/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* uC/OS-II

\* The Real-Time Kernel

\*

\* (c) Copyright 1992-2009, Micrium, Weston, FL

\* All Rights Reserved

\*

\* File : uCOS\_II.H

\* By : Jean J. Labrosse

\* Version : V2.91

\*

\* LICENSING TERMS:

\* ---------------

\* uC/OS-II is provided in source form for FREE evaluation, for educational use or for peaceful research.

\* If you plan on using uC/OS-II in a commercial product you need to contact Micri祄 to properly license

\* its use in your product. We provide ALL the source code for your convenience and to help you experience

\* uC/OS-II. The fact that the source is provided does NOT mean that you can use it without paying a

\* licensing fee.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#ifndef OS\_uCOS\_II\_H

#define OS\_uCOS\_II\_H

#ifdef \_\_cplusplus

extern "C" {

#endif

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* uC/OS-II VERSION NUMBER

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#define OS\_VERSION 291u /\* Version of uC/OS-II (Vx.yy mult. by 100) \*/

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* INCLUDE HEADER FILES

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

//#include <app\_cfg.h>

#include <os\_cfg.h>

#include <os\_cpu.h>

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* MISCELLANEOUS

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#ifdef OS\_GLOBALS

#define OS\_EXT

#else

#define OS\_EXT extern

#endif

#ifndef OS\_FALSE

#define OS\_FALSE 0u

#endif

#ifndef OS\_TRUE

#define OS\_TRUE 1u

#endif

#define OS\_ASCII\_NUL (INT8U)0

#define OS\_PRIO\_SELF 0xFFu /\* Indicate SELF priority \*/

#if OS\_TASK\_STAT\_EN > 0u

#define OS\_N\_SYS\_TASKS 2u /\* Number of system tasks \*/

#else

#define OS\_N\_SYS\_TASKS 1u

#endif

#define OS\_TASK\_STAT\_PRIO (OS\_LOWEST\_PRIO - 1u) /\* Statistic task priority \*/

#define OS\_TASK\_IDLE\_PRIO (OS\_LOWEST\_PRIO) /\* IDLE task priority \*/

#if OS\_LOWEST\_PRIO <= 63u

#define OS\_EVENT\_TBL\_SIZE ((OS\_LOWEST\_PRIO) / 8u + 1u) /\* Size of event table \*/

#define OS\_RDY\_TBL\_SIZE ((OS\_LOWEST\_PRIO) / 8u + 1u) /\* Size of ready table \*/

#else

#define OS\_EVENT\_TBL\_SIZE ((OS\_LOWEST\_PRIO) / 16u + 1u)/\* Size of event table \*/

#define OS\_RDY\_TBL\_SIZE ((OS\_LOWEST\_PRIO) / 16u + 1u)/\* Size of ready table \*/

#endif

#define OS\_TASK\_IDLE\_ID 65535u /\* ID numbers for Idle, Stat and Timer tasks \*/

#define OS\_TASK\_STAT\_ID 65534u

#define OS\_TASK\_TMR\_ID 65533u

#define OS\_EVENT\_EN (((OS\_Q\_EN > 0u) && (OS\_MAX\_QS > 0u)) || (OS\_MBOX\_EN > 0u) || (OS\_SEM\_EN > 0u) || (OS\_MUTEX\_EN > 0u))

#define OS\_TCB\_RESERVED ((OS\_TCB \*)1)

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* TASK STATUS (Bit definition for OSTCBStat)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#define OS\_STAT\_RDY 0x00u /\* Ready to run \*/

#define OS\_STAT\_SEM 0x01u /\* Pending on semaphore \*/

#define OS\_STAT\_MBOX 0x02u /\* Pending on mailbox \*/

#define OS\_STAT\_Q 0x04u /\* Pending on queue \*/

#define OS\_STAT\_SUSPEND 0x08u /\* Task is suspended \*/

#define OS\_STAT\_MUTEX 0x10u /\* Pending on mutual exclusion semaphore \*/

#define OS\_STAT\_FLAG 0x20u /\* Pending on event flag group \*/

#define OS\_STAT\_MULTI 0x80u /\* Pending on multiple events \*/

#define OS\_STAT\_PEND\_ANY (OS\_STAT\_SEM | OS\_STAT\_MBOX | OS\_STAT\_Q | OS\_STAT\_MUTEX | OS\_STAT\_FLAG)

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* TASK PEND STATUS (Status codes for OSTCBStatPend)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#define OS\_STAT\_PEND\_OK 0u /\* Pending status OK, not pending, or pending complete \*/

#define OS\_STAT\_PEND\_TO 1u /\* Pending timed out \*/

#define OS\_STAT\_PEND\_ABORT 2u /\* Pending aborted \*/

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* OS\_EVENT types

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#define OS\_EVENT\_TYPE\_UNUSED 0u

#define OS\_EVENT\_TYPE\_MBOX 1u

#define OS\_EVENT\_TYPE\_Q 2u

#define OS\_EVENT\_TYPE\_SEM 3u

#define OS\_EVENT\_TYPE\_MUTEX 4u

#define OS\_EVENT\_TYPE\_FLAG 5u

#define OS\_TMR\_TYPE 100u /\* Used to identify Timers ... \*/

/\* ... (Must be different value than OS\_EVENT\_TYPE\_xxx) \*/

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* EVENT FLAGS

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#define OS\_FLAG\_WAIT\_CLR\_ALL 0u /\* Wait for ALL the bits specified to be CLR (i.e. 0) \*/

#define OS\_FLAG\_WAIT\_CLR\_AND 0u

#define OS\_FLAG\_WAIT\_CLR\_ANY 1u /\* Wait for ANY of the bits specified to be CLR (i.e. 0) \*/

#define OS\_FLAG\_WAIT\_CLR\_OR 1u

#define OS\_FLAG\_WAIT\_SET\_ALL 2u /\* Wait for ALL the bits specified to be SET (i.e. 1) \*/

#define OS\_FLAG\_WAIT\_SET\_AND 2u

#define OS\_FLAG\_WAIT\_SET\_ANY 3u /\* Wait for ANY of the bits specified to be SET (i.e. 1) \*/

#define OS\_FLAG\_WAIT\_SET\_OR 3u

#define OS\_FLAG\_CONSUME 0x80u /\* Consume the flags if condition(s) satisfied \*/

#define OS\_FLAG\_CLR 0u

#define OS\_FLAG\_SET 1u

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Values for OSTickStepState

\*

\* Note(s): This feature is used by uC/OS-View.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#if OS\_TICK\_STEP\_EN > 0u

#define OS\_TICK\_STEP\_DIS 0u /\* Stepping is disabled, tick runs as mormal \*/

#define OS\_TICK\_STEP\_WAIT 1u /\* Waiting for uC/OS-View to set OSTickStepState to \_ONCE \*/

#define OS\_TICK\_STEP\_ONCE 2u /\* Process tick once and wait for next cmd from uC/OS-View \*/

#endif

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Possible values for 'opt' argument of OSSemDel(), OSMboxDel(), OSQDel() and OSMutexDel()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#define OS\_DEL\_NO\_PEND 0u

#define OS\_DEL\_ALWAYS 1u

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* OS???Pend() OPTIONS

\*

\* These #defines are used to establish the options for OS???PendAbort().

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#define OS\_PEND\_OPT\_NONE 0u /\* NO option selected \*/

#define OS\_PEND\_OPT\_BROADCAST 1u /\* Broadcast action to ALL tasks waiting \*/

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* OS???PostOpt() OPTIONS

\*

\* These #defines are used to establish the options for OSMboxPostOpt() and OSQPostOpt().

