24.1 INTRODUCTION

SYSLOD is a system software module which is linked to the TOSS Monitor. It takes care of application loading and monitor and application configuration on the whole range of PTS systems; PTS6805 and PTS6810 with or without overlay, and PTS6820 with MMU and/or overlay.

SYSLOD performs five functions:-

* Loading the application load module.
* Reading the configuration file.
* Monitor configuration.
* Application configuration.
* Starting the System.

24.2 I/O REQUIREMENTS

SYSLOD has its own I/O routines and is independent of drivers contained in the Monitor. Loading can be done from fixed, cartridge or flexible disk, or from cassette. Monitor, application load module, and configuration file must all be loaded from the same input medium type, but need not be on the same volume.

24.3 LOADING PROCEDURE

The System loading program consists of three modules. When the Monitor has been loaded by the initial program loader, control is passed to SYSLOD. SYSLOD loads the application and reads the configuration file, SYSLDM then performs monitor configuration and SYSLDA performs application configuration. SYSLDA will queue all the tasks in the dispatcher queue. After that, all the drivers are initialized and control is passed to the DEBUGGER or the Interpreter.

24.4 APPLICATION LOADING

First the application load module as created by the Linkage Editor (LKE) is loaded into core. If loading is from disk and the program is segmented, only segment zero and the other core resident segments are loaded. If loading is from a sequential access medium or if the program is not segmented the entire application is loaded.

The application, segment 0, and the memory-resident segments are placed as high as possible in memory.

24.4.1 Memory Layout after Application Loading

```
 -------------------------
| MONITOR                 |
|-------------------------|
| SYSLOD                  |
| SYSLDA                  |
| SYSLDM                  |
|-------------------------|
| DWT prototypes          |    generated by SYSGEN
|-------------------------|
|                         |
|                         |
| Free area               |
|                         |
|                         |
|-------------------------|
| S:GTAB                  |
|-------------------------|
| DDIV prototype          |
|-------------------------|_
| P:MTAB                  | |
|-------------------------| |
| P:PIL                   | |
|-------------------------| |  Segment 00
| ASSEMBLER subroutines   | |
|-------------------------| |
| INTERPRETER             |_|
|-------------------------|
| P:S001                  |    Segment 1 _
|-------------------------|               |
| P:S002                  |    Segment 2  |
|-------------------------|               |
~                         ~               |  core resident
                                          |  segments
~                         ~               |
|                         |               |
|-------------------------|               |
|_P:S0nn__________________|    Segment n _|
```

24.5 READING THE CONFIGURATION FILE

Now the configuration file is read into memory and, in a 64Kb machine, placed immediately before the application. With a 256Kb memory, it is placed at address X'FFFE' and lower.

The part of the SYSLOD program which is still needed, and the DWT prototypes, are then placed immediately before the configuration data to make room for monitor tables and buffers.

24.5.1 Resulting Memory Layout

```
 ________________________
| MONITOR                |
|------------------------|
| area for MONITOR       |
| tables and buffers     |
|------------------------|
|                        |
|                        |
|------------------------|
| SYSLDA                 |
| SYSLDM                 |
|------------------------|
| DWT prototypes         |
|------------------------|
| Configuration data     |
|------------------------| 64Kb
|                        |
|                        |
|------------------------|
| S:GTAB                 |
|------------------------|
| DDIV prototype         |
|------------------------|_
| P:MTAB                 | |
|------------------------| |
| P:PIL                  | |
|------------------------| |  Segment 00
| ASSEMBLER subroutines  | |
|------------------------| |
| INTERPRETER            |_|
|------------------------|
| P:S001                 |    Segment 1 _
|------------------------|               |
| P:S002                 |    Segment 2  |
|------------------------|               |
~                        ~               |  memory-resident
                                         |  segments
~                        ~               |
|                        |               |
|------------------------|               |
|_P:S0nn_________________|    Segment n _|
                          256Kb
```

24.6 MONITOR CONFIGURATION

SYSLDM now performs Monitor configuration. Input data for Monitor configuration is the configuration file, the DWT prototypes generated by SYSGEN, and the data arrays SCLASS and TCLASS with information on the special device classes and terminal device classes as specified during the SYSGEN dialogue.
The tables MONTAB and SYSTAB hold the addresses of the run-time tables (configuration tables).

