System-Globale-Variable (sysglob)

- **Vom User-Prozeß aus erfolgt der Zugriff mittels C-Bib-Funktion: *_os_getsys und die Bestimmung des Offsets in der Struktur sysglob mittels offsetof().***

Beispiel: Auslesen der Systemvariablen ticks/timeslice d_slice, Type u_int16 in die Variable my_slice.

```
#include <sysglob.h>
#include <types.h>
#include <stddef.h>


glob_buff my_slice;
_os_getsys ( offsetof (sysglobs, d_slice), sizeof (u_int16) ,
&my_slice);

...= my_slice.wrd ;
```

Bereits in sysglob.h definiert:

```
typedef union glob_buff {
    u_char   byt; /* 8-bit value */
    u_int16 wrd;  /* 16-bit value */
    u_int32 lng;  /* 32-bit value */
} glob_buff;
```

--*sysglob.h*---

```
#if !defined(_SYSGLOB_H)
#define _SYSGLOB_H

/*
 * $Header:   /h0/MWOS/OS9000/SRC/DEFS/VCS/sysglob.h_v   1.40   30 Jun 1998 11:47:20   afh  $
 * $Revision:   1.40  $
 */

/*--------------------------------------------------------------------------,
|                                                                           |
|             Copyright 1996,1997 by Microware Systems Corporation          |
|                          Reproduced Under License                         |
|                                                                           |
|  This source code is the proprietary confidential property of Microware   |
|  Systems Corporation, and is provided to licensee for documentation and   |
|  educational purposes only. Reproduction, publication, or distribution    |
|  in any form to any party other than the licensee is strictly prohibited. |
|                                                                           |
|---------------------------------------------------------------------------|
|                                                                           |
|  Edition History:                                                         |
|   #   Date    Comments                                                By  |
|  -- -------- ------------------------------------------------------- ---  |
|  01 87/02/23 Created                                                 rg   |
|  02 89/08/14 Added d_end variable to mark end of the system globals.  afh |
|  03 89/10/13 Changed time related globals.                                afh |
|  04 89/10/26 Changed declaration of dispatch tables.                      afh |
|  05 90/01/10 Added resource locks table.                                  afh |
|  06 94/01/25 Rearranged & added fields to the system globals.        afh |
|  07 94/09/09 Modified FPU routine pointer for the PowerPC.               afh |
|  08 94/11/04 Added edition ranges to dispatch table (used reserved).  afh |
|  09 94/12/09 Simplified inclusion of regs.h file.                afh |
|  10 95/03/06 Added d_cpudata array for CPU specific use.             afh |
|  11 95/03/30 Renamed "d_exctbl" to "d_globs" which will become a          |
|                  reserved portion of the globals.                          afh |
|  12 95/04/04 Added "d_exctbl" back in as a pointer to the exception       |
|                  table so that the table can be externally located.    afh |
|  13 95/04/28 Added structure tag to "dispatch_tbl" structure.         afh |
|  14 95/07/10 Change a PowerPC specific macro to a more general              |
|                  FPU emulation macro.                                          afh |
|  15 95/11/21 Added external debug module support fields and power           |
|                  management support fields.                                afh |
|  16 95/11/27 Changed revision 15's changes to be backward compatible. rry |
|                    ---- OS-9000/PPC V2.0 Release ----                        |
|  17 96/02/06 Added d_stackclean, d_stackcleanl, & d_callicpt call out    dwj     |
|              variables, as well as some reserved space.                          |
|  18 96/04/24 Added d_call_out array for future kernel callouts.          dwj     |
|  19 96/05/13 Added optional prototypes.                                            rry |
|  20 96/05/14 Added "d_excptrtn" for 80X86 platforms.                          afh |
|  21 96/06/21 Modified the types of the PowerMan support fields.          afh |
|              ---- OS-9000/x86 V2.1 Released ----                         |
|  22 96/10/03 Added some MIPS architecture conditionals.                cdg |
|              ---- OS-9000/PPC V2.1.1 Released ----                       |
|  23 96/10/15 Added SH conditionals.                                          afh |
|  24 96/12/05 Added ARM conditionals.                                         dwj |
|  25 97/01/16 Added d_excptexit callout.                                            dwj   |
|  26 97/03/05 Added d_dbgclean callout.                                             dwj   |
|  27 97/03/14 Fixed d_cctldata to be a pointer                          rkw |
|  28 97/04/10 MIPS targets use _FPU_NEW model.                          cdg |
|              ---- OS-9000/ARMv3 V2.2 Released ----                       |
|  29 97/06/16 Added prototype for "_os_config".                               afh |
|  30 97/06/17 Added Sparc support.                                                  afh |
|  31 97/06/18 Fixed the "_os_config" prototype.                               afh |
|              ---- OS-9000/ARMv3 V2.2.1 Released ----                     |
|  32 97/06/30 Removed _FPUEMUL conditionals.                                  afh |
|  33 97/08/05 Removed remains of x86 V1.x global definitions.           cdg |
|              x86 now uses _FPU_NEW model.                                |
|              ---- OS-9000/PPC V2.2.2 Released ----                       |
|              ---- OS-9000/SH3 V2.2.4 OS Component Released ----          |
|              ---- OS-9000/ARM V2.2.3 OS Component Released ----          |
|  34 98/01/30 Removed conditional around d_vectors and d_excptrtn.        |
|              Now used on all processors.                               gdb |
|              ---- OS-9000/SH3 V2.2.6 Released ----                       |
|              ---- OS-9000/SPARC V2.2.7 Released ----                     |
|              ---- OS-9000 OS Sub-component v2.2.8 Released ----          |
|              ---- OS-9000 OS Sub-component V2.2.9 Released ----          |
|              $$                    <RELEASE_INFO>                      $$ |
`---------------------------------------------------------------------------*/
.........