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#define OS\_POST\_OPT\_NONE 0x00u /\* NO option selected \*/

#define OS\_POST\_OPT\_BROADCAST 0x01u /\* Broadcast message to ALL tasks waiting \*/

#define OS\_POST\_OPT\_FRONT 0x02u /\* Post to highest priority task waiting \*/

#define OS\_POST\_OPT\_NO\_SCHED 0x04u /\* Do not call the scheduler if this option is selected \*/

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* TASK OPTIONS (see OSTaskCreateExt())

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#define OS\_TASK\_OPT\_NONE 0x0000u /\* NO option selected \*/

#define OS\_TASK\_OPT\_STK\_CHK 0x0001u /\* Enable stack checking for the task \*/

#define OS\_TASK\_OPT\_STK\_CLR 0x0002u /\* Clear the stack when the task is create \*/

#define OS\_TASK\_OPT\_SAVE\_FP 0x0004u /\* Save the contents of any floating-point registers \*/

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* TIMER OPTIONS (see OSTmrStart() and OSTmrStop())

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#define OS\_TMR\_OPT\_NONE 0u /\* No option selected \*/

#define OS\_TMR\_OPT\_ONE\_SHOT 1u /\* Timer will not automatically restart when it expires \*/

#define OS\_TMR\_OPT\_PERIODIC 2u /\* Timer will automatically restart when it expires \*/

#define OS\_TMR\_OPT\_CALLBACK 3u /\* OSTmrStop() option to call 'callback' w/ timer arg. \*/

#define OS\_TMR\_OPT\_CALLBACK\_ARG 4u /\* OSTmrStop() option to call 'callback' w/ new arg. \*/

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* TIMER STATES

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#define OS\_TMR\_STATE\_UNUSED 0u

#define OS\_TMR\_STATE\_STOPPED 1u

#define OS\_TMR\_STATE\_COMPLETED 2u

#define OS\_TMR\_STATE\_RUNNING 3u

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* ERROR CODES

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#define OS\_ERR\_NONE 0u

#define OS\_ERR\_EVENT\_TYPE 1u

#define OS\_ERR\_PEND\_ISR 2u

#define OS\_ERR\_POST\_NULL\_PTR 3u

#define OS\_ERR\_PEVENT\_NULL 4u

#define OS\_ERR\_POST\_ISR 5u

#define OS\_ERR\_QUERY\_ISR 6u

#define OS\_ERR\_INVALID\_OPT 7u

#define OS\_ERR\_ID\_INVALID 8u

#define OS\_ERR\_PDATA\_NULL 9u

#define OS\_ERR\_TIMEOUT 10u

#define OS\_ERR\_EVENT\_NAME\_TOO\_LONG 11u

#define OS\_ERR\_PNAME\_NULL 12u

#define OS\_ERR\_PEND\_LOCKED 13u

#define OS\_ERR\_PEND\_ABORT 14u

#define OS\_ERR\_DEL\_ISR 15u

#define OS\_ERR\_CREATE\_ISR 16u

#define OS\_ERR\_NAME\_GET\_ISR 17u

#define OS\_ERR\_NAME\_SET\_ISR 18u

#define OS\_ERR\_ILLEGAL\_CREATE\_RUN\_TIME 19u

#define OS\_ERR\_MBOX\_FULL 20u

#define OS\_ERR\_Q\_FULL 30u

#define OS\_ERR\_Q\_EMPTY 31u

#define OS\_ERR\_PRIO\_EXIST 40u

#define OS\_ERR\_PRIO 41u

#define OS\_ERR\_PRIO\_INVALID 42u

#define OS\_ERR\_SCHED\_LOCKED 50u

#define OS\_ERR\_SEM\_OVF 51u

#define OS\_ERR\_TASK\_CREATE\_ISR 60u

#define OS\_ERR\_TASK\_DEL 61u

#define OS\_ERR\_TASK\_DEL\_IDLE 62u

#define OS\_ERR\_TASK\_DEL\_REQ 63u

#define OS\_ERR\_TASK\_DEL\_ISR 64u

#define OS\_ERR\_TASK\_NAME\_TOO\_LONG 65u

#define OS\_ERR\_TASK\_NO\_MORE\_TCB 66u

#define OS\_ERR\_TASK\_NOT\_EXIST 67u

#define OS\_ERR\_TASK\_NOT\_SUSPENDED 68u

#define OS\_ERR\_TASK\_OPT 69u

#define OS\_ERR\_TASK\_RESUME\_PRIO 70u

#define OS\_ERR\_TASK\_SUSPEND\_IDLE 71u

#define OS\_ERR\_TASK\_SUSPEND\_PRIO 72u

#define OS\_ERR\_TASK\_WAITING 73u

#define OS\_ERR\_TIME\_NOT\_DLY 80u

#define OS\_ERR\_TIME\_INVALID\_MINUTES 81u

#define OS\_ERR\_TIME\_INVALID\_SECONDS 82u

#define OS\_ERR\_TIME\_INVALID\_MS 83u

#define OS\_ERR\_TIME\_ZERO\_DLY 84u

#define OS\_ERR\_TIME\_DLY\_ISR 85u

#define OS\_ERR\_MEM\_INVALID\_PART 90u

#define OS\_ERR\_MEM\_INVALID\_BLKS 91u

#define OS\_ERR\_MEM\_INVALID\_SIZE 92u

#define OS\_ERR\_MEM\_NO\_FREE\_BLKS 93u

#define OS\_ERR\_MEM\_FULL 94u

#define OS\_ERR\_MEM\_INVALID\_PBLK 95u

#define OS\_ERR\_MEM\_INVALID\_PMEM 96u

#define OS\_ERR\_MEM\_INVALID\_PDATA 97u

#define OS\_ERR\_MEM\_INVALID\_ADDR 98u

#define OS\_ERR\_MEM\_NAME\_TOO\_LONG 99u

#define OS\_ERR\_NOT\_MUTEX\_OWNER 100u

#define OS\_ERR\_FLAG\_INVALID\_PGRP 110u

#define OS\_ERR\_FLAG\_WAIT\_TYPE 111u

#define OS\_ERR\_FLAG\_NOT\_RDY 112u

#define OS\_ERR\_FLAG\_INVALID\_OPT 113u

#define OS\_ERR\_FLAG\_GRP\_DEPLETED 114u

#define OS\_ERR\_FLAG\_NAME\_TOO\_LONG 115u

#define OS\_ERR\_PIP\_LOWER 120u

#define OS\_ERR\_TMR\_INVALID\_DLY 130u

#define OS\_ERR\_TMR\_INVALID\_PERIOD 131u

#define OS\_ERR\_TMR\_INVALID\_OPT 132u

#define OS\_ERR\_TMR\_INVALID\_NAME 133u

#define OS\_ERR\_TMR\_NON\_AVAIL 134u

#define OS\_ERR\_TMR\_INACTIVE 135u

#define OS\_ERR\_TMR\_INVALID\_DEST 136u

#define OS\_ERR\_TMR\_INVALID\_TYPE 137u

#define OS\_ERR\_TMR\_INVALID 138u

#define OS\_ERR\_TMR\_ISR 139u

#define OS\_ERR\_TMR\_NAME\_TOO\_LONG 140u

#define OS\_ERR\_TMR\_INVALID\_STATE 141u

#define OS\_ERR\_TMR\_STOPPED 142u

#define OS\_ERR\_TMR\_NO\_CALLBACK 143u

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* EVENT CONTROL BLOCK

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#if OS\_LOWEST\_PRIO <= 63u

typedef INT8U OS\_PRIO;

#else

typedef INT16U OS\_PRIO;

#endif

#if (OS\_EVENT\_EN) && (OS\_MAX\_EVENTS > 0u)

typedef struct os\_event {

INT8U OSEventType; /\* Type of event control block (see OS\_EVENT\_TYPE\_xxxx) \*/

void \*OSEventPtr; /\* Pointer to message or queue structure \*/

INT16U OSEventCnt; /\* Semaphore Count (not used if other EVENT type) \*/

OS\_PRIO OSEventGrp; /\* Group corresponding to tasks waiting for event to occur \*/

OS\_PRIO OSEventTbl[OS\_EVENT\_TBL\_SIZE]; /\* List of tasks waiting for event to occur \*/

#if OS\_EVENT\_NAME\_EN > 0u

INT8U \*OSEventName;

#endif

} OS\_EVENT;

#endif

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* EVENT FLAGS CONTROL BLOCK

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#if (OS\_FLAG\_EN > 0u) && (OS\_MAX\_FLAGS > 0u)

#if OS\_FLAGS\_NBITS == 8u /\* Determine the size of OS\_FLAGS (8, 16 or 32 bits) \*/

typedef INT8U OS\_FLAGS;

#endif

#if OS\_FLAGS\_NBITS == 16u

typedef INT16U OS\_FLAGS;

#endif

#if OS\_FLAGS\_NBITS == 32u

typedef INT32U OS\_FLAGS;

#endif

typedef struct os\_flag\_grp { /\* Event Flag Group \*/

INT8U OSFlagType; /\* Should be set to OS\_EVENT\_TYPE\_FLAG \*/

void \*OSFlagWaitList; /\* Pointer to first NODE of task waiting on event flag \*/

OS\_FLAGS OSFlagFlags; /\* 8, 16 or 32 bit flags \*/

#if OS\_FLAG\_NAME\_EN > 0u

INT8U \*OSFlagName;

#endif

} OS\_FLAG\_GRP;

typedef struct os\_flag\_node { /\* Event Flag Wait List Node \*/

void \*OSFlagNodeNext; /\* Pointer to next NODE in wait list \*/

void \*OSFlagNodePrev; /\* Pointer to previous NODE in wait list \*/

void \*OSFlagNodeTCB; /\* Pointer to TCB of waiting task \*/

void \*OSFlagNodeFlagGrp; /\* Pointer to Event Flag Group \*/

OS\_FLAGS OSFlagNodeFlags; /\* Event flag to wait on \*/

INT8U OSFlagNodeWaitType; /\* Type of wait: \*/

/\* OS\_FLAG\_WAIT\_AND \*/

/\* OS\_FLAG\_WAIT\_ALL \*/

/\* OS\_FLAG\_WAIT\_OR \*/

/\* OS\_FLAG\_WAIT\_ANY \*/

} OS\_FLAG\_NODE;

