

```

#ifndef _MINIX_COM_H
#define _MINIX_COM_H

/*=====*/
*           Magic process numbers           *
/*=====*/

/* These may not be any valid endpoint (see <minix/endpoint.h>). */
#define ANY      0x7ace /* used to indicate 'any process' */
#define NONE     0x6ace /* used to indicate 'no process at all' */
#define SELF    0x8ace /* used to indicate 'own process' */
#define _MAX_MAGIC_PROC (SELF) /* used by <minix/endpoint.h>
to determine generation size */

/*=====*/
*           Process numbers of processes in the system image   *
/*=====*/

/* The values of several task numbers depend on whether they or other tasks
* are enabled. They are defined as (PREVIOUS_TASK - ENABLE_TASK) in general.
* ENABLE_TASK is either 0 or 1, so a task either gets a new number, or gets
* the same number as the previous task and is further unused. Note that the
* order should correspond to the order in the task table defined in table.c.
*/

/* Kernel tasks. These all run in the same address space. */
#define IDLE      -4 /* runs when no one else can run */
#define CLOCK    -3 /* alarms and other clock functions */
#define SYSTEM   -2 /* request system functionality */
#define KERNEL    -1 /* pseudo-process for IPC and scheduling */
#define HARDWARE  KERNEL /* for hardware interrupt handlers */

/* Number of tasks. Note that NR_PROCS is defined in <minix/config.h>. */
#define NR_TASKS  4

/* User-space processes, that is, device drivers, servers, and INIT. */
#define PM_PROC_NR  0 /* process manager */
#define FS_PROC_NR  1 /* file system */
#define RS_PROC_NR  2 /* reincarnation server */
#define MEM_PROC_NR  3 /* memory driver (RAM disk, null, etc.) */
#define LOG_PROC_NR  4 /* log device driver */
#define TTY_PROC_NR  5 /* terminal (TTY) driver */
#define DS_PROC_NR  6 /* data store server */
#define INIT_PROC_NR  7 /* init -- goes multiuser */

/* Number of processes contained in the system image. */
#define NR_BOOT_PROCS (NR_TASKS + INIT_PROC_NR + 1)

/*=====*/
*           Kernel notification types           *
/*=====*/

/* Kernel notification types. In principle, these can be sent to any process,
* so make sure that these types do not interfere with other message types.
* Notifications are prioritized because of the way they are unhold() and
* blocking notifications are delivered. The lowest numbers go first. The
* offset are used for the per-process notification bit maps.

```

```

*/
#define NOTIFY_MESSAGE          0x1000
#define NOTIFY_FROM(p_nr)      (NOTIFY_MESSAGE | ((p_nr) + NR_TASKS))
# define PROC_EVENT          NOTIFY_FROM(PM_PROC_NR) /* process status change */
# define SYN_ALARM NOTIFY_FROM(CLOCK) /* synchronous alarm */
# define SYS_SIG    NOTIFY_FROM(SYSTEM) /* system signal */
# define HARD_INT   NOTIFY_FROM(HARDWARE) /* hardware interrupt */
# define NEW_KSIG   NOTIFY_FROM(HARDWARE) /* new kernel signal */
# define FKEY_PRESSED NOTIFY_FROM(TTY_PROC_NR) /* function key press */
# define DEV_PING   NOTIFY_FROM(RS_PROC_NR) /* driver liveness ping */

/* Shorthands for message parameters passed with notifications. */
#define NOTIFY_SOURCE      m_source
#define NOTIFY_TYPE        m_type
#define NOTIFY_ARG         m2_l1
#define NOTIFY_TIMESTAMP   m2_l2
#define NOTIFY_FLAGS       m2_i1

/*-----*
*           Messages for BUS controller drivers           *
*-----*/
#define BUSEC_RQ_BASE      0x300 /* base for request types */
#define BUSEC_RS_BASE      0x380 /* base for response types */

#define BUSEC_PCI_INIT      (BUSEC_RQ_BASE + 0) /* First message to
* PCI driver
*/
#define BUSEC_PCI_FIRST_DEV (BUSEC_RQ_BASE + 1) /* Get index (and
* vid/did) of the
* first PCI device
*/
#define BUSEC_PCI_NEXT_DEV  (BUSEC_RQ_BASE + 2) /* Get index (and
* vid/did) of the
* next PCI device
*/
#define BUSEC_PCI_FIND_DEV  (BUSEC_RQ_BASE + 3) /* Get index of a
* PCI device based on
* bus/dev/function
*/
#define BUSEC_PCI_IDS       (BUSEC_RQ_BASE + 4) /* Get vid/did from an
* index
*/
#define BUSEC_PCI_DEV_NAME  (BUSEC_RQ_BASE + 5) /* Get the name of a
* PCI device
*/
#define BUSEC_PCI_SLOT_NAME (BUSEC_RQ_BASE + 6) /* Get the name of a
* PCI slot
*/
#define BUSEC_PCI_RESERVE   (BUSEC_RQ_BASE + 7) /* Reserve a PCI dev */
#define BUSEC_PCI_ATTR_R8   (BUSEC_RQ_BASE + 8) /* Read 8-bit
* attribute value
*/
#define BUSEC_PCI_ATTR_R16  (BUSEC_RQ_BASE + 9) /* Read 16-bit
* attribute value
*/
#define BUSEC_PCI_ATTR_R32  (BUSEC_RQ_BASE + 10) /* Read 32-bit
* attribute value

