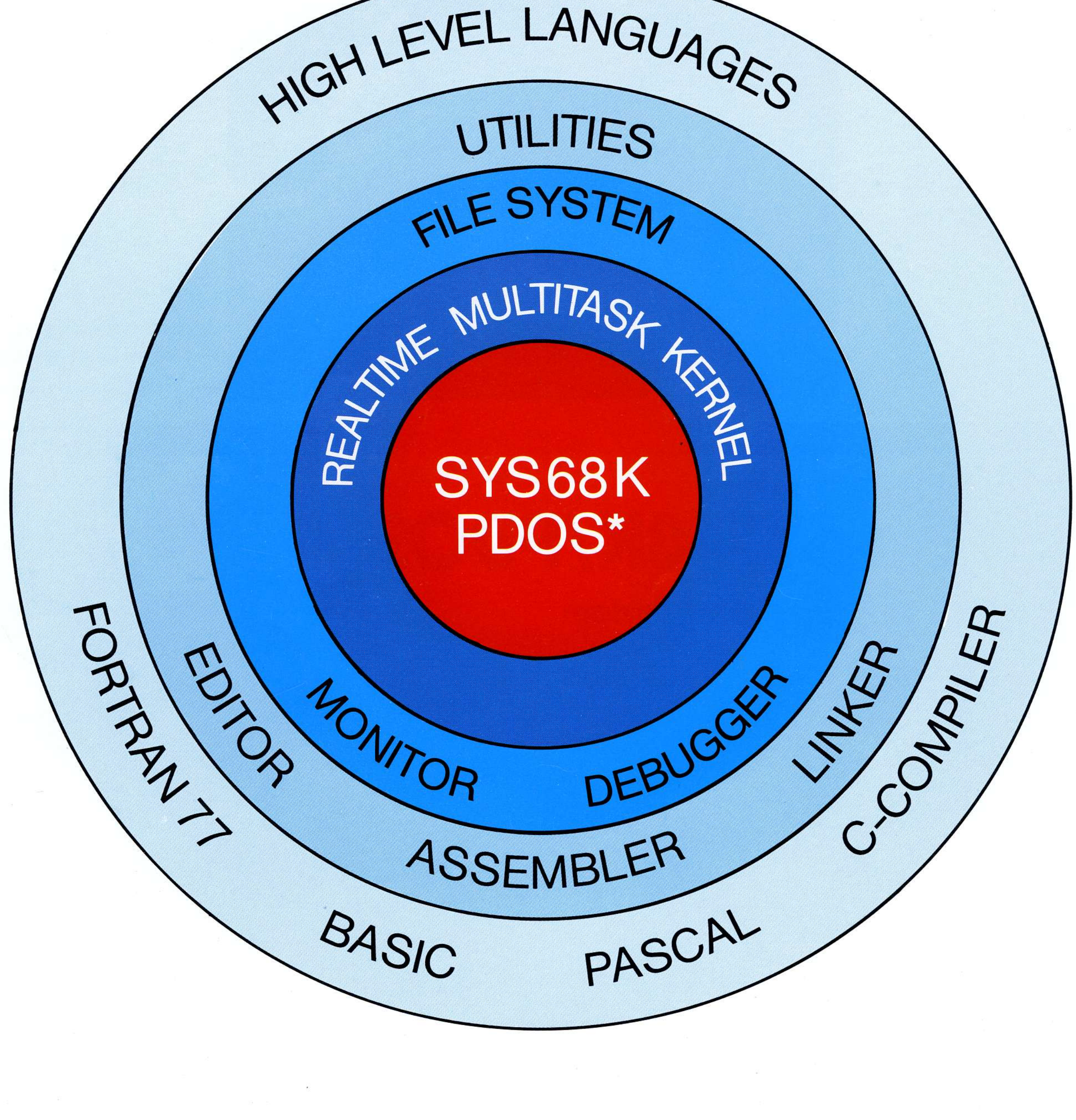
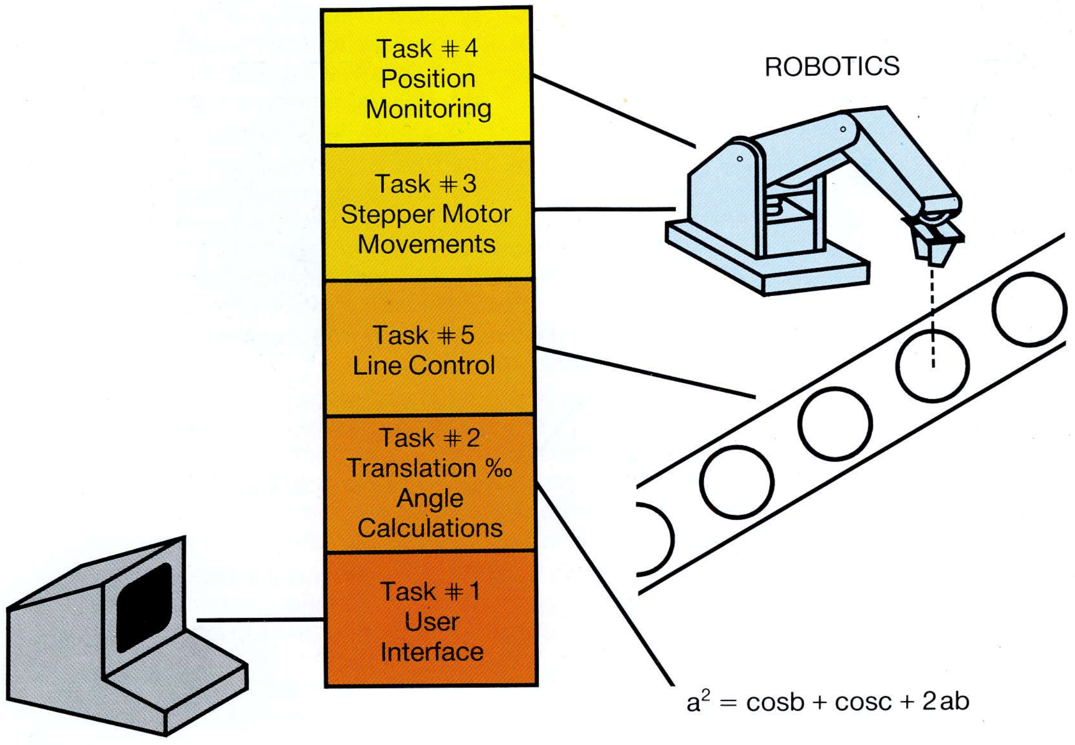
System 68000 VME
SYS68K/PDOS*
Operating System


