrType;
  short           pyldPos;
} PAYLOAD_ENTRY;

typedef struct{
   short   nlevels;
   PAYLOAD_ENTRY   payload[3];
} PAYLOAD;

#endif


 Makefile

CC            = g++

DEST          = .

HDRS          = bitUtil.h \
                cME.h \
```

```
                cPMfile.h \
                cPP.h \
                cSPM.h \
                spmtypes.h

LD            = g++

MAKEFILE      = Makefile

OBJS          = bitUtil.o \
                cME.o \
                cPMfile.o \
                cPP.o \
                cSPM.o \
                main.o

PROGRAM       = PMparse

SRCS          = bitUtil.c \
                cME.cpp \
                cPMfile.cpp \
                cPP.cpp \
                cSPM.cpp \
                main.cpp

all:            $(PROGRAM)

$(PROGRAM):     $(OBJS)
                @echo "Linking $(PROGRAM) ..."
                @$(LD) $(OBJS) -o $(PROGRAM)
                @echo "done"

$(OBJS):     $(SRCS)
                @echo "Compiling $(PROGRAM) ..."
                @$(CC) -c $(SRCS)
                @echo "done"

clean:;         @rm -f $(OBJS) core

depend:;        @mkmf -f $(MAKEFILE)

###
cME.o: spmtypes.h bitUtil.h cSPM.h cME.h cPP.h
cPMfile.o: spmtypes.h bitUtil.h cPMfile.h cSPM.h
cPP.o: spmtypes.h cPP.h
cSPM.o: spmtypes.h bitUtil.h cSPM.h cME.h cPP.h
main.o: cPMfile.h spmtypes.h
```