24.6.1 Building Monitor Tables

The following Monitor tables are built:-

* Task Control Table, TCTAB, with pointers to all the TTABs.
* Task Tables, TTAB, with specific information for every task.
* Device Work Tables, DWT, for the terminal devices, special devices and common devices.
* Interrupt Tables for the terminal devices according to the line connections specified in the configuration data.

For a system with a Memory Management Unit (MMU) the TTABs are extended with 16 words to contain the logical MMU addresses connected with this task.
For a segmented application the segment table SEGTAB is built, holding status information, disk sector address, length and load address in memory of each segment.
The corresponding page table PAGTAB is generated which contains the page queue pointers, physical page addresses and the segment block address if the page is used.

24.6.2 Workblocks

If swappable workblocks are defined in the configuration file, each SWB is described in a block SWBBLK which contains information about number of copies of the block, disk address, number of sectors occupied by each copy, and the block length in bytes. Table SWBTAB with pointers to all SWBBLK's is also built.

24.6.3 Buffers

Buffer areas are reserved for data communications, data management, and other devices or functions as required. In a system with MMU, extra I/O buffers in the System area are allocated. These areas are still used by SYSLDM and SYSLDA during configuration. SYSLDM ends by generating the Monitor blocks.

S:GTAB and the DWT prototypes are now no longer needed and may be overwritten. The area occupied by SYSLDM is then released and can be used for application configuration.

24.7 APPLICATION CONFIGURATION

Application configuration is performed by SYSLDA. An auxiliary table with the number of tasks per taskclass is formed to build the terminal control area table T:ATAB. For user workblocks and swappable workblocks, tables are set up according to the number of blocks specified in the configuration data.
If the application contains disk resident segments, memory pages are reserved as read only areas to contain these segments.

24.7.1 Generating the Data Division

SYSLDA and the data division prototype are relocated to make room for the real data division, which can now be built from the DDIV prototype and the configuration data. Workblocks with their descriptor blocks are generated.

Pointers to T:ATAB, U:BTAB, and S:BTAB are updated in P:MTAB. For a system with MMU, task-connected MMU addresses are filled in in TTAB. If the application contains disk resident segments, memory pages are reserved as read only areas to contain these segments.

Application data can be divided into three types and building of the DDIV is accordingly done in three steps:-

1. The part common to all tasks is generated, the CWB's and UWB's that are used by more than one taskclass.

2. The task class data, CWB's, UWB's, and SWB's that are used within a taskclass.

3. The task local data is generated, the Task Control Area, TCA, the terminal stack, dataset buffers, and TWB's and SWB's.

24.7.2 Generating the Tasks

One task of each taskclass is now completely generated.
Task control areas T:A are copied as many times as indicated in the auxiliary T:ATAB table.
The task identifiers TID are updated in the T:A's and saved in the T:AID table.
Now every task is put in the dispatcher queue and SYSLDA gives control to the module PFINIT to initialize the drivers. A branch to the dispatcher is then performed to schedule the first task.

24.8 ALLOCATION RULES

In a system without MMU, configuration simply consists of making as many copies of each type of task and workblock as specified in the configuration data. If the application is also not segmented, application buffers are allocated following the monitor tables and buffers upwards, over the area still used by SYSLDA.

For a segmented application or a system with MMU, application buffers are allocated from the highest free address in the free area downwards. This leaves space for memory pages for the disk resident segments between the application buffers and the monitor tables and buffers.

24.8.1 Task Window

For a system with MMU, configuration is fairly complicated. The MMU table with 16 entries, each needed to address 4K bytes, provides the task with a logical task window of 64K bytes. Segment zero, with the common part of the data division and the task-class common data, P:MTAB, P:PIL, the assembler subroutines and the interpreter, must be a part of all task windows.

This occupies the MMU table to a high degree already. The number of entries necessary to address one page in core must also be reserved. The rest of the MMU table entries may be used for the part of the data division used by the task.