#if defined(_MPFMIPS) || defined(_MPF386)
#if !defined(_FPU_NEW)
#define _FPU_NEW             /* Target uses new FPU model */
#endif /*_FPU_NEW */
#endif

#define MAXLOCKS 128         /* maximum number of locks per lock block */

/* Service request attribute definitions. */
#define SR_UNKNOWN       0x80000000 /* unknown service request                      */
#define SR_BLOCK   0x40000000 /* service request potentialy blocks           */
#define SR_IRQOK   0x20000000 /* service request ok from IRQ context    */
#define SR_NOCONDEMN  0x10000000 /* service request restricted for condemned  */
#define SR_SWITCHABLE 0x08000000 /* service request is system-state switchable*/
#define SR_REPLACABLE 0x04000000 /* service request is replacable by users    */
#define SR_NOTIFYDBG  0x02000000 /* notify parent when child makes call       */
#define SR_REMOTE        0x01000000 /* service request remotely servicable (MP)  */


/* Exception jump table format */
typedef struct {
    u_int32     pea;              /* pea.l (XXX).w  instruction       */
    u_int16     jmp;              /* jmp (xxxx).l   instruction       */
    void  (*destin)();      /* absolute address (xxxx)         */
} excpt_jmp, *Excpt_jmp;


/* System call dispatch table declaration.  Note: the actual size of the */
/* system call dispatch tables is established during coldstart. */
typedef struct dispatch_tbl {                 /* dispatch table structure */
    u_int32
          (*service)();           /* service routine table                       */
    void
          *data;                        /* service routine data pointer table     */
    u_int32
          attr;                   /* service request attributes                  */
    u_int16
          ed_low,                       /* low bound of service edition           */
          ed_high;                /* high bound of service edition         */
} dispatch_tbl, *Dispatch_tbl;


/*   System global structure definition */
typedef struct sysglobs {
    u_int16
          d_id,                   /* sync code (system globals ID)                     */
          d_rev1[15];             /* reserved  first 32 bytes of globals              */
    u_int32
          d_mputyp,               /* mpu type 680XX/80X86 ect...                              */
          d_fputyp;               /* non-zero if FPU (identification) exists           */
    u_int16
```

```
            d_compat,               /* compatibility/control flags                              */
            d_minpty,               /* system minimum priority                                  */
            d_maxage,               /* system maximum natural age                               */
            d_maxsigs,              /* default maximum number of signals queued          */
            d_dsptblsz,             /* system call dispatch table size (entries)    */
            d_alloctype;            /* memory allocator type                                    */
      u_int32
            d_totram,               /* total RAM available at startup                    */
            d_blksiz,               /* system minimum allocatable block size        */
            d_minblk,               /* process minimun allocatable block size            */
            d_preempt,              /* system-state preemption flag: 0 = switchable */
            d_irqflag;              /* interrupt service context flag                    */
      u_int16
            d_tick,                       /* current tick (count down tick)                 */
            d_tcksec,               /* clock tickrate (number of ticks per second)  */
            d_slice,                /* current time slice remaining                      */
            d_tslice;               /* ticks per slice                                                */
      int32
            d_elapse;               /* time to elapse before system proc is summoned*/
      u_int32
            d_time,                       /* system time: seconds since reference date   */
            d_ticks,                /* system heartbeat (current tick counter)           */
            d_actage,               /* active process queue age delta value         */
            d_unkirq,               /* unknown IRQ count (unserviced IRQ count)          */
            d_evid;                       /* event creation counter                                 */
      Mh_config
            d_init;                       /* pointer to initialization module               */
      Rominfo
            d_sysrom;               /* Bootstrap ROM information structure pointer */
      Evnt_tbl
            d_evtbl;                /* system event block table pointer                  */
      Mod_dir
            d_mdroot,               /* system module directory root node pointer    */
            d_shmdroot;             /* shared module directory root node pointer    */
      Proc_tbl