```

```

*/
#define BUSE_PCI_ATTR_W8      (BUSE_RQ_BASE + 11) /* Write 8-bit
*/
*/
#define BUSE_PCI_ATTR_W16    (BUSE_RQ_BASE + 12) /* Write 16-bit
*/
*/
#define BUSE_PCI_ATTR_W32    (BUSE_RQ_BASE + 13) /* Write 32-bit
*/
*/
#define BUSE_PCI_RESCAN      (BUSE_RQ_BASE + 14) /* Rescan bus */

/*-----*
*           Messages for BLOCK and CHARACTER device drivers           *
*-----*/

/* Message types for device drivers. */
#define DEV_RQ_BASE          0x400 /* base for device request types */
#define DEV_RS_BASE          0x500 /* base for device response types */

#define CANCEL                (DEV_RQ_BASE + 0) /* force a task to cancel */
#define DEV_READ              (DEV_RQ_BASE + 3) /* read from minor device */
#define DEV_WRITE             (DEV_RQ_BASE + 4) /* write to minor device */
#define DEV_IOCTL             (DEV_RQ_BASE + 5) /* I/O control code */
#define DEV_OPEN              (DEV_RQ_BASE + 6) /* open a minor device */
#define DEV_CLOSE             (DEV_RQ_BASE + 7) /* close a minor device */
#define DEV_SCATTER           (DEV_RQ_BASE + 8) /* write from a vector */
#define DEV_GATHER            (DEV_RQ_BASE + 9) /* read into a vector */
#define TTY_SETPGRP           (DEV_RQ_BASE + 10) /* set process group */
#define TTY_EXIT              (DEV_RQ_BASE + 11) /* process group leader exited */
#define DEV_SELECT            (DEV_RQ_BASE + 12) /* request select() attention */
#define DEV_STATUS            (DEV_RQ_BASE + 13) /* request driver status */

#define DEV_REPLY             (DEV_RS_BASE + 0) /* general task reply */
#define DEV_CLONED            (DEV_RS_BASE + 1) /* return cloned minor */
#define DEV_REVIVE            (DEV_RS_BASE + 2) /* driver revives process */
#define DEV_IO_READY          (DEV_RS_BASE + 3) /* selected device ready */
#define DEV_NO_STATUS         (DEV_RS_BASE + 4) /* empty status reply */

/* Field names for messages to block and character device drivers. */
#define DEVICE                m2_i1 /* major-minor device */
#define IO_ENDPT              m2_i2 /* which (proc/endpoint) wants I/O? */
#define COUNT                 m2_i3 /* how many bytes to transfer */
#define REQUEST               m2_i3 /* ioctl request code */
#define POSITION                m2_l1 /* file offset */
#define ADDRESS                m2_p1 /* core buffer address */

/* Field names for DEV_SELECT messages to device drivers. */
#define DEV_MINOR             m2_i1 /* minor device */
#define DEV_SEL_OPS           m2_i2 /* which select operations are requested */
#define DEV_SEL_WATCH         m2_i3 /* request notify if no operations are ready */

/* Field names used in reply messages from tasks. */
#define REP_ENDPT             m2_i1 /* # of proc on whose behalf I/O was done */
#define REP_STATUS            m2_i2 /* bytes transferred or error number */
#define SUSPEND               -998 /* status to suspend caller, reply later */