HIGH LEVEL LANGUAGES
UTILITIES
FILE SYSTEM
REALTIME MULTITASK KERNEL
SYS68K
PDOS*
MONITOR
DEBUGGER
EDITOR
ASSEMBLER
LINKER
FORTRAN 77
BASIC
PASCAL
C-COMPILER

PDOS* DISK OPERATING SYSTEM

FEATURES:

- REAL TIME, MULTI-USER, MULTI-TASKING
- PRIORITIZED, ROUND-ROBIN SCHEDULING
- INTERTASK COMMUNICATION AND SYNCHRONIZATION
- TASK MEMORY MAP CONTROL FOR PROGRAM SECURITY
- FULL EXCEPTION PROCESSING
- SEQUENTIAL, RANDOM, AND SHARED FILE MANAGEMENT
- HARDWARE INDEPENDENCE
- 68000 LAYERED DESIGN OF KERNEL, FILE MANAGER, MONITOR
- COMPLETE FLOATING POINT SUPPORT
- CONFIGURABLE, MODULAR, ROMABLE STANDALONE SUPPORT
- NO MEMORY RESTRICTIONS


ROBOTICS
Task # 4 Position Monitoring
Task # 3 Stepper Motor Movements
Task # 5 Line Control
Task # 2 Translation ‰ Angle Calculations
Task # 1 User Interface
a² = cosb + cosc + 2ab

1. DESCRIPTION:

PDOS* is a powerful multi-user, multi-tasking operating system developed for the 32-bit Motorola 68000 processor family. This development software is designed for scientific, educational, industrial, and business applications.

PDOS* consists of a small, real time, multi-tasking kernel layered by file management, floating point, and user monitor modules. The 2k byte kernel provides synchronization and control of events occurring in a real-time environment using semaphores, events, messages, mailboxes, and suspension primitives. All user console I/O as well as other useful conversion and housekeeping routines are included in the PDOS* kernel.

The file management module supports named files with sequential, random, and shared access. Mass storage device independance is achieved through read and write logical sector primitives. The designer is relieved of real-time and task management problems as well as user console interaction and file manipulation so that efforts can be concentrated on the application.

Assembly language floating point applications are no longer a problem. Conversion modules, assembler directives, and operating system calls allow easy integration of floating point operations into user application programs.