#endif

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* MESSAGE MAILBOX DATA

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#if OS\_MBOX\_EN > 0u

typedef struct os\_mbox\_data {

void \*OSMsg; /\* Pointer to message in mailbox \*/

OS\_PRIO OSEventTbl[OS\_EVENT\_TBL\_SIZE]; /\* List of tasks waiting for event to occur \*/

OS\_PRIO OSEventGrp; /\* Group corresponding to tasks waiting for event to occur \*/

} OS\_MBOX\_DATA;

#endif

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* MEMORY PARTITION DATA STRUCTURES

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#if (OS\_MEM\_EN > 0u) && (OS\_MAX\_MEM\_PART > 0u)

typedef struct os\_mem { /\* MEMORY CONTROL BLOCK \*/

void \*OSMemAddr; /\* Pointer to beginning of memory partition \*/

void \*OSMemFreeList; /\* Pointer to list of free memory blocks \*/

INT32U OSMemBlkSize; /\* Size (in bytes) of each block of memory \*/

INT32U OSMemNBlks; /\* Total number of blocks in this partition \*/

INT32U OSMemNFree; /\* Number of memory blocks remaining in this partition \*/

#if OS\_MEM\_NAME\_EN > 0u

INT8U \*OSMemName; /\* Memory partition name \*/

#endif

} OS\_MEM;

typedef struct os\_mem\_data {

void \*OSAddr; /\* Pointer to the beginning address of the memory partition \*/

void \*OSFreeList; /\* Pointer to the beginning of the free list of memory blocks \*/

INT32U OSBlkSize; /\* Size (in bytes) of each memory block \*/

INT32U OSNBlks; /\* Total number of blocks in the partition \*/

INT32U OSNFree; /\* Number of memory blocks free \*/

INT32U OSNUsed; /\* Number of memory blocks used \*/

} OS\_MEM\_DATA;

#endif

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* MUTUAL EXCLUSION SEMAPHORE DATA

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#if OS\_MUTEX\_EN > 0u

typedef struct os\_mutex\_data {

OS\_PRIO OSEventTbl[OS\_EVENT\_TBL\_SIZE]; /\* List of tasks waiting for event to occur \*/

OS\_PRIO OSEventGrp; /\* Group corresponding to tasks waiting for event to occur \*/

BOOLEAN OSValue; /\* Mutex value (OS\_FALSE = used, OS\_TRUE = available) \*/

INT8U OSOwnerPrio; /\* Mutex owner's task priority or 0xFF if no owner \*/

INT8U OSMutexPIP; /\* Priority Inheritance Priority or 0xFF if no owner \*/

} OS\_MUTEX\_DATA;

#endif

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* MESSAGE QUEUE DATA

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#if OS\_Q\_EN > 0u

typedef struct os\_q { /\* QUEUE CONTROL BLOCK \*/

struct os\_q \*OSQPtr; /\* Link to next queue control block in list of free blocks \*/

void \*\*OSQStart; /\* Pointer to start of queue data \*/

void \*\*OSQEnd; /\* Pointer to end of queue data \*/

void \*\*OSQIn; /\* Pointer to where next message will be inserted in the Q \*/

void \*\*OSQOut; /\* Pointer to where next message will be extracted from the Q \*/

INT16U OSQSize; /\* Size of queue (maximum number of entries) \*/

INT16U OSQEntries; /\* Current number of entries in the queue \*/

} OS\_Q;

typedef struct os\_q\_data {

void \*OSMsg; /\* Pointer to next message to be extracted from queue \*/

INT16U OSNMsgs; /\* Number of messages in message queue \*/

INT16U OSQSize; /\* Size of message queue \*/

OS\_PRIO OSEventTbl[OS\_EVENT\_TBL\_SIZE]; /\* List of tasks waiting for event to occur \*/

OS\_PRIO OSEventGrp; /\* Group corresponding to tasks waiting for event to occur \*/

} OS\_Q\_DATA;

#endif

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* SEMAPHORE DATA

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#if OS\_SEM\_EN > 0u

typedef struct os\_sem\_data {

INT16U OSCnt; /\* Semaphore count \*/

OS\_PRIO OSEventTbl[OS\_EVENT\_TBL\_SIZE]; /\* List of tasks waiting for event to occur \*/

OS\_PRIO OSEventGrp; /\* Group corresponding to tasks waiting for event to occur \*/

} OS\_SEM\_DATA;

#endif

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* TASK STACK DATA

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#if OS\_TASK\_CREATE\_EXT\_EN > 0u

typedef struct os\_stk\_data {

INT32U OSFree; /\* Number of free bytes on the stack \*/

INT32U OSUsed; /\* Number of bytes used on the stack \*/

} OS\_STK\_DATA;

#endif

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* TASK CONTROL BLOCK

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

typedef struct os\_tcb {

OS\_STK \*OSTCBStkPtr; /\* Pointer to current top of stack \*/

#if OS\_TASK\_CREATE\_EXT\_EN > 0u

void \*OSTCBExtPtr; /\* Pointer to user definable data for TCB extension \*/

OS\_STK \*OSTCBStkBottom; /\* Pointer to bottom of stack \*/

INT32U OSTCBStkSize; /\* Size of task stack (in number of stack elements) \*/

INT16U OSTCBOpt; /\* Task options as passed by OSTaskCreateExt() \*/

INT16U OSTCBId; /\* Task ID (0..65535) \*/

#endif

struct os\_tcb \*OSTCBNext; /\* Pointer to next TCB in the TCB list \*/

struct os\_tcb \*OSTCBPrev; /\* Pointer to previous TCB in the TCB list \*/

#if (OS\_EVENT\_EN)

OS\_EVENT \*OSTCBEventPtr; /\* Pointer to event control block \*/

#endif

#if (OS\_EVENT\_EN) && (OS\_EVENT\_MULTI\_EN > 0u)

OS\_EVENT \*\*OSTCBEventMultiPtr; /\* Pointer to multiple event control blocks \*/

#endif

#if ((OS\_Q\_EN > 0u) && (OS\_MAX\_QS > 0u)) || (OS\_MBOX\_EN > 0u)

void \*OSTCBMsg; /\* Message received from OSMboxPost() or OSQPost() \*/

#endif

#if (OS\_FLAG\_EN > 0u) && (OS\_MAX\_FLAGS > 0u)

#if OS\_TASK\_DEL\_EN > 0u

OS\_FLAG\_NODE \*OSTCBFlagNode; /\* Pointer to event flag node \*/

#endif

OS\_FLAGS OSTCBFlagsRdy; /\* Event flags that made task ready to run \*/

#endif

INT32U OSTCBDly; /\* Nbr ticks to delay task or, timeout waiting for event \*/

INT8U OSTCBStat; /\* Task status \*/

INT8U OSTCBStatPend; /\* Task PEND status \*/

INT8U OSTCBPrio; /\* Task priority (0 == highest) \*/

INT8U OSTCBX; /\* Bit position in group corresponding to task priority \*/

INT8U OSTCBY; /\* Index into ready table corresponding to task priority \*/

OS\_PRIO OSTCBBitX; /\* Bit mask to access bit position in ready table \*/

OS\_PRIO OSTCBBitY; /\* Bit mask to access bit position in ready group \*/

#if OS\_TASK\_DEL\_EN > 0u

INT8U OSTCBDelReq; /\* Indicates whether a task needs to delete itself \*/

#endif

#if OS\_TASK\_PROFILE\_EN > 0u

INT32U OSTCBCtxSwCtr; /\* Number of time the task was switched in \*/

INT32U OSTCBCyclesTot; /\* Total number of clock cycles the task has been running \*/

INT32U OSTCBCyclesStart; /\* Snapshot of cycle counter at start of task resumption \*/

OS\_STK \*OSTCBStkBase; /\* Pointer to the beginning of the task stack \*/

INT32U OSTCBStkUsed; /\* Number of bytes used from the stack \*/

#endif

#if OS\_TASK\_NAME\_EN > 0u

INT8U \*OSTCBTaskName;

#endif

#if OS\_TASK\_REG\_TBL\_SIZE > 0u

INT32U OSTCBRegTbl[OS\_TASK\_REG\_TBL\_SIZE];

#endif

} OS\_TCB;

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* TIMER DATA TYPES

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#if OS\_TMR\_EN > 0u

typedef void (\*OS\_TMR\_CALLBACK)(void \*ptmr, void \*parg);

typedef struct os\_tmr {

INT8U OSTmrType; /\* Should be set to OS\_TMR\_TYPE \*/

OS\_TMR\_CALLBACK OSTmrCallback; /\* Function to call when timer expires \*/

void \*OSTmrCallbackArg; /\* Argument to pass to function when timer expires \*/

void \*OSTmrNext; /\* Double link list pointers \*/

void \*OSTmrPrev;

