2. LKM REQUESTS

2.1. Introduction

As mentioned in section 1.4. the TOSS Monitor provides application tasks with the following services:

- Task scheduling
- Input / Output
- Buffer control
- Memory management
- Monitor clock control

These services are requested by executing a LKM instruction. The DATA directive following the LKM instruction must contain a numeric request type specifying the type of service required. Certain LKM requests require further information to be loaded into register(s) and possibly further DATA directive(s), depending upon the type of request.

Following a LKM request, registers A1 to A14 are saved by the Monitor. They are restored immediately prior to handing control back to the application task.

In the following lists LKM requests are described briefly under the generic headings mentioned above.
Detailed descriptions of each request type follow in section 2.2.

Task scheduling

Request type	Description
0	Switch the running task for the next available task in the dispatcher queue
3	Exit task
4	Restart a paused task
–4	Activate another task
5	Pause the running task
6	Delay the running task
–6	Activate another task after a delay

Note: if memory management is being used LKM request with activation must specify a start point in the same segment as the LKM request.

Input/output

Request type	Description
1	Perform normal input or output
–1	Perform input or output followed by task activation
2	Wait for completion of input or output
10	Abort a previous input or output request
15	Assign a file code to a file.
16	Assign an index to a data file.

Note: The LKM DATA (–)1 knows two special calls, the ATTACH/DETACH and INTERTASK COMMUNICATION; see I/O drivers, section 3.6.

If memory management is being used the LKM request with activation must specify a start point in the same segment as the LKM request.

Buffer Control

Request type	Description
−7	Allocate buffers to the task
−8	Release buffers from the task.

Memory Management:

Request type	Description
9	Load program segment into memory

Clock Control:

Request type	Description
12	Get time from monitor clock
13	Set time in monitor clock

2.2. UKM Request Reference

This section contains a detailed description of the calling sequence and use of each UKM request.
The descriptions are in request type order.

0 SWITCH TASKS 0

Calling sequence	:	LKM DATA 0
Type	:	Task scheduling
Description	:	Execution of the requesting task is suspended and control is handed to the next available task in the dispatcher queue (this task will have the same priority as the requesting task). The requesting task is placed at the back of the dispatcher queue.

1 NORMAL I/O 1

Calling sequence :

```
LDK  A7, code
LDKL A8, ecb-address
LKM
DATA 1
```

Type : Input/Output

Description : The I/O operation specified in register A7 will be performed. The device, memory buffer etc. to be used are specified in an event control block whose start address must be held in register A8.

If the wait bit in register A7 is set to 0, the task continues executing instructions (starting at the instruction following the I/O request) while the I/O operation is in progress. If the wait bit is set to 1 execution will be suspended until the I/O operation is complete. Execution will recommence at the instruction following the I/O request.

If the no-wait option is used a wait (request type 2) LKM request must later be issued to synchronise with the I/O completion.

The formats of register A7 and of the event control block are described in chapter 3 (I/O Drivers).

The calls ATTACH/DETACH and INTERTASK COMMUNICATION are described in section 3.6.

−1 I/O AND ACTIVATE −1

Calling sequence	: LDK A1, parameter LDK A7, code LDKL A8, ecb-address LKM DATA −1 DATA start address
Type	: Input/Output
Description	: An I/O operation is performed on the specified device. When this I/O operation is completed the current task is activated. If the task is still active at this time, the activation request is placed in the pending queue, see Appendix A. The task will be activated at "start-address" with "parameter" in register A1. "Parameter" is a means of passing information to the task being activated and its use depends upon the requirements of the application program. The I/O operation to be performed is specified in register A7. The device, memory buffer etc. to be used are specified in an event control block whose address must be held in register A8. The wait bit in register A7 must be zero, and the task continues executing instructions (starting at the instruction following the I/O request) while the I/O operation is in progress. An exit LKM request must be issued subsequent to an I/O and activation request so that the task can be re-activated on completion of the requested I/O. The calls ATTACH/DETACH and INTERTASK COMMUNICATION are described in section 3.6.