```

```

/* Field names for messages to TTY driver. */
#define TTY_LINE      DEVICE /* message parameter: terminal line */
#define TTY_REQUEST  COUNT  /* message parameter: ioctl request code */
#define TTY_SPEK     POSITION /* message parameter: ioctl speed, erasing */
#define TTY_FLAGS    m2_l2  /* message parameter: ioctl tty mode */
#define TTY_PGRP     m2_i3  /* message parameter: process group */

```

```

/* Field names for the QIC 02 status reply from tape driver */

```

```

#define TAPE_STAT0  m2_l1
#define TAPE_STAT1  m2_l2

```

```

/*-----*
 *           Messages for networking layer           *
 *-----*/

```

```

/* Message types for network layer requests. This layer acts like a driver. */

```

```

#define NW_OPEN      DEV_OPEN
#define NW_CLOSE     DEV_CLOSE
#define NW_READ      DEV_READ
#define NW_WRITE     DEV_WRITE
#define NW_IOCTL     DEV_IOCTL
#define NW_CANCEL    CANCEL

```

```

/* Base type for data link layer requests and responses. */

```

```

#define DL_RQ_BASE  0x800
#define DL_RS_BASE  0x900

```

```

/* Message types for data link layer requests. */

```

```

#define DL_WRITE      (DL_RQ_BASE + 3)
#define DL_WRITEV     (DL_RQ_BASE + 4)
#define DL_READ       (DL_RQ_BASE + 5)
#define DL_READV      (DL_RQ_BASE + 6)
#define DL_INIT       (DL_RQ_BASE + 7)
#define DL_STOP       (DL_RQ_BASE + 8)
#define DL_GETSTAT    (DL_RQ_BASE + 9)
#define DL_GETNAME    (DL_RQ_BASE + 10)

```

```

/* Message type for data link layer replies. */

```

```

#define DL_INIT_REPLY  (DL_RS_BASE + 20)
#define DL_TASK_REPLY  (DL_RS_BASE + 21)
#define DL_NAME_REPLY  (DL_RS_BASE + 22)

```

```

/* Field names for data link layer messages. */

```

```

#define DL_PORT      m2_i1
#define DL_PROC      m2_i2 /* endpoint */
#define DL_COUNT     m2_i3
#define DL_MODE      m2_l1
#define DL_CLCK      m2_l2
#define DL_ADDR      m2_p1
#define DL_STAT      m2_l1
#define DL_NAME      m3_ca1

```

```

/* Bits in 'DL_STAT' field of DL replies. */

```

```

# define DL_PACK_SEND  0x01
# define DL_PACK_RECV  0x02
# define DL_READ_IP    0x04