INT32U OSTmrMatch; /\* Timer expires when OSTmrTime == OSTmrMatch \*/

INT32U OSTmrDly; /\* Delay time before periodic update starts \*/

INT32U OSTmrPeriod; /\* Period to repeat timer \*/

#if OS\_TMR\_CFG\_NAME\_EN > 0u

INT8U \*OSTmrName; /\* Name to give the timer \*/

#endif

INT8U OSTmrOpt; /\* Options (see OS\_TMR\_OPT\_xxx) \*/

INT8U OSTmrState; /\* Indicates the state of the timer: \*/

/\* OS\_TMR\_STATE\_UNUSED \*/

/\* OS\_TMR\_STATE\_RUNNING \*/

/\* OS\_TMR\_STATE\_STOPPED \*/

} OS\_TMR;

typedef struct os\_tmr\_wheel {

OS\_TMR \*OSTmrFirst; /\* Pointer to first timer in linked list \*/

INT16U OSTmrEntries;

} OS\_TMR\_WHEEL;

#endif

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* GLOBAL VARIABLES

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

OS\_EXT INT32U OSCtxSwCtr; /\* Counter of number of context switches \*/

#if (OS\_EVENT\_EN) && (OS\_MAX\_EVENTS > 0u)

OS\_EXT OS\_EVENT \*OSEventFreeList; /\* Pointer to list of free EVENT control blocks \*/

OS\_EXT OS\_EVENT OSEventTbl[OS\_MAX\_EVENTS];/\* Table of EVENT control blocks \*/

#endif

#if (OS\_FLAG\_EN > 0u) && (OS\_MAX\_FLAGS > 0u)

OS\_EXT OS\_FLAG\_GRP OSFlagTbl[OS\_MAX\_FLAGS]; /\* Table containing event flag groups \*/

OS\_EXT OS\_FLAG\_GRP \*OSFlagFreeList; /\* Pointer to free list of event flag groups \*/

#endif

#if OS\_TASK\_STAT\_EN > 0u

OS\_EXT INT8U OSCPUUsage; /\* Percentage of CPU used \*/

OS\_EXT INT32U OSIdleCtrMax; /\* Max. value that idle ctr can take in 1 sec. \*/

OS\_EXT INT32U OSIdleCtrRun; /\* Val. reached by idle ctr at run time in 1 sec. \*/

OS\_EXT BOOLEAN OSStatRdy; /\* Flag indicating that the statistic task is rdy \*/

OS\_EXT OS\_STK OSTaskStatStk[OS\_TASK\_STAT\_STK\_SIZE]; /\* Statistics task stack \*/

#endif

OS\_EXT INT8U OSIntNesting; /\* Interrupt nesting level \*/

OS\_EXT INT8U OSLockNesting; /\* Multitasking lock nesting level \*/

OS\_EXT INT8U OSPrioCur; /\* Priority of current task \*/

OS\_EXT INT8U OSPrioHighRdy; /\* Priority of highest priority task \*/

OS\_EXT OS\_PRIO OSRdyGrp; /\* Ready list group \*/

OS\_EXT OS\_PRIO OSRdyTbl[OS\_RDY\_TBL\_SIZE]; /\* Table of tasks which are ready to run \*/

OS\_EXT BOOLEAN OSRunning; /\* Flag indicating that kernel is running \*/

OS\_EXT INT8U OSTaskCtr; /\* Number of tasks created \*/

OS\_EXT volatile INT32U OSIdleCtr; /\* Idle counter \*/

#ifdef OS\_SAFETY\_CRITICAL\_IEC61508

OS\_EXT BOOLEAN OSSafetyCriticalStartFlag;

#endif

OS\_EXT OS\_STK OSTaskIdleStk[OS\_TASK\_IDLE\_STK\_SIZE]; /\* Idle task stack \*/

OS\_EXT OS\_TCB \*OSTCBCur; /\* Pointer to currently running TCB \*/

OS\_EXT OS\_TCB \*OSTCBFreeList; /\* Pointer to list of free TCBs \*/

OS\_EXT OS\_TCB \*OSTCBHighRdy; /\* Pointer to highest priority TCB R-to-R \*/

OS\_EXT OS\_TCB \*OSTCBList; /\* Pointer to doubly linked list of TCBs \*/

OS\_EXT OS\_TCB \*OSTCBPrioTbl[OS\_LOWEST\_PRIO + 1u]; /\* Table of pointers to created TCBs \*/

OS\_EXT OS\_TCB OSTCBTbl[OS\_MAX\_TASKS + OS\_N\_SYS\_TASKS]; /\* Table of TCBs \*/

#if OS\_TICK\_STEP\_EN > 0u

OS\_EXT INT8U OSTickStepState; /\* Indicates the state of the tick step feature \*/

#endif

#if (OS\_MEM\_EN > 0u) && (OS\_MAX\_MEM\_PART > 0u)

OS\_EXT OS\_MEM \*OSMemFreeList; /\* Pointer to free list of memory partitions \*/

OS\_EXT OS\_MEM OSMemTbl[OS\_MAX\_MEM\_PART];/\* Storage for memory partition manager \*/

#endif

#if (OS\_Q\_EN > 0u) && (OS\_MAX\_QS > 0u)

OS\_EXT OS\_Q \*OSQFreeList; /\* Pointer to list of free QUEUE control blocks \*/

OS\_EXT OS\_Q OSQTbl[OS\_MAX\_QS]; /\* Table of QUEUE control blocks \*/

#endif

#if OS\_TIME\_GET\_SET\_EN > 0u

OS\_EXT volatile INT32U OSTime; /\* Current value of system time (in ticks) \*/

#endif

#if OS\_TMR\_EN > 0u

OS\_EXT INT16U OSTmrFree; /\* Number of free entries in the timer pool \*/

OS\_EXT INT16U OSTmrUsed; /\* Number of timers used \*/

OS\_EXT INT32U OSTmrTime; /\* Current timer time \*/

OS\_EXT OS\_EVENT \*OSTmrSem; /\* Sem. used to gain exclusive access to timers \*/

OS\_EXT OS\_EVENT \*OSTmrSemSignal; /\* Sem. used to signal the update of timers \*/

OS\_EXT OS\_TMR OSTmrTbl[OS\_TMR\_CFG\_MAX]; /\* Table containing pool of timers \*/

OS\_EXT OS\_TMR \*OSTmrFreeList; /\* Pointer to free list of timers \*/

OS\_EXT OS\_STK OSTmrTaskStk[OS\_TASK\_TMR\_STK\_SIZE];

OS\_EXT OS\_TMR\_WHEEL OSTmrWheelTbl[OS\_TMR\_CFG\_WHEEL\_SIZE];

#endif

extern INT8U const OSUnMapTbl[256]; /\* Priority->Index lookup table \*/

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* FUNCTION PROTOTYPES

\* (Target Independent Functions)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* MISCELLANEOUS

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#if (OS\_EVENT\_EN)

#if (OS\_EVENT\_NAME\_EN > 0u)

INT8U OSEventNameGet (OS\_EVENT \*pevent,

INT8U \*\*pname,

INT8U \*perr);

void OSEventNameSet (OS\_EVENT \*pevent,

INT8U \*pname,

INT8U \*perr);

#endif

#if (OS\_EVENT\_MULTI\_EN > 0u)

INT16U OSEventPendMulti (OS\_EVENT \*\*pevents\_pend,

OS\_EVENT \*\*pevents\_rdy,

void \*\*pmsgs\_rdy,

INT32U timeout,

INT8U \*perr);

#endif

#endif

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* EVENT FLAGS MANAGEMENT

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#if (OS\_FLAG\_EN > 0u) && (OS\_MAX\_FLAGS > 0u)

#if OS\_FLAG\_ACCEPT\_EN > 0u

OS\_FLAGS OSFlagAccept (OS\_FLAG\_GRP \*pgrp,

OS\_FLAGS flags,

INT8U wait\_type,

INT8U \*perr);

#endif

OS\_FLAG\_GRP \*OSFlagCreate (OS\_FLAGS flags,

INT8U \*perr);

#if OS\_FLAG\_DEL\_EN > 0u

OS\_FLAG\_GRP \*OSFlagDel (OS\_FLAG\_GRP \*pgrp,

INT8U opt,

INT8U \*perr);

#endif

#if (OS\_FLAG\_EN > 0u) && (OS\_FLAG\_NAME\_EN > 0u)

INT8U OSFlagNameGet (OS\_FLAG\_GRP \*pgrp,

INT8U \*\*pname,

INT8U \*perr);

void OSFlagNameSet (OS\_FLAG\_GRP \*pgrp,

INT8U \*pname,

INT8U \*perr);