            d_prcdbt;               /* process descriptor block table pointer            */
      Pr_desc
            d_proc,                       /* pointer to current process descriptor       */
            d_sysprc,               /* pointer to system process descriptor         */
            d_fproc;                /* pointer to process with context in FPU regs. */
      Pr_desc
            d_activq[2],            /* active process queue head node                    */
            d_sleepq[2],            /* sleeping process queue head node                  */
            d_waitq[2];             /* waiting process queue head node                   */
      Thread
            d_thread[2],            /* system alarm thread queue head node               */
            d_alarm[2],             /* system timed alarm thread head node               */
            d_seths[2],             /* system execution thread queue head node           */
            d_frseths[2];           /* free system execution thread queue head node */
      Mem_color
            d_freemem[2],           /* head of system memory free list                   */
            d_shfree[2],            /* head of shared memory free list                   */
            d_shfrags[2];           /* head of shared memory fragment list               */
      u_char
            *d_mminlim,             /* minimum memory address allocatable                */
            *d_mmaxlim,             /* maximum memory address allocatable                */
            *d_addrlim,             /* highest address found during startup         */
            *d_sstklm,              /* System IRQ stack low bound                               */
            *d_sysstk;              /* system state IRQ pointer                          */
      u_int32
            d_locks[2+(MAXLOCKS*4)];/* resource lock table                           */
      Dispatch_tbl
            d_sysdis,               /* system service dispatch table pointer        */
            d_usrdis;               /* user service dispatch table pointer               */
      u_int32
            *d_globs[64];           /* reserved space                                           */
      u_int32
            (*d_clock)(),           /* pointer to system tick routine                    */
            (*d_sysdbg)(),          /* system debugger entry point address               */
            *d_dbgmem;              /* system debugger memory pointer                    */
      Mh_com
            d_fpumod;               /* pointer to FPU support module                     */
      u_int32
            *d_fpudata,             /* FPU static storage pointer                        */

                                          /* FPU module flags */
#define FPU_INIT1             /* initialize process FPU context */
#define FPU_TERM2             /* terminate process FPU context */
#define FPU_SWITCH      3           /* task switch process FPU context */
#define FPU_PUSH4             /* push process FPU context */
#define FPU_POP         5           /* pop process FPU context */
#define FPU_COPYIN      6           /* copy in process FPU context */
#define FPU_COPYOUT     7           /* copy out process FPU context */
            (*d_fpucntxt)(),  /* FPU software emulation context routine      */
            d_fpursrv[3],           /* reserved                                                       */
            *d_ssmdata,             /* SSM static storage pointer                        */
            (*d_ssmperm)(),         /* SSM grant permissions routine pointer        */
            (*d_ssmprot)(),         /* SSM remove permissions routine pointer       */
            (*d_ssmatsk)(),         /* SSM allocate task routine pointer                 */
            (*d_ssmdtsk)(),         /* SSM delete task routine pointer                   */
            (*d_ssmchkm)(),         /* SSM check access routine pointer                  */
            *d_cache,               /* disk block cache pointer                                */
            *d_cctldata,            /* Cache control static storage pointer              */
            d_disinst,              /* Instruction cache disable depth                   */
            d_disdata,              /* Data cache disable depth                                */
            d_cachmode,             /* current cache control mode                        */
            (*d_cctl)();            /* Cache control routine pointer                     */

    #ifdef _MPF386
        Task_seg
            d_tss;                      /* pointer to task segment                                   */
    #endif /* _MPF386 */

        u_char
            *d_vectors;             /* exception table                                           */
        int
            (*d_excptrtn)();  /* kernels exception cleanup entry                   */