```

```

/* Bits in 'DL_MODE' field of DL requests. */
# define DL_NOMODE      0x0
# define DL_PROMISC_REQ 0x2
# define DL_MULTI_REQ   0x4
# define DL_BROAD_REQ   0x8

/*=====
*          SYSTASK request types and field names          *
*=====*/

/* System library calls are dispatched via a call vector, so be careful when
* modifying the system call numbers. The numbers here determine which call
* is made from the call vector.
*/
#define KERNEL_CALL 0x600    /* base for kernel calls to SYSTEM */

# define SYS_FORK      (KERNEL_CALL + 0) /* sys_fork() */
# define SYS_EXEC      (KERNEL_CALL + 1) /* sys_exec() */
# define SYS_EXIT      (KERNEL_CALL + 2) /* sys_exit() */
# define SYS_NICE      (KERNEL_CALL + 3) /* sys_nice() */
# define SYS_PRIVCTL   (KERNEL_CALL + 4) /* sys_privctl() */
# define SYS_TRACE     (KERNEL_CALL + 5) /* sys_trace() */
# define SYS_KILL      (KERNEL_CALL + 6) /* sys_kill() */

# define SYS_GETKSIG   (KERNEL_CALL + 7) /* sys_getsig() */
# define SYS_ENDKSIG   (KERNEL_CALL + 8) /* sys_endsig() */
# define SYS_SIGSEND   (KERNEL_CALL + 9) /* sys_sigsend() */
# define SYS_SIGRETURN (KERNEL_CALL + 10) /* sys_sigreturn() */

# define SYS_NEWMAP    (KERNEL_CALL + 11) /* sys_newmap() */
# define SYS_SEGCTL    (KERNEL_CALL + 12) /* sys_segctl() */
# define SYS_MEMSET     (KERNEL_CALL + 13) /* sys_memset() */

# define SYS_UMAP      (KERNEL_CALL + 14) /* sys_umap() */
# define SYS_VIRCOPY   (KERNEL_CALL + 15) /* sys_vircopy() */
# define SYS_PHYSCOPY  (KERNEL_CALL + 16) /* sys_physcopy() */
# define SYS_VIRVCOPY  (KERNEL_CALL + 17) /* sys_virvcopy() */
# define SYS_PHYSVCOPY (KERNEL_CALL + 18) /* sys_physvcopy() */

# define SYS_IRQCTL    (KERNEL_CALL + 19) /* sys_irqctl() */
# define SYS_INT86     (KERNEL_CALL + 20) /* sys_int86() */
# define SYS_DEVIO     (KERNEL_CALL + 21) /* sys_devio() */
# define SYS_SDEVIO    (KERNEL_CALL + 22) /* sys_sdevio() */
# define SYS_VDEVIO    (KERNEL_CALL + 23) /* sys_vdevio() */

# define SYS_SETALARM  (KERNEL_CALL + 24) /* sys_setalarm() */
# define SYS_TIMES     (KERNEL_CALL + 25) /* sys_times() */
# define SYS_GETINFO   (KERNEL_CALL + 26) /* sys_getinfo() */
# define SYS_ABORT     (KERNEL_CALL + 27) /* sys_abort() */
# define SYS_IOPENABLE (KERNEL_CALL + 28) /* sys_enable_iop() */
# define SYS_VM_SETBUF (KERNEL_CALL + 29) /* sys_vm_setbuf() */
# define SYS_VM_MAP    (KERNEL_CALL + 30) /* sys_vm_map() */

#define NR_SYS_CALLS 31 /* number of system calls */

/* Subfunctions for SYS_PRIVCTL */
#define SYS_PRIV_INIT 1 /* Initialize a privilege structure */