24.8.2 Resulting Logical Task Window

```
                                                           __________
                                                          | SCT      |
                                                          | pointers |
                                                          |__________|

 ________________________
|                        |         TWBs with related D:T's
|                        |         SWBs with related D:T's
|   Task data            |         T:A with related control
|                        |         information, data set
|                        |         buffers and terminal stack.
|------------------------|
|                        |         CWB's and UWB's with
|   Task class data      |         related D:T's, T:D,
|                        |         TWB descriptor tables  ________
|------------------------| <---------------------------| SCTMMC |
|                        |         CWB's and UWB's with |________|
|   Common data          |         related D:T's,
|                        |         T:ATAB, U:BTAB, S:BTAB ________
|------------------------| <---------------------------| SCTLAC |
|   P:MTAB               |         Program table        |________|
|------------------------|
|   P:PIL                |         common code part,
|                        |         common pool
|------------------------|
|   ASS                  |         Assembler subroutines
|------------------------|
|   INT                  |         Interpreter
|                        |                                ________
|------------------------| <---------------------------| SCTMMP |
|                        |         one page of          |________|
|   One code page        |         disk- or memory-resident
|________________________|         coding
```

24.9 MEMORY LAYOUT AT END OF APPLICATION CONFIGURATION

```
 ------------------------
| MONITOR                |
|------------------------|
| MONITOR                |
| tables and buffers     |
|------------------------|
|                        |
| Free Area              |
|                        |
|------------------------|
| Page 1                 |
|------------------------|
|                        |
~                        ~    Pages reserved for
~                        ~    disk-resident segments
|                        |
|------------------------|
| Page n                 |
|------------------------|
| DDIV                   |
|------------------------|_
| P:MTAB                 | |
|------------------------| |
| P:PIL                  | |
|------------------------| |  Segment 00
| ASSEMBLER subroutines  | |
|------------------------| |
| INTERPRETER            | |
|------------------------|_|
| P:S001                 |    Segment 1 _
|------------------------|               |
| P:S002                 |    Segment 2  |
|------------------------|               |
~                        ~               |  memory-resident
~                        ~               |  segments
|                        |               |
|------------------------|               |
|_P:S0nn_________________|    Segment n _|
```

24.10 CREDIT APPLICATION IN SECONDARY MEMORY

The Application Load Module created by LKE.

```
 ________________________
| S:GTAB                 |     Segment table
|------------------------|
| DDIV prototype         |
|------------------------|
| P:MTAB                 |     Program table
|------------------------|_
| P:PIL                  | |   Common code part
|                        | |   and common pool
|------------------------| |
| Assembler subroutines  | |
|------------------------| |
| INTERPRETER            | |
|------------------------|_|
| P:S001                 |     Segment 1
|------------------------|
| P:S002                 |     Segment 2
|------------------------|
|                        |
~                        ~

~                        ~
|                        |
|------------------------|
|_P:S00n_________________|     Segment n
```

24.10.1 Segment Table (S:GTAB)

S:GTAB is the segment table used by SYSLDM and afterwards overwritten.

Each item is two bytes in length.

```
             ________________________
            | P:MTAB                 |    Pointer to P:MTAB
            |------------------------|
            | PRGTYP                 |    Program type, CR=CREDIT
            |------------------------|                  AS=Assembler
            | Reserved               |
            |------------------------|
            | Reserved               |
            |------------------------|
            | PAGLG                  |    Page length(in bytes)
            |------------------------|
            | NUMSEG                 |    Number of segments
            |------------------------|
one        _| SEGTYP                 |    Segment type; R=core resident
block     |¯|------------------------|                  D=disk resident
for       | | ADDRESS                |    Logical record number
each      | |------------------------|
segment   |_| SEGLG                  |    Length in bytes
            |------------------------|
            |                        |
            ~                        ~

            ~                        ~
            |                        |
            |------------------------|
            |________________________|
```

24.10.2 Program Table (P:MTAB)

P:MTAB is the program table used by the interpreter. It contains the logical addresses of the tables that are set up for the application program. Each item is two bytes in length.