#endif

OS\_FLAGS OSFlagPend (OS\_FLAG\_GRP \*pgrp,

OS\_FLAGS flags,

INT8U wait\_type,

INT32U timeout,

INT8U \*perr);

OS\_FLAGS OSFlagPendGetFlagsRdy (void);

OS\_FLAGS OSFlagPost (OS\_FLAG\_GRP \*pgrp,

OS\_FLAGS flags,

INT8U opt,

INT8U \*perr);

#if OS\_FLAG\_QUERY\_EN > 0u

OS\_FLAGS OSFlagQuery (OS\_FLAG\_GRP \*pgrp,

INT8U \*perr);

#endif

#endif

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* MESSAGE MAILBOX MANAGEMENT

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#if OS\_MBOX\_EN > 0u

#if OS\_MBOX\_ACCEPT\_EN > 0u

void \*OSMboxAccept (OS\_EVENT \*pevent);

#endif

OS\_EVENT \*OSMboxCreate (void \*pmsg);

#if OS\_MBOX\_DEL\_EN > 0u

OS\_EVENT \*OSMboxDel (OS\_EVENT \*pevent,

INT8U opt,

INT8U \*perr);

#endif

void \*OSMboxPend (OS\_EVENT \*pevent,

INT32U timeout,

INT8U \*perr);

#if OS\_MBOX\_PEND\_ABORT\_EN > 0u

INT8U OSMboxPendAbort (OS\_EVENT \*pevent,

INT8U opt,

INT8U \*perr);

#endif

#if OS\_MBOX\_POST\_EN > 0u

INT8U OSMboxPost (OS\_EVENT \*pevent,

void \*pmsg);

#endif

#if OS\_MBOX\_POST\_OPT\_EN > 0u

INT8U OSMboxPostOpt (OS\_EVENT \*pevent,

void \*pmsg,

INT8U opt);

#endif

#if OS\_MBOX\_QUERY\_EN > 0u

INT8U OSMboxQuery (OS\_EVENT \*pevent,

OS\_MBOX\_DATA \*p\_mbox\_data);

#endif

#endif

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* MEMORY MANAGEMENT

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#if (OS\_MEM\_EN > 0u) && (OS\_MAX\_MEM\_PART > 0u)

OS\_MEM \*OSMemCreate (void \*addr,

INT32U nblks,

INT32U blksize,

INT8U \*perr);

void \*OSMemGet (OS\_MEM \*pmem,

INT8U \*perr);

#if OS\_MEM\_NAME\_EN > 0u

INT8U OSMemNameGet (OS\_MEM \*pmem,

INT8U \*\*pname,

INT8U \*perr);

void OSMemNameSet (OS\_MEM \*pmem,

INT8U \*pname,

INT8U \*perr);

#endif

INT8U OSMemPut (OS\_MEM \*pmem,

void \*pblk);

#if OS\_MEM\_QUERY\_EN > 0u

INT8U OSMemQuery (OS\_MEM \*pmem,

OS\_MEM\_DATA \*p\_mem\_data);

#endif

#endif

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* MUTUAL EXCLUSION SEMAPHORE MANAGEMENT

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#if OS\_MUTEX\_EN > 0u

#if OS\_MUTEX\_ACCEPT\_EN > 0u

BOOLEAN OSMutexAccept (OS\_EVENT \*pevent,

INT8U \*perr);

#endif

OS\_EVENT \*OSMutexCreate (INT8U prio,

INT8U \*perr);

#if OS\_MUTEX\_DEL\_EN > 0u

OS\_EVENT \*OSMutexDel (OS\_EVENT \*pevent,

INT8U opt,

INT8U \*perr);

#endif

void OSMutexPend (OS\_EVENT \*pevent,

INT32U timeout,

INT8U \*perr);

INT8U OSMutexPost (OS\_EVENT \*pevent);

#if OS\_MUTEX\_QUERY\_EN > 0u

INT8U OSMutexQuery (OS\_EVENT \*pevent,

OS\_MUTEX\_DATA \*p\_mutex\_data);

#endif

#endif

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* MESSAGE QUEUE MANAGEMENT

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#if (OS\_Q\_EN > 0u) && (OS\_MAX\_QS > 0u)

#if OS\_Q\_ACCEPT\_EN > 0u

void \*OSQAccept (OS\_EVENT \*pevent,

INT8U \*perr);

#endif

OS\_EVENT \*OSQCreate (void \*\*start,

INT16U size);

#if OS\_Q\_DEL\_EN > 0u

OS\_EVENT \*OSQDel (OS\_EVENT \*pevent,

INT8U opt,

INT8U \*perr);

#endif

#if OS\_Q\_FLUSH\_EN > 0u

INT8U OSQFlush (OS\_EVENT \*pevent);

#endif

void \*OSQPend (OS\_EVENT \*pevent,

INT32U timeout,

INT8U \*perr);

#if OS\_Q\_PEND\_ABORT\_EN > 0u

INT8U OSQPendAbort (OS\_EVENT \*pevent,

INT8U opt,

INT8U \*perr);

#endif

#if OS\_Q\_POST\_EN > 0u

INT8U OSQPost (OS\_EVENT \*pevent,

void \*pmsg);

#endif

#if OS\_Q\_POST\_FRONT\_EN > 0u

INT8U OSQPostFront (OS\_EVENT \*pevent,

void \*pmsg);

#endif

#if OS\_Q\_POST\_OPT\_EN > 0u

INT8U OSQPostOpt (OS\_EVENT \*pevent,

void \*pmsg,

INT8U opt);

#endif

#if OS\_Q\_QUERY\_EN > 0u

INT8U OSQQuery (OS\_EVENT \*pevent,

OS\_Q\_DATA \*p\_q\_data);

#endif

#endif

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* SEMAPHORE MANAGEMENT

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#if OS\_SEM\_EN > 0u

#if OS\_SEM\_ACCEPT\_EN > 0u

INT16U OSSemAccept (OS\_EVENT \*pevent);

#endif

OS\_EVENT \*OSSemCreate (INT16U cnt);

#if OS\_SEM\_DEL\_EN > 0u

OS\_EVENT \*OSSemDel (OS\_EVENT \*pevent,

INT8U opt,

INT8U \*perr);

#endif

void OSSemPend (OS\_EVENT \*pevent,

INT32U timeout,

INT8U \*perr);

#if OS\_SEM\_PEND\_ABORT\_EN > 0u

INT8U OSSemPendAbort (OS\_EVENT \*pevent,

INT8U opt,

INT8U \*perr);

#endif

INT8U OSSemPost (OS\_EVENT \*pevent);

#if OS\_SEM\_QUERY\_EN > 0u

INT8U OSSemQuery (OS\_EVENT \*pevent,

OS\_SEM\_DATA \*p\_sem\_data);

#endif

#if OS\_SEM\_SET\_EN > 0u

void OSSemSet (OS\_EVENT \*pevent,

INT16U cnt,

INT8U \*perr);

#endif

#endif

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* TASK MANAGEMENT

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#if OS\_TASK\_CHANGE\_PRIO\_EN > 0u

INT8U OSTaskChangePrio (INT8U oldprio,

INT8U newprio);

#endif

#if OS\_TASK\_CREATE\_EN > 0u

INT8U OSTaskCreate (void (\*task)(void \*p\_arg),

void \*p\_arg,

OS\_STK \*ptos,

INT8U prio);

#endif

#if OS\_TASK\_CREATE\_EXT\_EN > 0u

INT8U OSTaskCreateExt (void (\*task)(void \*p\_arg),

void \*p\_arg,

OS\_STK \*ptos,

INT8U prio,

INT16U id,

OS\_STK \*pbos,

INT32U stk\_size,

void \*pext,

INT16U opt);

#endif

#if OS\_TASK\_DEL\_EN > 0u

INT8U OSTaskDel (INT8U prio);

INT8U OSTaskDelReq (INT8U prio);

#endif

#if OS\_TASK\_NAME\_EN > 0u

INT8U OSTaskNameGet (INT8U prio,

INT8U \*\*pname,

INT8U \*perr);

void OSTaskNameSet (INT8U prio,

INT8U \*pname,

INT8U \*perr);

#endif

#if OS\_TASK\_SUSPEND\_EN > 0u

INT8U OSTaskResume (INT8U prio);

INT8U OSTaskSuspend (INT8U prio);

#endif

#if (OS\_TASK\_STAT\_STK\_CHK\_EN > 0u) && (OS\_TASK\_CREATE\_EXT\_EN > 0u)

INT8U OSTaskStkChk (INT8U prio,

OS\_STK\_DATA \*p\_stk\_data);

#endif

#if OS\_TASK\_QUERY\_EN > 0u

INT8U OSTaskQuery (INT8U prio,

OS\_TCB \*p\_task\_data);