```

```

#define SYS_PRIV_ADD_IO      2    /* Add I/O range (struct io_range) */
#define SYS_PRIV_ADD_MEM    3    /* Add memory range (struct mem_range)
    */
#define SYS_PRIV_ADD_IRQ    4    /* Add IRQ */

/* Field names for SYS_MEMSET, SYS_SEGCTL. */
#define MEM_PTR      m2_p1    /* base */
#define MEM_COUNT    m2_l1    /* count */
#define MEM_PATTERN  m2_l2    /* pattern to write */
#define MEM_CHUNK_BASE m4_l1    /* physical base address */
#define MEM_CHUNK_SIZE m4_l2    /* size of mem chunk */
#define MEM_TOT_SIZE  m4_l3    /* total memory size */
#define MEM_CHUNK_TAG  m4_l4    /* tag to identify chunk of mem */

/* Field names for SYS_DEVIO, SYS_VDEVIO, SYS_SDEVIO. */
#define DIO_REQUEST m2_i3    /* device in or output */
# define DIO_INPUT  0    /* input */
# define DIO_OUTPUT  1    /* output */
#define DIO_TYPE      m2_i1    /* flag indicating byte, word, or long */
# define DIO_BYTE    'b'    /* byte type values */
# define DIO_WORD    'w'    /* word type values */
# define DIO_LONG    'l'    /* long type values */
#define DIO_PORT      m2_l1    /* single port address */
#define DIO_VALUE     m2_l2    /* single I/O value */
#define DIO_VEC_ADDR  m2_p1    /* address of buffer or (p,v)-pairs */
#define DIO_VEC_SIZE  m2_l2    /* number of elements in vector */
#define DIO_VEC_ENDPT m2_i2    /* number of process where vector is */

/* Field names for SYS_SIGNARLM, SYS_FLAGARLM, SYS_SYNCALRM. */
#define ALRM_EXP_TIME  m2_l1    /* expire time for the alarm call */
#define ALRM_ABS_TIME  m2_i2    /* set to 1 to use absolute alarm time */
#define ALRM_TIME_LEFT m2_l1    /* how many ticks were remaining */
#define ALRM_ENDPT     m2_i1    /* which process wants the alarm? */
#define ALRM_FLAG_PTR  m2_p1    /* virtual address of timeout flag */

/* Field names for SYS_IRQCTL. */
#define IRQ_REQUEST    m5_c1    /* what to do? */
# define IRQ_SETPOLICY  1    /* manage a slot of the IRQ table */
# define IRQ_RMPOLICY  2    /* remove a slot of the IRQ table */
# define IRQ_ENABLE    3    /* enable interrupts */
# define IRQ_DISABLE   4    /* disable interrupts */
#define IRQ_VECTOR     m5_c2    /* irq vector */
#define IRQ_POLICY     m5_i1    /* options for IRQCTL request */
# define IRQ_REENABLE  0x001    /* reenablr IRQ line after interrupt */
# define IRQ_BYTE      0x100    /* byte values */
# define IRQ_WORD      0x200    /* word values */
# define IRQ_LONG      0x400    /* long values */
#define IRQ_ENDPT      m5_i2    /* endpoint number, SELF, NONE */
#define IRQ_HOOK_ID    m5_l3    /* id of irq hook at kernel */

/* Field names for SYS_SEGCTL. */
#define SEG_SELECT     m4_l1    /* segment selector returned */
#define SEG_OFFSET     m4_l2    /* offset in segment returned */
#define SEG_PHYS       m4_l3    /* physical address of segment */
#define SEG_SIZE       m4_l4    /* segment size */
#define SEG_INDEX      m4_l5    /* segment index in remote map */