2 WAIT 2

Calling sequence	:	LDKL A8, ecb-address LKM DATA 2
Type	:	Input/Output
Description	:	Execution of the current task is suspended until the I/O operation associated with the event control block pointed to by register A8 is complete. Execution of this task will be resumed at the instruction following the wait request. This request is used after an I/O LKM request with no-wait. The wait LKM request synchronises the execution of the task with the completion of the I/O. The format of the event control block is described in chapter 3 (I/O Drivers).

3 EXIT 3

Calling sequence	:	LKM DATA 3
Type	:	Task Scheduling
Description	:	Execution of the current task is ended. The task can be re-activated subsequently by another task via a LKM request type –4 or –6. However, the previous contents of registers etc. will be lost.

4 RESTART 4

Calling sequence	:	LDK A7, tid LKM DATA 4
Type	:	Task scheduling
Description	:	The paused task identified by "tid" is restarted (i.e. put into the dispatcher queue). The restarted task must be in a paused state as a result of issuing a LKM request type 5. It will begin execution at instruction following the pause request. The requesting task (i.e. the one issuing the restart request) continues executing at the instruction following the restart request as soon as the restarted task has been placed in the dispatcher queue.

−4 ACTIVATE −4

Calling sequence	: LDK A1, parameter LDK A7, tid LKM DATA −4 DATA start-address
Type	: Task scheduling
Description	: The task with identifier "tid" is placed in the dispatcher queue. This task will be activated at the instruction indicated by "start-address" with "parameter" in register A1 and "tid" in register A2. "Parameter" is a means of passing information to the task being activated and its use depends upon the requirement of the application program. The contents of other registers is not relevant. If the task to be activated is already active the request becomes pending. The current task resumes execution at the instruction following the activate request. Execution of the requesting task recommences as soon as the activated task is placed in the dispatcher queue or becomes pending.

5 PAUSE 5

Calling sequence	:	LKM DATA 5
Type	:	Task scheduling
Description	:	Execution of the current task is suspended and the task is no longer considered for dispatching. The task can be restarted only by another task issuing a LKM request type 4.

6 DELAY 6

Calling sequence	:	LDKL A8, time LKM DATA 6
Type	:	Task scheduling
Description	:	Execution of the current task is suspended for the specified period of time. During this period the task will not be considered for dispatching. At the end of this period the task will again be put into the dispatcher queue. "Time" specifies the delay in multiples of 100 ms.

–6 DELAY AND ACTIVATE –6

Calling sequence :

```
LDK  A1, parameter
LDK  A7, tid
LDKL A8, time
LKM
DATA –6
DATA start-address
```

Type : Task scheduling

Description : When the period of time in register A8 has expired the task with identifier "tid" is placed in the dispatcher queue. The new task will be activated at the instruction indicated by "start-address". The task is activated with "parameter" in register A1 and "tid" in register A2. "Parameter is a means of passing information to the task being activated and its use depends upon the requirements of the application program. The contents of the other registers is not relevant.

If the task to be activated is already active the request becomes pending.

"Time" specifies the delay in multiples of 100ms.

Execution of the requesting task is resumed at the instruction following the delay and activate request. Execution of the requesting task re-commences as soon as the delay period starts.

−7 GET BUFFER −7

Calling sequence	:	LDK A7, number-of-buffers LKM DATA −7 DATA bcb-address
Type	:	Buffer control
Description	:	If the required number of contiguous buffers are free and there are no other requests queued for the buffer pool the specified number of buffers will be allocated to the requesting task. The start point of the first buffer allocated is returned to the task in register A8.

"bcb-address" refers to a two word buffer-control block containing the buffer pool start address and the buffer size in bytes.

If the required number of buffers are not available the task is placed in a "get buffer request" queue until the buffers are available. This queue is organised "first-in first-out".