2. FUNCTIONAL DESCRIPTION:

PDOS* KERNEL. PDOS* is written in 68000 assembly language for fast, efficient execution. The small kernel provides multi-tasking, real-time clock, event processing, and memory management functions. Ready tasks are scheduled using a prioritized, round-robin method. Three XOP vectors are used to interface over 75 system primitives to a user task.

MULTI-TASKING EXECUTION ENVIRONMENT. Tasks are the components comprising a real-time application. Each task is an independant program that shares the processor with other tasks in the system. Tasks provide a mechanism that allows a complicated application to be subdivided into several independant, understandable, and manageable modules. Real-time, concurrent tasks are allocated in 2k byte increments. Task system overhead is less than 1k bytes.

INTERTASK COMMUNICATION & SYNCHRONIZATION. Semaphores and events provide a low overhead facility for one task to signal another. Events can be used to indicate availability of a shared resource, timing pulses, or hardware interrupt occurrences. Messages and mailboxes are used in conjunction with system lock, unlock, suspend, and event primitives. PDOS* provides timing events that can be used in conjunction with desired events to prevent system lockouts. Other special system events signal character inputs and outputs.

MEMORY REQUIREMENTS. PDOS* is very memory efficient. The PDOS* kernel, floating point module, file manager, and user monitor utilities require only 8k bytes of memory plus an additional 4k bytes for system buffers and stacks. Most applications can be developed and implemented on the target system. Further memory reduction can be achieved by linking the user application to a 2k byte PDOS* kernel for a small, ROMable, standalone, multitasking module. A fast, 6k byte scientific orientated BASIC interpreter with real-time primitives provides interactive high level language support as well. For large system configurations, PDOS* effectively addresses up to a 32 bit address space.

FILE MANAGEMENT. The PDOS* file management module provides sequential, random, read only, and shared access to named files on a secondary storage device. These low overhead file primitives use a linked, random access file structure and a logical sector bit map for allocation of secondary storage. No file compaction is ever required. Files are time stamped with date of creation and last update. Up to 32 files can be simultaneously opened. Complete device independence is achieved through read and write logical sector primitives.

COMMAND LINE INTERPRETER. A resident command line interpreter allows multiple commands to be enterred on a single line. Command utilities such as append, define, delete, copy, rename, and show file are also resident and can be executed without destroying current memory programs. Other functions resident in the monitor include setting the baud rate of a port, checksumming memory, creating tasks, listing tasks, files and open file status, asking for help, setting file level, file attributes, interrupt mask, and system disk, and directing console output.

INTERRUPT MANAGEMENT. The PDOS* kernel handles user console, system clock, and other designated hardware interrupts. User consoles have interrupt driven character I/0 with type ahead. A task can be suspended pending a hardware or software event. PDOS* will switch control to a task suspended on an external event within 100 microseconds after the occurrence of the event (provided the system mask is high enough). Otherwise, a prioritized, round-robin scheduling of ready tasks occurs at 10 millisecond intervals.

PORTABILITY. PDOS* gives software portability through hardware independance of read/write logical sector primitives. All other hardware functions such as clocks, mappers, and UARTS are conveniently isolated for minimal customization to new 68000 based systems.

CUSTOMER SUPPORT. Numerous support utilities including virtual screen editors, assembler, linker, macroprocessor, disk diagnostics, link, and recovery, disk cataloging are standard. Single stepping, multiple break points, memory snap shots, save and restore task commands, and error trapping primitives are provided in all languages to aid in program debugging.

3. LANGUAGE SUPPORT:

– Basic	Standard Dartmouth Basic with enhancements, such as program debugging, inter-task communication and real-time support.
– Pascal	multi-pass, optimizing compiler that generates assembler text for the 68000 microprocessor. The PDOS* Pascal compiler implements a superset of the Pascal language defined by Jensen and Wirth.
– Fortran 77	compiler, supporting the full ANSI Fortran 77 standard (available later).
– C	Compiler for the C language (available later).