Byte		Pointers to:-
0	T:ATAB	Task control area table
2	U:BTAB	User work block control table
4	I:NTPA	System start address (in interpreter)
6	P:BAS	Start of base module (P:PIL)
8	T:BAT	Branch address table
/A	T:CAT	Call address table
/C	T:PAT	Perform address table
/E	T:LIT	Literal pool table
/10		Highest index +1 in T:LIT
/12	T:KEY	Keytable pool table
/14		Highest index +1 in T:KEY
/16	T:PIC	Picture pool table
/18		Highest index +1 in T:PIC
/1A	T:FMT	Format pool table
/1C		Highest index +1 in T:FMT
/1E	P:END	End of base module (P:PIL)
/20	T:AID	Task ID table (for CREDIT debugger)
/22	OPTION	SYSTEM option (SCOPT)
/24	LITADR	Literal addressing mode
/26	ADRMOD	Data addressing mode
/28	S:BTAB	Swappable workblock control table
/2A	5 words reserved for Assembler Debugger.	

T:ATAB Task control area table (contains logical addrɩsses).

Layout of table T:ATAB.

```
-2 | Length    |
   |___________|
 0 |           |
   |___________|
+2 |           |
   |___________|
+4 |           |
   ~           ~
```

Length is the table length in bytes, including this word. The rest of the table contains pointers to task control areas, ie. T:Axxy's.

U:BTAB User workblock control table. Contains pointer to U:BTAB.

I:NTPA System start address.
Contains start address, from the interpreter.

P:BAS Start of base module.
Start address of the interpretive code module P:PIL.

T:BAT Pointer to branch address table.

T:CAT Pointer to call table.

T:PAT Pointer to perform table.

T:LIT Literal pool table.
This word contains a pointer to another pointer pair, in segment zero, which consists of one pointer to the data descriptor table for literal constants, the second pointer points to the actual pool.
The descriptor table consists of two words for each literal constant in the pool.
The layout of the descriptor table is the same as for data item (see D:zz0).

Highest index in T:LIT.
A value which indicates the highest index in the descriptor table of segment zero.

T:KEY Keytable pool table.
This word contains a pointer to another pointer pair in segment zero, which consists of one pointer to the data descriptor table for keytables, the second pointer points to the actual pool.
The descriptor table consists of two words for every keytable in the pool.

Highest index in T:KEY.
A value which indicates the highest index in the descriptor table of segment zero.

T:PIC This word contains a pointer to another pointer pair in segment zero, which consists of one pointer to the data descriptor table for the pictures, the second pointer points to the actual pool.
The descriptor table consists of two words for each picture definition. The layout of the descriptor table is the same as for data items (see D.zz0).

Highest index in T:PIC.
A value which indicates the highest index in the descriptor table of segment zero.

T:FMT Format pool table.
This word contains a pointer to another pointer pair in segment zero, which consists of one pointer to the data descriptor table for formats, the second pointer points to the actual pool. The descriptor table consists of two words for each format list definition. The layout of the descriptor table is the same as for data items (sww D:zz0).

Highest index in T:FMT.
A value which indicates the highest index in the descriptor table of segment zero.

P:END End of base module.
End address of the interpretive code module P:PIL.

T:AID Task identification table.
This word contains a pointer to a table in which all task identifiers are stored; only used by the CREDIT debugger.

Layout of T:AID.

```
     ------------
-2 | Length     |
   |____________|
 0 | Taskid-1   |
   |____________|
+2 | Taskid-2   |
   |____________|
   |            |
   ~            ~
```

Length is the table length in bytes including this word.
Following words contain the id's of the user tasks.

OPTION System Options.
A value indicating the system options.
0 - Standard (No MMU, no disk paging).
1 - MMU system.
2 - Disk paging system.
3 - MMU and disk paging system.

LITADR Literal addressing mode.
Two bytes are used to indicate the addressing mode.
X'1111' means one byte addressing for literal constants, keytables, pictures, and formats.
The value is derived from the LITADR option in CREDIT..

ADRMOD Addressing mode.
A value 1 or 2 specifying which addressing mode is used.
This is derived from the ADRMOD option in CREDIT.

S:BTAB Swappable Work Block Table.
Pointer to the swappable work block table.