#endif

#if OS\_TASK\_REG\_TBL\_SIZE > 0u

INT32U OSTaskRegGet (INT8U prio,

INT8U id,

INT8U \*perr);

void OSTaskRegSet (INT8U prio,

INT8U id,

INT32U value,

INT8U \*perr);

#endif

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* TIME MANAGEMENT

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

void OSTimeDly (INT32U ticks);

#if OS\_TIME\_DLY\_HMSM\_EN > 0u

INT8U OSTimeDlyHMSM (INT8U hours,

INT8U minutes,

INT8U seconds,

INT16U ms);

#endif

#if OS\_TIME\_DLY\_RESUME\_EN > 0u

INT8U OSTimeDlyResume (INT8U prio);

#endif

#if OS\_TIME\_GET\_SET\_EN > 0u

INT32U OSTimeGet (void);

void OSTimeSet (INT32U ticks);

#endif

void OSTimeTick (void);

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* TIMER MANAGEMENT

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#if OS\_TMR\_EN > 0u

OS\_TMR \*OSTmrCreate (INT32U dly,

INT32U period,

INT8U opt,

OS\_TMR\_CALLBACK callback,

void \*callback\_arg,

INT8U \*pname,

INT8U \*perr);

BOOLEAN OSTmrDel (OS\_TMR \*ptmr,

INT8U \*perr);

#if OS\_TMR\_CFG\_NAME\_EN > 0u

INT8U OSTmrNameGet (OS\_TMR \*ptmr,

INT8U \*\*pdest,

INT8U \*perr);

#endif

INT32U OSTmrRemainGet (OS\_TMR \*ptmr,

INT8U \*perr);

INT8U OSTmrStateGet (OS\_TMR \*ptmr,

INT8U \*perr);

BOOLEAN OSTmrStart (OS\_TMR \*ptmr,

INT8U \*perr);

BOOLEAN OSTmrStop (OS\_TMR \*ptmr,

INT8U opt,

void \*callback\_arg,

INT8U \*perr);

INT8U OSTmrSignal (void);

#endif

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* MISCELLANEOUS

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

void OSInit (void);

void OSIntEnter (void);

void OSIntExit (void);

#ifdef OS\_SAFETY\_CRITICAL\_IEC61508

void OSSafetyCriticalStart (void);

#endif

#if OS\_SCHED\_LOCK\_EN > 0u

void OSSchedLock (void);

void OSSchedUnlock (void);

#endif

void OSStart (void);

void OSStatInit (void);

INT16U OSVersion (void);

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* INTERNAL FUNCTION PROTOTYPES

\* (Your application MUST NOT call these functions)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#if OS\_TASK\_DEL\_EN > 0u

void OS\_Dummy (void);

#endif

#if (OS\_EVENT\_EN)

INT8U OS\_EventTaskRdy (OS\_EVENT \*pevent,

void \*pmsg,

INT8U msk,

INT8U pend\_stat);

void OS\_EventTaskWait (OS\_EVENT \*pevent);

void OS\_EventTaskRemove (OS\_TCB \*ptcb,

OS\_EVENT \*pevent);

#if (OS\_EVENT\_MULTI\_EN > 0u)

void OS\_EventTaskWaitMulti (OS\_EVENT \*\*pevents\_wait);

void OS\_EventTaskRemoveMulti (OS\_TCB \*ptcb,

OS\_EVENT \*\*pevents\_multi);

#endif

void OS\_EventWaitListInit (OS\_EVENT \*pevent);

#endif

#if (OS\_FLAG\_EN > 0u) && (OS\_MAX\_FLAGS > 0u)

void OS\_FlagInit (void);

void OS\_FlagUnlink (OS\_FLAG\_NODE \*pnode);

#endif

void OS\_MemClr (INT8U \*pdest,

INT16U size);

void OS\_MemCopy (INT8U \*pdest,

INT8U \*psrc,

INT16U size);

#if (OS\_MEM\_EN > 0u) && (OS\_MAX\_MEM\_PART > 0u)

void OS\_MemInit (void);

#endif

#if OS\_Q\_EN > 0u

void OS\_QInit (void);

#endif

void OS\_Sched (void);

#if (OS\_EVENT\_NAME\_EN > 0u) || (OS\_FLAG\_NAME\_EN > 0u) || (OS\_MEM\_NAME\_EN > 0u) || (OS\_TASK\_NAME\_EN > 0u)

INT8U OS\_StrLen (INT8U \*psrc);

#endif

void OS\_TaskIdle (void \*p\_arg);

void OS\_TaskReturn (void);

#if OS\_TASK\_STAT\_EN > 0u

void OS\_TaskStat (void \*p\_arg);

#endif

#if (OS\_TASK\_STAT\_STK\_CHK\_EN > 0u) && (OS\_TASK\_CREATE\_EXT\_EN > 0u)

void OS\_TaskStkClr (OS\_STK \*pbos,

INT32U size,

INT16U opt);

#endif

#if (OS\_TASK\_STAT\_STK\_CHK\_EN > 0u) && (OS\_TASK\_CREATE\_EXT\_EN > 0u)

void OS\_TaskStatStkChk (void);

#endif

INT8U OS\_TCBInit (INT8U prio,

OS\_STK \*ptos,

OS\_STK \*pbos,

INT16U id,

INT32U stk\_size,

void \*pext,

INT16U opt);

#if OS\_TMR\_EN > 0u

void OSTmr\_Init (void);

#endif

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* FUNCTION PROTOTYPES

\* (Target Specific Functions)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#if OS\_DEBUG\_EN > 0u

void OSDebugInit (void);

#endif

void OSInitHookBegin (void);

void OSInitHookEnd (void);

void OSTaskCreateHook (OS\_TCB \*ptcb);

void OSTaskDelHook (OS\_TCB \*ptcb);

void OSTaskIdleHook (void);

void OSTaskReturnHook (OS\_TCB \*ptcb);

void OSTaskStatHook (void);

OS\_STK \*OSTaskStkInit (void (\*task)(void \*p\_arg),

void \*p\_arg,

OS\_STK \*ptos,

INT16U opt);

#if OS\_TASK\_SW\_HOOK\_EN > 0u

void OSTaskSwHook (void);

#endif

void OSTCBInitHook (OS\_TCB \*ptcb);

#if OS\_TIME\_TICK\_HOOK\_EN > 0u

void OSTimeTickHook (void);

#endif

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* FUNCTION PROTOTYPES

\* (Application Specific Functions)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#if OS\_APP\_HOOKS\_EN > 0u

void App\_TaskCreateHook (OS\_TCB \*ptcb);

void App\_TaskDelHook (OS\_TCB \*ptcb);

void App\_TaskIdleHook (void);

void App\_TaskReturnHook (OS\_TCB \*ptcb);

void App\_TaskStatHook (void);

#if OS\_TASK\_SW\_HOOK\_EN > 0u

void App\_TaskSwHook (void);

#endif

void App\_TCBInitHook (OS\_TCB \*ptcb);

#if OS\_TIME\_TICK\_HOOK\_EN > 0u

void App\_TimeTickHook (void);

#endif

#endif

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* FUNCTION PROTOTYPES

\*

\* IMPORTANT: These prototypes MUST be placed in OS\_CPU.H

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#if 0

void OSStartHighRdy (void);

void OSIntCtxSw (void);

void OSCtxSw (void);

#endif

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* LOOK FOR MISSING #define CONSTANTS

\*

\* This section is used to generate ERROR messages at compile time if certain #define constants are

\* MISSING in OS\_CFG.H. This allows you to quickly determine the source of the error.

\*

\* You SHOULD NOT change this section UNLESS you would like to add more comments as to the source of the

\* compile time error.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* EVENT FLAGS

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#ifndef OS\_FLAG\_EN

#error "OS\_CFG.H, Missing OS\_FLAG\_EN: Enable (1) or Disable (0) code generation for Event Flags"

#else

#ifndef OS\_MAX\_FLAGS

#error "OS\_CFG.H, Missing OS\_MAX\_FLAGS: Max. number of Event Flag Groups in your application"

#else

#if OS\_MAX\_FLAGS > 65500u

#error "OS\_CFG.H, OS\_MAX\_FLAGS must be <= 65500"

#endif

#endif

#ifndef OS\_FLAGS\_NBITS

#error "OS\_CFG.H, Missing OS\_FLAGS\_NBITS: Determine #bits used for event flags, MUST be either 8, 16 or 32"

#endif

#ifndef OS\_FLAG\_WAIT\_CLR\_EN

#error "OS\_CFG.H, Missing OS\_FLAG\_WAIT\_CLR\_EN: Include code for Wait on Clear EVENT FLAGS"

#endif

#ifndef OS\_FLAG\_ACCEPT\_EN

#error "OS\_CFG.H, Missing OS\_FLAG\_ACCEPT\_EN: Include code for OSFlagAccept()"

#endif

#ifndef OS\_FLAG\_DEL\_EN

#error "OS\_CFG.H, Missing OS\_FLAG\_DEL\_EN: Include code for OSFlagDel()"