PDOS* SYSTEM CALLS

Append file	Execute PDOS call to D7.W
Baud console port	Exit to monitor
Build file directory list	Flush buffers
Debug call	Fix file name
Check for break character	Fix time and date
Convert binary to decimal	Free user memory
Convert binary to hex	Get character conditional
Convert to dec w/message	Get character
Check for break or pause	Get line in buffer
Convert to decimal in buffer	Get line in monitor buffer
Convert decimal to binary	Get line in user buffer
Close file w/attribute	Get memory limits
Chain command	Get next Parameter
Convert binary to hex in buffer	Get task message
Close file	Get user memory
Clear screen	Initialize sector
Copy file	Kill task
Create task block	Kill task message
Delay set / reset event	Load file
Define file	Load error register
Delete file	Look for name in file slots
Define trap vectors	Lock file
Return error do to monitor	Lock task

Load status register	Reset disk
List file directory	Read sector zero
Open non-exclusive random	Read time
Put buffer to console	Read task status
Put character(s) to console	Rewind file
Put CRLF	Set event flag
Put data to console	Open sequential
Put encoded message to console	Set port flag
Put line to console	Send task message
Put message to console	Set / read task priority
Position cursor	Suspend until interrupt
Position file	Enter supervisor mode
Put space to console	Swap to next task
Read bytes from file	Get disk size
Reset console inputs	Tab to column
Read port cursor position	Test event flag
Read next directory entry	Unpack date
Dump registers	Unlock file
Read directory entry by name	Unlock task
Read date	Unpack time
Read file attributes	Write bytes from file
Read line from file	Write date
Rename file	Write file attributes
Open random read only	Write line from file
Open random	Write sector
Read port status	Write time
Read sector	Zero file

PDOS* UTILITIES

MASM	68000 assembler.
MBACK	Disk backup.
BXREF	Basic cross reference.
COMP	Compare ASCII files.
MCHATLE	Changes attributes levels of selected files.
MDDMAP	Disk diagnostic. Reads files by links.
MDDUMP	Disk sector dump and alter.
MDNAME	Renames PDOS disks.
MFDUMP	Output logical dump of PDOS files.
FFRMT	Format logical unit.
MFSAVE	Restore files from links.
MINIT	Initialize PDOS disk.
MLDIR	Wild card list directory.
MLEVEL	Short listing by level.
LIBGEN	Create user module library.
QLINK	Link relocatable object.
MORDIR	Alphabetizes and compresses disk directory.
SYFILE	Generate SY file from OB.
MTERM	Set terminal cursor functions for task only.
MTRANS	Selektive file transfers.
RENUMBER	Renumbers BASIC programs.
UPTIME	System uptime

INTERRUPT MANAGEMENT. The PDOS* kernel handles user console, system clock, and other designated hardware interrupts. User consoles have interrupt driven character I/O with type ahead. A task can be suspended pending a hardware or software event. PDOS* will switch control to a task suspended on an external event within 100 microseconds after the occurrence of the event (provided the system mask is high enough). Otherwise, a prioritized, round-robin scheduling of ready tasks occurs at 10 millisecond intervals.

PORTABILITY. PDOS* gives software portability through hardware independance of read/write logical sector primitives. All other hardware functions such as clocks, mappers, and UARTS are conveniently isolated for minimal customization to new 68000 based systems.

CUSTOMER SUPPORT. Numerous support utilities including virtual screen editors, assembler, linker, macroprocessor, disk diagnostics, link, and recovery, disk cataloging are standard. Single stepping, multiple break points, memory snap shots, save and restore task commands, and error trapping primitives are provided in all languages to aid in program debugging.

3. LANGUAGE SUPPORT:

– Basic	Standard Dartmouth Basic with enhancements, such as program debugging, inter-task communication and real-time support.
– Pascal	multi-pass, optimizing compiler that generates assembler text for the 68000 microprocessor. The PDOS* Pascal compiler implements a superset of the Pascal language defined by Jensen and Wirth.
– Fortran 77	compiler, supporting the full ANSI Fortran 77 standard (available later).
– C	Compiler for the C language (available later).

PDOS* SYSTEM CALLS

Append file
Baud console port
Build file directory list
Debug call
Check for break character
Convert binary to decimal
Convert binary to hex
Convert to dec w/message
Check for break or pause
Convert to decimal in buffer
Convert decimal to binary
Close file w/attribute
Chain command
Convert binary to hex in buffer
Close file
Clear screen
Copy file
Create task block
Delay set / reset event
Define file
Delete file
Define trap vectors
Return error do to monitor

Execute PDOS call to D7.W
Exit to monitor
Flush buffers
Fix file name
Fix time and date
Free user memory
Get character conditional
Get character
Get line in buffer
Get line in monitor buffer
Get line in user buffer
Get memory limits
Get next Parameter
Get task message
Get user memory
Initialize sector
Kill task
Kill task message
Load file
Load error register
Look for name in file slots
Lock file
Lock task