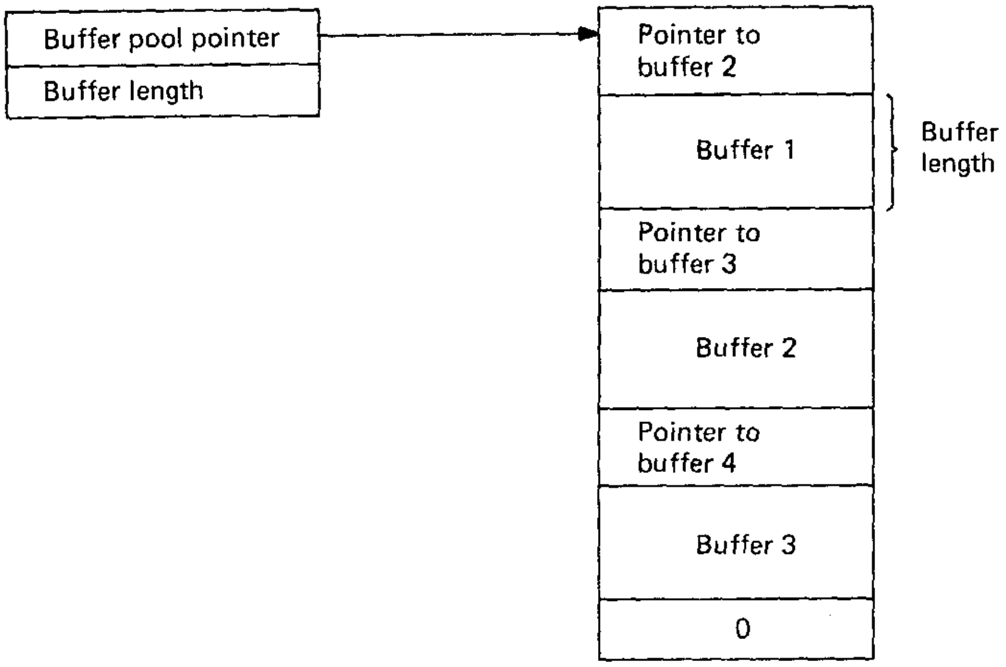
In order to use the get buffer and release buffer LKM requests (types—7 and −8) the application program must contain a correctly structured buffer pool. There must be one buffer pool per program. It may be set up using DATA and/or RES directives.

The buffer pool must be divided into fixed length buffers. Each buffer must be prefixed by a pointer containing the address of the next buffer in the pool. A zero pointer indicates the end of the buffer pool.

The get and release buffer requests both refer to a two word block containing the buffer pool start address and the buffer size in bytes. The buffer size excludes the pointer word at the start of the buffer.

The following diagram summarises the structure of a buffer pool.

Continued



The pointers at the start of each buffer must be set up by the application program before the first GET BUFFER request. However, they need not be **maintained** by application tasks. That is, when a GET BUFFER request is made an area of memory, including the pointer, is allocated to the task. The task may overwrite the buffer pointer in addition to the associated buffer. When a release buffer request is made the pointers are automatically regenerated by the Monitor.

Note: if memory management is being used, all GET BUFFER requests must be in the root segment of the program.

–8 RELEASE BUFFER –8

Calling sequence	:	LDKL A8, buffer-address LKM DATA –8 DATA bcb-address
Type	:	Buffer control
Description	:	The buffer pointed to by "buffer address" is returned to the buffer pool. The requesting task may not refer to this buffer again. "bcb-address" refers to a two word buffer control block containing the buffer pool start address and buffer size in bytes. For details on the use of a buffer pool see the GET BUFFER LKM request (type-7). Note: if memory management is being used, all RELEASE BUFFER requests must be in the root segment of the program.