```

```
/* Field names for SYS_VIDCOPY. */
```

```
#define VID_REQUEST m4_l1 /* what to do? */
# define VID_VID_COPY 1 /* request vid_vid_copy() */
# define MEM_VID_COPY 2 /* request mem_vid_copy() */
#define VID_SRC_ADDR m4_l2 /* virtual address in memory */
#define VID_SRC_OFFSET m4_l3 /* offset in video memory */
#define VID_DST_OFFSET m4_l4 /* offset in video memory */
#define VID_CP_COUNT m4_l5 /* number of words to be copied */
```

```
/* Field names for SYS_ABORT. */
```

```
#define ABRT_HOW m1_i1 /* RBT_REBOOT, RBT_HALT, etc. */
#define ABRT_MON_ENDPT m1_i2 /* process where monitor params are */
#define ABRT_MON_LEN m1_i3 /* length of monitor params */
#define ABRT_MON_ADDR m1_p1 /* virtual address of monitor params */
```

```
/* Field names for _UMAP, _VIRCOPY, _PHYSCOPY. */
```

```
#define CP_SRC_SPACE m5_c1 /* T or D space (stack is also D) */
#define CP_SRC_ENDPT m5_i1 /* process to copy from */
#define CP_SRC_ADDR m5_l1 /* address where data come from */
#define CP_DST_SPACE m5_c2 /* T or D space (stack is also D) */
#define CP_DST_ENDPT m5_i2 /* process to copy to */
#define CP_DST_ADDR m5_l2 /* address where data go to */
#define CP_NR_BYTES m5_l3 /* number of bytes to copy */
```

```
/* Field names for SYS_VCOPY and SYS_VVIRCOPY. */
```

```
#define VCP_NR_OK m1_i2 /* number of successfull copies */
#define VCP_VEC_SIZE m1_i3 /* size of copy vector */
#define VCP_VEC_ADDR m1_p1 /* pointer to copy vector */
```

```
/* Field names for SYS_GETINFO. */
```

```
#define I_REQUEST m7_i3 /* what info to get */
# define GET_KINFO 0 /* get kernel information structure */
# define GET_IMAGE 1 /* get system image table */
# define GET_PROCTAB 2 /* get kernel process table */
# define GET_RANDOMNESS 3 /* get randomness buffer */
# define GET_MONPARAMS 4 /* get monitor parameters */
# define GET_KENV 5 /* get kernel environment string */
# define GET_IRQHOOKS 6 /* get the IRQ table */
# define GET_KMESSAGES 7 /* get kernel messages */
# define GET_PRIVTAB 8 /* get kernel privileges table */
# define GET_KADDRESSES 9 /* get various kernel addresses */
# define GET_SCHEDINFO 10 /* get scheduling queues */
# define GET_PROC 11 /* get process slot if given process */
# define GET_MACHINE 12 /* get machine information */
# define GET_LOCKTIMING 13 /* get lock()/unlock() latency timing */
# define GET_BIOSBUFFER 14 /* get a buffer for BIOS calls */
# define GET_LOADINFO 15 /* get load average information */
#define I_ENDPT m7_i4 /* calling process */
#define I_VAL_PTR m7_p1 /* virtual address at caller */
#define I_VAL_LEN m7_i1 /* max length of value */
#define I_VAL_PTR2 m7_p2 /* second virtual address */
#define I_VAL_LEN2_E m7_i2 /* second length, or proc nr */
# define GET_IRQACTIDS 16 /* get the IRQ masks */
/* CODIGO PRACTICA 3 */
# define GET_MESSGSN 17
```

```
/* Field names for SYS_TIMES. */
```

```

#define T_ENDPT      m4_l1  /* process to request time info for */
#define T_USER_TIME  m4_l1  /* user time consumed by process */
#define T_SYSTEM_TIME m4_l2 /* system time consumed by process */
#define T_CHILD_UTIME m4_l3 /* user time consumed by process' children */
#define T_CHILD_STIME m4_l4 /* sys time consumed by process' children */
#define T_BOOT_TICKS m4_l5 /* number of clock ticks since boot time */

/* vm_map */
#define VM_MAP_ENDPT      m4_l1
#define VM_MAP_MAPUNMAP   m4_l2
#define VM_MAP_BASE       m4_l3
#define VM_MAP_SIZE       m4_l4
#define VM_MAP_ADDR       m4_l5

/* Field names for SYS_TRACE, SYS_PRIVCTL. */
#define CTL_ENDPT      m2_i1 /* process number of the caller */
#define CTL_REQUEST    m2_i2 /* server control request */
#define CTL_MM_PRIV    m2_i3 /* privilege as seen by PM */
#define CTL_ARG_PTR    m2_p1 /* pointer to argument */
#define CTL_ADDRESS    m2_l1 /* address at traced process' space */
#define CTL_DATA       m2_l2 /* data field for tracing */

/* Field names for SYS_KILL, SYS_SIGCTL */
#define SIG_REQUEST    m2_l2 /* PM signal control request */
#define S_GETSIG       0     /* get pending kernel signal */
#define S_ENDSIG       1     /* finish a kernel signal */
#define S_SENDSIG      2     /* POSIX style signal handling */
#define S_SIGRETURN    3     /* return from POSIX handling */
#define S_KILL         4     /* servers kills process with signal */
#define SIG_ENDPT     m2_i1 /* process number for inform */
#define SIG_NUMBER     m2_i2 /* signal number to send */
#define SIG_FLAGS      m2_i3 /* signal flags field */
#define SIG_MAP        m2_l1 /* used by kernel to pass signal bit map */
#define SIG_CTXT_PTR   m2_p1 /* pointer to info to restore signal context */

/* Field names for SYS_FORK, _EXEC, _EXIT, _NEWMAP. */
#define PR_ENDPT      m1_i1 /* indicates a process */
#define PR_PRIORITY    m1_i2 /* process priority */
#define PR_SLOT       m1_i2 /* indicates a process slot */
#define PR_PID        m1_i3 /* process id at process manager */
#define PR_STACK_PTR   m1_p1 /* used for stack ptr in sys_exec, sys_getsp */
#define PR_TRACING     m1_i3 /* flag to indicate tracing is on/ off */
#define PR_NAME_PTR    m1_p2 /* tells where program name is for dmp */
#define PR_IP_PTR      m1_p3 /* initial value for ip after exec */
#define PR_MEM_PTR     m1_p1 /* tells where memory map is for sys_newmap */

/* Field names for SYS_INT86 */
#define INT86_REG86    m1_p1 /* pointer to registers */

/* Field names for SELECT (FS). */
#define SEL_NFDS       m8_i1
#define SEL_READFDS    m8_p1
#define SEL_WRITEFDS   m8_p2
#define SEL_ERRORFDS   m8_p3
#define SEL_TIMEOUT    m8_p4

/*-----*