    #ifdef _MP                      /* multi-processor support variables */
        u_int16
            d_mpid,                     /* processor identifier                                      */
            d_mprsrv1;              /* reserved (maintain long alignment)                */
        Mpglobs
            d_mpglobs;              /* pointer to multi-processor system globals   */
        Spglobs
            d_spglobs;              /* pointer to processor's shared memory globals*/
        Pr_desc
            d_rwaitq[2];            /* remote processor call wait queue                  */
        Dispatch_tbl
            d_orgudt,               /* pointer to original user dispatch table           */
            d_orgsdt;               /* pointer to original system dispatch table   */
        Rio_stats
            d_urios,                /* user state remote I/O get/setstat service tbl*/
            d_srios;                /* system state remote I/O gs service table          */
        u_int32
            d_minriogs,             /* minimum range of user remote I/O get/setstats*/
            d_maxriogs,             /* maximum range of user remote I/O get/setstats*/
            d_mprsrv[8];            /* reserved space                                            */
    #endif /* _MP */

        u_int32
            d_cpudata[4],           /* cpu specific data support                         */
            d_rev2[8];              /* reserved space                                          */
        void
            *d_exctbl;              /* pointer to exception handler table                */
        u_int32
            d_switches;             /* context switch counter for idle checks      */
        void
            (*d_idle)(),            /* Idle loop call out routine                        */
            *d_idledata;            /* Idle loop call out routine data pointer           */
        u_int32
            d_dbgrsrv[8];           /* external debug support data space (reserved) */
        void
            (*d_stackclean)(void),  /*    Call out to stackclean routine                 */
            (*d_stackcleanl)(Regs), /*    Call out to stackcleanl routine                */
            (*d_callicpt)(void),    /*    Call out to callicpt routine             */
            (*d_excptexit)(void),   /*    Call out to excptexit routine            */
            (*d_dbgclean)(void);    /*    Call out to dbg stackclean routine       */
        void
            (*(d_call_out[3]))(void);     /*    Reserved call out routine space          */
        u_int32
            d_fpusize,              /* size of a process' FPU image                            */
            d_endresv[11];          /*    Reserved fields at end of structure            */

        u_int32
            d_end;                      /***** MARKER for end of system globals *****/
    } sysglobs, *Sysglobs;


    #ifdef _KERNEL
    sysglobs globs;
    #else
    extern sysglobs globs;
    #endif

    #define GLOB(var) globs.var

    /* System memory allocatory type (d_allloctype) definitions. */
    #define MA_STD 1/* standard first-fit allocator */
    #define MA_BUDDY 2    /* buddy (binary) allocator */

    /* Macro for testing whether or not to call the debugger */
    #define DBG_ENABLED (GLOB(d_sysrom)->rom_calldebug != 0)

    /* global queue head definitions */
    #define globs_activhd FAKEHD(Pr_desc, GLOB(d_activq[0]), p_queuen)
```

```
#define globs_waithd  FAKEHD(Pr_desc, GLOB(d_waitq[0]), p_queuen)
#define globs_sleephd FAKEHD(Pr_desc, GLOB(d_sleepq[0]), p_queuen)
/* #define d_freehd   FAKEHD(Mem_color, GLOB(d_freemem[0]), nxtptr) */
/* #define globs_freehd     FAKEHD(Mem_list, GLOB(d_fremem[0]), nxtptr) */
#define globs_threadhdFAKEHD(Thread, GLOB(d_alarm[0]), t_next)
#define globs_alarmhd FAKEHD(Thread, GLOB(d_thread[0]), t_next)

/* memory list queue heads */
#define globs_memhd         FAKEHD(Mem_color, GLOB(d_freemem[0]), nxt)
#define sys_frags           FAKEHD(Mem_color, GLOB(d_sysprc)->p_frag[0], nxt)
#define proc_frags          FAKEHD(Mem_color, GLOB(d_proc)->p_frag[0], nxt)
#define globs_shmemhd FAKEHD(Mem_color, GLOB(d_shfree[0]), nxt)
#define globs_shfrags FAKEHD(Mem_color, GLOB(d_shfrags[0]), nxt)

typedef union glob_buff {
    u_char      byt;        /* 8-bit value */
    u_int16 wrd;            /* 16-bit value */
    u_int32 lng;            /* 32-bit value */
} glob_buff;

#if defined(_ANSI_EXT) || defined(__STDC__) || defined(__cplusplus)
#define _OP(x) x
#else
#define _OP(x) ()
#endif

#if defined(__cplusplus)
extern "C" {
#endif /* __cplusplus */

error_code _os_getsys _OP((u_int32, u_int32, glob_buff *));
error_code _os_setsys _OP((u_int32, u_int32, glob_buff));
error_code _os_config _OP((u_int32, void *));

#if defined(_OPT_PROTOS)
int _getsys _OP((int, int));
int _setsys _OP((int, int, int));
#endif

#if defined(__cplusplus)
}
#endif /* __cplusplus */

#undef _OP

#endif
```