9 LOAD SEGMENT 9

Calling sequence	: LDK A7, segment-no LDKL A8, segment-index LKM DATA 9
Type	: Memory management
Description	: The program segment identified by "segment-no" is loaded into memory from disk. Segments are numbered sequentially from 0 (root) when they are copied into disk by the TOSS utility CPP. Register A8 must contain an index (starting at 1) to the segment entry point table SEGENT. This table must contain a list of one word addresses comprising segment entry points relative to the start of the segment. If the specified segment does not exist, no segment will be loaded. Control will be handed to the instruction following the load segment request. If the specified segment cannot be loaded due to a hardware error, control is handed to the address at the top of the sub-routine call interpreter table (SUBTAB).

10 ABORT 10

Calling sequence	: LDKL A8, ecb-address LKM DATA 10
Type	: Input/Output
Description	: The I/O operation associated with the event control block pointed to by register A8 is aborted. The I/O operation must have been requested by the running task (therefore I/O with wait cannot be aborted).

Return code /C000 is set in the event control block if the abortion was successful. If the abort request is not accepted (e.g. because the I/O is completed) register A7 is set to −1.

An abort request can only be issued for I/O operations controlled by the following drivers.

- DRKB01 – General Keyboard
- DRTW01 – Typewriter
- DRTP01 }
- DRTP02 } Teller Terminal Printer
- DRSOP1 – System Operators Panel

The format of the event-control-block is described in chapter 3 (I/O Drivers).

12 GET TIME 12

Calling sequence	: LDKL A8, tcb-address LKM DATA 12
Type	: Monitor clock control
Description	: The time of day is copied from the monitor clock into the timer control block (tcb) pointed to by register A8. The time is stored in the tcb in ISO–7 characters. The tcb must be set up in the application program and must have the following format:

byte		
0	hour	hour
2	minute	minute
4	second	second

13 SET TIME 13

Calling sequence : LDKL A8, tcb-address
LKM
DATA 13

Type : Monitor clock control

Description : The time of day is copied from the timer control block (TCB) (set up by the application program and pointed to by register A8) into the monitor clock. The monitor colock is periodically updated by the real time clock in order to maintain the correct time of day.

The time must be held in the TCB in ISO–7 characters. The TCB must be set up in the application program and must have the following format :

0	hour	hour
2	minute	minute
4	second	second

15 ASSIGN FILE CODE 15

Calling sequence :

```
LDK   A7, task-control
LDKL  A8, asblk-address
LKM
DATA  15
```

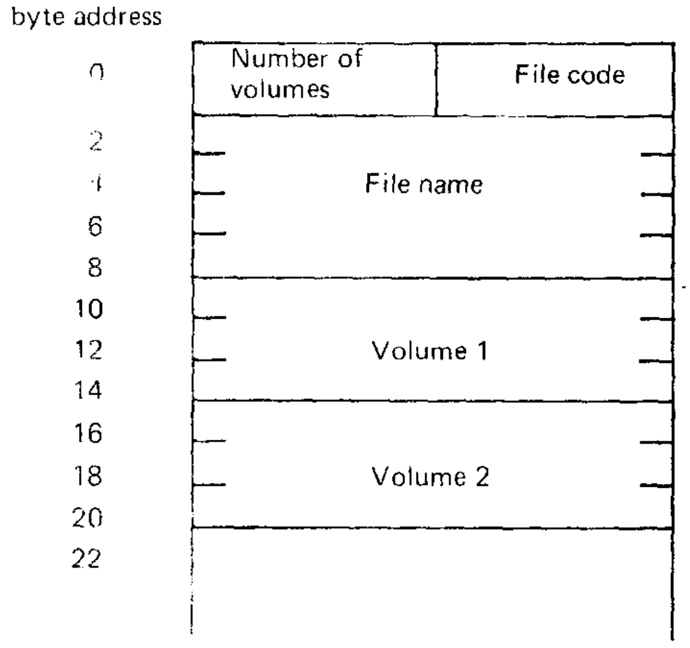
Type : Input/Output

Description : A file code is assigned to a data management file. The register A7 specifies whether the file is under control of one or more tasks. The file code and file name are specified in an assign-block pointed to by register A8.

The file must have been created previously by the data management utility CRF with the file name specified above, see section 4.5.