#endif

#ifndef OS\_FLAG\_NAME\_EN

#error "OS\_CFG.H, Missing OS\_FLAG\_NAME\_EN: Enable flag group names"

#endif

#ifndef OS\_FLAG\_QUERY\_EN

#error "OS\_CFG.H, Missing OS\_FLAG\_QUERY\_EN: Include code for OSFlagQuery()"

#endif

#endif

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* MESSAGE MAILBOXES

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#ifndef OS\_MBOX\_EN

#error "OS\_CFG.H, Missing OS\_MBOX\_EN: Enable (1) or Disable (0) code generation for MAILBOXES"

#else

#ifndef OS\_MBOX\_ACCEPT\_EN

#error "OS\_CFG.H, Missing OS\_MBOX\_ACCEPT\_EN: Include code for OSMboxAccept()"

#endif

#ifndef OS\_MBOX\_DEL\_EN

#error "OS\_CFG.H, Missing OS\_MBOX\_DEL\_EN: Include code for OSMboxDel()"

#endif

#ifndef OS\_MBOX\_PEND\_ABORT\_EN

#error "OS\_CFG.H, Missing OS\_MBOX\_PEND\_ABORT\_EN: Include code for OSMboxPendAbort()"

#endif

#ifndef OS\_MBOX\_POST\_EN

#error "OS\_CFG.H, Missing OS\_MBOX\_POST\_EN: Include code for OSMboxPost()"

#endif

#ifndef OS\_MBOX\_POST\_OPT\_EN

#error "OS\_CFG.H, Missing OS\_MBOX\_POST\_OPT\_EN: Include code for OSMboxPostOpt()"

#endif

#ifndef OS\_MBOX\_QUERY\_EN

#error "OS\_CFG.H, Missing OS\_MBOX\_QUERY\_EN: Include code for OSMboxQuery()"

#endif

#endif

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* MEMORY MANAGEMENT

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#ifndef OS\_MEM\_EN

#error "OS\_CFG.H, Missing OS\_MEM\_EN: Enable (1) or Disable (0) code generation for MEMORY MANAGER"

#else

#ifndef OS\_MAX\_MEM\_PART

#error "OS\_CFG.H, Missing OS\_MAX\_MEM\_PART: Max. number of memory partitions"

#else

#if OS\_MAX\_MEM\_PART > 65500u

#error "OS\_CFG.H, OS\_MAX\_MEM\_PART must be <= 65500"

#endif

#endif

#ifndef OS\_MEM\_NAME\_EN

#error "OS\_CFG.H, Missing OS\_MEM\_NAME\_EN: Enable memory partition names"

#endif

#ifndef OS\_MEM\_QUERY\_EN

#error "OS\_CFG.H, Missing OS\_MEM\_QUERY\_EN: Include code for OSMemQuery()"

#endif

#endif

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* MUTUAL EXCLUSION SEMAPHORES

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#ifndef OS\_MUTEX\_EN

#error "OS\_CFG.H, Missing OS\_MUTEX\_EN: Enable (1) or Disable (0) code generation for MUTEX"

#else

#ifndef OS\_MUTEX\_ACCEPT\_EN

#error "OS\_CFG.H, Missing OS\_MUTEX\_ACCEPT\_EN: Include code for OSMutexAccept()"

#endif

#ifndef OS\_MUTEX\_DEL\_EN

#error "OS\_CFG.H, Missing OS\_MUTEX\_DEL\_EN: Include code for OSMutexDel()"

#endif

#ifndef OS\_MUTEX\_QUERY\_EN

#error "OS\_CFG.H, Missing OS\_MUTEX\_QUERY\_EN: Include code for OSMutexQuery()"

#endif

#endif

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* MESSAGE QUEUES

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#ifndef OS\_Q\_EN

#error "OS\_CFG.H, Missing OS\_Q\_EN: Enable (1) or Disable (0) code generation for QUEUES"

#else

#ifndef OS\_MAX\_QS

#error "OS\_CFG.H, Missing OS\_MAX\_QS: Max. number of queue control blocks"

#else

#if OS\_MAX\_QS > 65500u

#error "OS\_CFG.H, OS\_MAX\_QS must be <= 65500"

#endif

#endif

#ifndef OS\_Q\_ACCEPT\_EN

#error "OS\_CFG.H, Missing OS\_Q\_ACCEPT\_EN: Include code for OSQAccept()"

#endif

#ifndef OS\_Q\_DEL\_EN

#error "OS\_CFG.H, Missing OS\_Q\_DEL\_EN: Include code for OSQDel()"

#endif

#ifndef OS\_Q\_FLUSH\_EN

#error "OS\_CFG.H, Missing OS\_Q\_FLUSH\_EN: Include code for OSQFlush()"

#endif

#ifndef OS\_Q\_PEND\_ABORT\_EN

#error "OS\_CFG.H, Missing OS\_Q\_PEND\_ABORT\_EN: Include code for OSQPendAbort()"

#endif

#ifndef OS\_Q\_POST\_EN

#error "OS\_CFG.H, Missing OS\_Q\_POST\_EN: Include code for OSQPost()"

#endif

#ifndef OS\_Q\_POST\_FRONT\_EN

#error "OS\_CFG.H, Missing OS\_Q\_POST\_FRONT\_EN: Include code for OSQPostFront()"

#endif

#ifndef OS\_Q\_POST\_OPT\_EN

#error "OS\_CFG.H, Missing OS\_Q\_POST\_OPT\_EN: Include code for OSQPostOpt()"

#endif

#ifndef OS\_Q\_QUERY\_EN

#error "OS\_CFG.H, Missing OS\_Q\_QUERY\_EN: Include code for OSQQuery()"

#endif

#endif

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* SEMAPHORES

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#ifndef OS\_SEM\_EN

#error "OS\_CFG.H, Missing OS\_SEM\_EN: Enable (1) or Disable (0) code generation for SEMAPHORES"

#else

#ifndef OS\_SEM\_ACCEPT\_EN

#error "OS\_CFG.H, Missing OS\_SEM\_ACCEPT\_EN: Include code for OSSemAccept()"

#endif

#ifndef OS\_SEM\_DEL\_EN

#error "OS\_CFG.H, Missing OS\_SEM\_DEL\_EN: Include code for OSSemDel()"

#endif

#ifndef OS\_SEM\_PEND\_ABORT\_EN

#error "OS\_CFG.H, Missing OS\_SEM\_PEND\_ABORT\_EN: Include code for OSSemPendAbort()"

#endif

#ifndef OS\_SEM\_QUERY\_EN

#error "OS\_CFG.H, Missing OS\_SEM\_QUERY\_EN: Include code for OSSemQuery()"

#endif

#ifndef OS\_SEM\_SET\_EN

#error "OS\_CFG.H, Missing OS\_SEM\_SET\_EN: Include code for OSSemSet()"

#endif

#endif

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* TASK MANAGEMENT

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#ifndef OS\_MAX\_TASKS

#error "OS\_CFG.H, Missing OS\_MAX\_TASKS: Max. number of tasks in your application"

#else

#if OS\_MAX\_TASKS < 2u

#error "OS\_CFG.H, OS\_MAX\_TASKS must be >= 2"

#endif

#if OS\_MAX\_TASKS > ((OS\_LOWEST\_PRIO - OS\_N\_SYS\_TASKS) + 1u)

#error "OS\_CFG.H, OS\_MAX\_TASKS must be <= OS\_LOWEST\_PRIO - OS\_N\_SYS\_TASKS + 1"

#endif

#endif

#if OS\_LOWEST\_PRIO > 254u

#error "OS\_CFG.H, OS\_LOWEST\_PRIO must be <= 254 in V2.8x and higher"

#endif

#ifndef OS\_TASK\_IDLE\_STK\_SIZE

#error "OS\_CFG.H, Missing OS\_TASK\_IDLE\_STK\_SIZE: Idle task stack size"

#endif

#ifndef OS\_TASK\_STAT\_EN

#error "OS\_CFG.H, Missing OS\_TASK\_STAT\_EN: Enable (1) or Disable(0) the statistics task"

#endif

#ifndef OS\_TASK\_STAT\_STK\_SIZE

#error "OS\_CFG.H, Missing OS\_TASK\_STAT\_STK\_SIZE: Statistics task stack size"

#endif

#ifndef OS\_TASK\_STAT\_STK\_CHK\_EN

#error "OS\_CFG.H, Missing OS\_TASK\_STAT\_STK\_CHK\_EN: Check task stacks from statistics task"

#endif

#ifndef OS\_TASK\_CHANGE\_PRIO\_EN

#error "OS\_CFG.H, Missing OS\_TASK\_CHANGE\_PRIO\_EN: Include code for OSTaskChangePrio()"