PDOS* MONITOR COMMANDS

Append file	Kill task
Available memory	Load file
Baud port	List directory
Copy file	List tasks
Create task	Directory level
Define file	Make file
Delete file	PDOS debugger
Delete multiple file	Reset console
Date and time	RAM disk
Set/reset event	Rename file
PDOS BASIC	Reset disk
Free Memory	Set file attributes
File slot usage	Show file
Get memory	Send message
Execute	Disk space
Help	Spool unit
Set system date/time	System disk
If processor	Transparent mode
Set interrupt mask	Task priority
Kill message	Output unit
	Zero memory

4. HARDWARE CONFIGURATION:

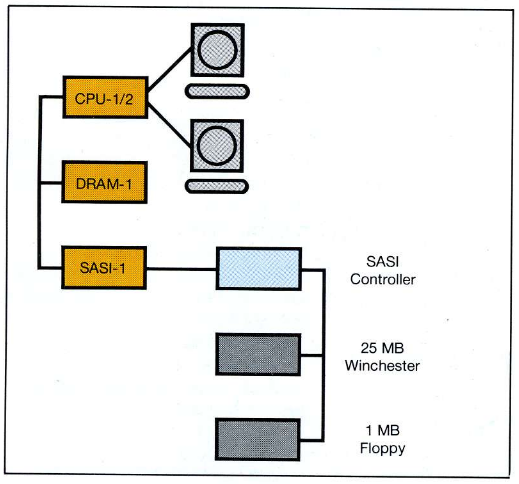
The SYS68K/PDOS* Operating System implementation requires one of the following hardware configurations:

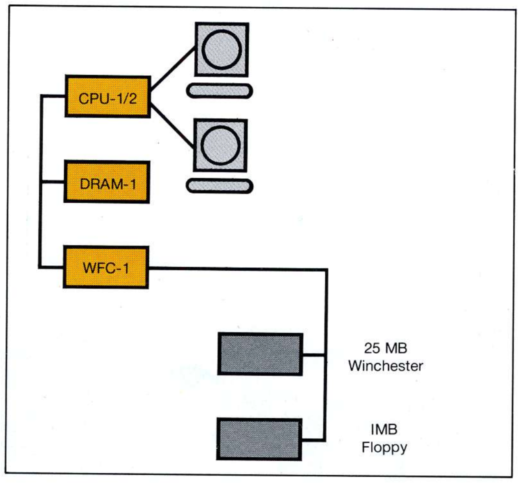
- CPU-1 or CPU-2
- SASI-1
- DTC-520
- DRAM-1

- CPU-1 or CPU-2
- WFC-1
- DRAM-1

FORCE COMPUTERS recommends the following devices for data storage:

Micropolis 1115	1 MB Floppy
Micropolis 1302	25 MB Winchester





5. DELIVERY MEDIA:

SYS68K/PDOS* is shipped on $5^{1}/_{4}$ inch Floppies. The package includes two boot EPROMS and documentation.

ORDERING INFORMATION

SYS68K/PDOS Part No. PDOS for CPU-1: 140001/10 Part No. PDOS for CPU-2: 140002/10	Operating System on 5 $^{1}/_{4}$ inch Floppies, boot EPROMS and documentation. Basic included.
SYS68K/PDOS/UM Part No. 800031	User's documentation.
SYS68K/PDOS/OV Part No. 800030	Product overview.
SYS68K/PDOS-PAS Part No. 140020	Pascal compiler and documentation.
SYS68K/PDOS-PAS/UM Part No. 800032	Pascal user's documentation.

NOTE: The SYS68K/PDOS* package is copyrighted and licenced by FORCE COMPUTERS GmbH and may only be used in accordance with and under the terms and conditions of such a licence aggreement.

* PDOS is a trade mark of EYRING RESEARCH INSTITUTE INC.

FORCE COMPUTERS INC.
727 University Ave.
Los Gatos, CA 95030
Phone (408) 354-3410
Tlx 172465
Telefax (408) 3957718

FORCE COMPUTERS GmbH
Daimlerstraße 9
D-8012 Ottobrunn
Telefon (0 89) 6 09 20 33
Telex 5 24 190 forc-d
Telefax (0 89) 6 09 77 93

FORCE COMPUTERS FRANCE
11, Rue Casteja
F-92100 Boulogne
Tel. (1) 620 37 37
Tlx 206 304 forc-f
Telefax (1) 621 35 19

7.85/5.0/PDOS/A2 Printed in West-Germany by