If task-control is 1 the file will be accessible only by the task which issued this 'assign file code' request.
If task-control is 0 this file is accessible by other tasks too.

The format of the assign block is as follows:



"Number of volumes" must not exceed 4.

"File code" is the file code to be assigned.

"File name" is a string of 8 characters, left-justified and padded with spaces, defining the file to be assigned.

"Volume" is a string of 6 characters, left-justified and padded with spaces, defining the volume where the file resides. A maximum of four volume names may be given.

A search is made in the volume table of contents (VTOC) for all volumes defined in the assign-block for the requested file name. In memory a file descriptor block (FDB) is built containing all relevant data for the file. The task table or common device table is updated with the FDB-address and the assigned file code. More than one file code may be assigned to the same data file.

Upon completion of the request, the system responds as follows:

A7	= 0	assignment performed
	≠ 0	assignment refused, for one of the following reasons
	= –1	Request error
	= 1	Disk I/O error
	= 2	no free entry in the common device table
	= 3	no file descriptor block available
	= 4	one or more volumes unknown
	= 5	file code already used
	= 6	filename unknown
	= 7	file section missing
	= 8	faulty disk format
	= 9	more than 4 extents exist

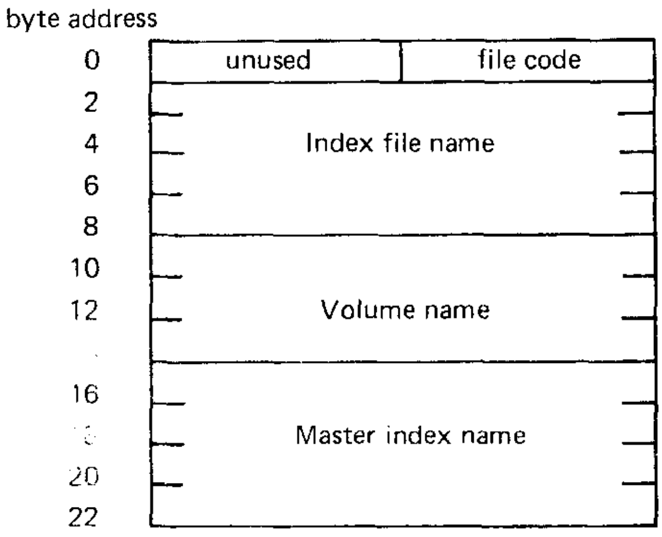
16 ASSIGN INDEX 16

Calling sequence : LDKL A8, asblk-address
LKM
DATA 16

Type : Input/Output

Description : An index file is assigned to a previously assigned data file. The index file is assigned to the file code specified in the assign block in which the index file name, volume name and master index name are specified.

The format of the assign block is as follows:



File code is the data file to which the index is assigned.

Index file name is a string of 8 characters, left-justified and padded with spaces, defining the index file name.

Volume name is a string of 6 characters, left-justified and padded with spaces, defining the volume on which the index file resides. An index file has access on only one volume.

Master index name is a string of 8 characters left-justified and padded with spaces, defining the name of the master index file to be used as index to the index file.

A maximum of 4 index files may be assigned to the same data file. The data file should have then different file codes, one for each index file.

All index-files in a file structure must be assigned within the same CDTAB/TTAB.

Continued

Upon completion of the request the system responds as follows:

A7	= 0	Assignment performed
	≠ 0	Assignment refused, for one of the following reasons:
	=−1	Request error
	= 1	Disk I/O error
	= 2	No free entry in CDTAB
	= 3	Not sufficient core memory available for Master Index or File descr. blocks
	= 4	Volume name unknown
	= 5	File already assigned from this task
	= 6	File name unknown. Incorrect file format.
	= 7	File section missing or found twice
	= 8	Faulty disk format
	= 9	More than 4 extents exist
	= 10	No data file assigned
	= 11	4 index files already assigned
	= 12	Size of disk buffers not sufficient
	= 13	Request busy. Repeat assign request