```

```

*           Messages for the Reincarnation Server           *
*-----*/

#define RS_RQ_BASE      0x700

#define RS_UP           (RS_RQ_BASE + 0)    /* start system service */
#define RS_DOWN        (RS_RQ_BASE + 1)    /* stop system service */
#define RS_REFRESH     (RS_RQ_BASE + 2)    /* restart system service */
#define RS_RESCUE      (RS_RQ_BASE + 3)    /* set rescue directory */
#define RS_SHUTDOWN    (RS_RQ_BASE + 4)    /* alert about shutdown */

# define RS_CMD_ADDR   m1_p1              /* command string */
# define RS_CMD_LEN    m1_i1              /* length of command */
# define RS_PID        m1_i1              /* pid of system service */
# define RS_PERIOD     m1_i2              /* heartbeat period */
# define RS_DEV_MAJOR  m1_i3              /* major device number */

/*-----*
*           Messages for the Data Store Server           *
*-----*/

#define DS_RQ_BASE      0x800

#define DS_PUBLISH     (DS_RQ_BASE + 0)    /* publish information */
#define DS_RETRIEVE    (DS_RQ_BASE + 1)    /* retrieve information */
#define DS_SUBSCRIBE    (DS_RQ_BASE + 2)    /* subscribe to information */

# define DS_KEY        m2_i1              /* key for the information */
# define DS_FLAGS      m2_i2              /* flags provided by caller */
# define DS_AUTH       m2_p1              /* authorization of caller */
# define DS_VAL_L1     m2_l1              /* first long data value */
# define DS_VAL_L2     m2_l2              /* second long data value */

/*-----*
*           Miscellaneous messages used by TTY           *
*-----*/

/* Miscellaneous request types and field names, e.g. used by IS server. */
#define FKEY_CONTROL   98                 /* control a function key at the TTY */
# define FKEY_REQUEST  m2_i1              /* request to perform at TTY */
# define FKEY_MAP      10                 /* observe function key */
# define FKEY_UNMAP    11                 /* stop observing function key */
# define FKEY_EVENTS   12                 /* request open key presses */
# define FKEY_FKEYS    m2_l1              /* F1-F12 keys pressed */
# define FKEY_SFKEYS   m2_l2              /* Shift-F1-F12 keys pressed */
#define DIAGNOSTICS    100                /* output a string without FS in between */
# define DIAG_PRINT_BUF m1_p1
# define DIAG_BUF_COUNT m1_i1
# define DIAG_ENDPT    m1_i2
#define GET_KMESS      101 /* get kmess from TTY */
# define GETKM_PTR     m1_p1

#endif /* _MINIX_COM_H */

```