#endif

#ifndef OS\_TASK\_CREATE\_EN

#error "OS\_CFG.H, Missing OS\_TASK\_CREATE\_EN: Include code for OSTaskCreate()"

#endif

#ifndef OS\_TASK\_CREATE\_EXT\_EN

#error "OS\_CFG.H, Missing OS\_TASK\_CREATE\_EXT\_EN: Include code for OSTaskCreateExt()"

#endif

#ifndef OS\_TASK\_DEL\_EN

#error "OS\_CFG.H, Missing OS\_TASK\_DEL\_EN: Include code for OSTaskDel()"

#endif

#ifndef OS\_TASK\_NAME\_EN

#error "OS\_CFG.H, Missing OS\_TASK\_NAME\_EN: Enable task names"

#endif

#ifndef OS\_TASK\_SUSPEND\_EN

#error "OS\_CFG.H, Missing OS\_TASK\_SUSPEND\_EN: Include code for OSTaskSuspend() and OSTaskResume()"

#endif

#ifndef OS\_TASK\_QUERY\_EN

#error "OS\_CFG.H, Missing OS\_TASK\_QUERY\_EN: Include code for OSTaskQuery()"

#endif

#ifndef OS\_TASK\_REG\_TBL\_SIZE

#error "OS\_CFG.H, Missing OS\_TASK\_REG\_TBL\_SIZE: Include code for task specific registers"

#else

#if OS\_TASK\_REG\_TBL\_SIZE > 255u

#error "OS\_CFG.H, OS\_TASK\_REG\_TBL\_SIZE must be <= 255"

#endif

#endif

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* TIME MANAGEMENT

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#ifndef OS\_TICKS\_PER\_SEC

#error "OS\_CFG.H, Missing OS\_TICKS\_PER\_SEC: Sets the number of ticks in one second"

#endif

#ifndef OS\_TIME\_DLY\_HMSM\_EN

#error "OS\_CFG.H, Missing OS\_TIME\_DLY\_HMSM\_EN: Include code for OSTimeDlyHMSM()"

#endif

#ifndef OS\_TIME\_DLY\_RESUME\_EN

#error "OS\_CFG.H, Missing OS\_TIME\_DLY\_RESUME\_EN: Include code for OSTimeDlyResume()"

#endif

#ifndef OS\_TIME\_GET\_SET\_EN

#error "OS\_CFG.H, Missing OS\_TIME\_GET\_SET\_EN: Include code for OSTimeGet() and OSTimeSet()"

#endif

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* TIMER MANAGEMENT

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#ifndef OS\_TMR\_EN

#error "OS\_CFG.H, Missing OS\_TMR\_EN: When (1) enables code generation for Timer Management"

#elif OS\_TMR\_EN > 0u

#if OS\_SEM\_EN == 0u

#error "OS\_CFG.H, Semaphore management is required (set OS\_SEM\_EN to 1) when enabling Timer Management."

#error " Timer management require TWO semaphores."

#endif

#ifndef OS\_TMR\_CFG\_MAX

#error "OS\_CFG.H, Missing OS\_TMR\_CFG\_MAX: Determines the total number of timers in an application (2 .. 65500)"

#else

#if OS\_TMR\_CFG\_MAX < 2u

#error "OS\_CFG.H, OS\_TMR\_CFG\_MAX should be between 2 and 65500"

#endif

#if OS\_TMR\_CFG\_MAX > 65500u

#error "OS\_CFG.H, OS\_TMR\_CFG\_MAX should be between 2 and 65500"

#endif

#endif

#ifndef OS\_TMR\_CFG\_WHEEL\_SIZE

#error "OS\_CFG.H, Missing OS\_TMR\_CFG\_WHEEL\_SIZE: Sets the size of the timer wheel (1 .. 1023)"

#else

#if OS\_TMR\_CFG\_WHEEL\_SIZE < 2u

#error "OS\_CFG.H, OS\_TMR\_CFG\_WHEEL\_SIZE should be between 2 and 1024"

#endif

#if OS\_TMR\_CFG\_WHEEL\_SIZE > 1024u

#error "OS\_CFG.H, OS\_TMR\_CFG\_WHEEL\_SIZE should be between 2 and 1024"

#endif

#endif

#ifndef OS\_TMR\_CFG\_NAME\_EN

#error "OS\_CFG.H, Missing OS\_TMR\_CFG\_NAME\_EN: Enable Timer names"

#endif

#ifndef OS\_TMR\_CFG\_TICKS\_PER\_SEC

#error "OS\_CFG.H, Missing OS\_TMR\_CFG\_TICKS\_PER\_SEC: Determines the rate at which tiem timer management task will run (Hz)"

#endif

#ifndef OS\_TASK\_TMR\_STK\_SIZE

#error "OS\_CFG.H, Missing OS\_TASK\_TMR\_STK\_SIZE: Determines the size of the Timer Task's stack"

#endif

#endif

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* MISCELLANEOUS

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#ifndef OS\_ARG\_CHK\_EN

#error "OS\_CFG.H, Missing OS\_ARG\_CHK\_EN: Enable (1) or Disable (0) argument checking"

#endif

#ifndef OS\_CPU\_HOOKS\_EN

#error "OS\_CFG.H, Missing OS\_CPU\_HOOKS\_EN: uC/OS-II hooks are found in the processor port files when 1"

#endif

#ifndef OS\_APP\_HOOKS\_EN

#error "OS\_CFG.H, Missing OS\_APP\_HOOKS\_EN: Application-defined hooks are called from the uC/OS-II hooks"

#endif

#ifndef OS\_DEBUG\_EN

#error "OS\_CFG.H, Missing OS\_DEBUG\_EN: Allows you to include variables for debugging or not"

#endif

#ifndef OS\_LOWEST\_PRIO

#error "OS\_CFG.H, Missing OS\_LOWEST\_PRIO: Defines the lowest priority that can be assigned"

#endif

#ifndef OS\_MAX\_EVENTS

#error "OS\_CFG.H, Missing OS\_MAX\_EVENTS: Max. number of event control blocks in your application"

#else

#if OS\_MAX\_EVENTS > 65500u

#error "OS\_CFG.H, OS\_MAX\_EVENTS must be <= 65500"

#endif

#endif

#ifndef OS\_SCHED\_LOCK\_EN

#error "OS\_CFG.H, Missing OS\_SCHED\_LOCK\_EN: Include code for OSSchedLock() and OSSchedUnlock()"

#endif

#ifndef OS\_EVENT\_MULTI\_EN

#error "OS\_CFG.H, Missing OS\_EVENT\_MULTI\_EN: Include code for OSEventPendMulti()"

#endif

#ifndef OS\_TASK\_PROFILE\_EN

#error "OS\_CFG.H, Missing OS\_TASK\_PROFILE\_EN: Include data structure for run-time task profiling"

#endif

#ifndef OS\_TASK\_SW\_HOOK\_EN

#error "OS\_CFG.H, Missing OS\_TASK\_SW\_HOOK\_EN: Allows you to include the code for OSTaskSwHook() or not"

#endif

#ifndef OS\_TICK\_STEP\_EN

#error "OS\_CFG.H, Missing OS\_TICK\_STEP\_EN: Allows to 'step' one tick at a time with uC/OS-View"

#endif

#ifndef OS\_TIME\_TICK\_HOOK\_EN

#error "OS\_CFG.H, Missing OS\_TIME\_TICK\_HOOK\_EN: Allows you to include the code for OSTimeTickHook() or not"

#endif

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* SAFETY CRITICAL USE

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#ifdef SAFETY\_CRITICAL\_RELEASE

#if OS\_ARG\_CHK\_EN < 1u

#error "OS\_CFG.H, OS\_ARG\_CHK\_EN must be enabled for safety-critical release code"

#endif

#if OS\_APP\_HOOKS\_EN > 0u

#error "OS\_CFG.H, OS\_APP\_HOOKS\_EN must be disabled for safety-critical release code"

#endif

#if OS\_DEBUG\_EN > 0u

#error "OS\_CFG.H, OS\_DEBUG\_EN must be disabled for safety-critical release code"

#endif

#ifdef CANTATA

#error "OS\_CFG.H, CANTATA must be disabled for safety-critical release code"

#endif

#ifdef OS\_SCHED\_LOCK\_EN

#error "OS\_CFG.H, OS\_SCHED\_LOCK\_EN must be disabled for safety-critical release code"

#endif

#ifdef VSC\_VALIDATION\_MODE

#error "OS\_CFG.H, VSC\_VALIDATION\_MODE must be disabled for safety-critical release code"

#endif

#if OS\_TASK\_STAT\_EN > 0u

#error "OS\_CFG.H, OS\_TASK\_STAT\_EN must be disabled for safety-critical release code"

#endif

#if OS\_TICK\_STEP\_EN > 0u

#error "OS\_CFG.H, OS\_TICK\_STEP\_EN must be disabled for safety-critical release code"

#endif

#if OS\_FLAG\_EN > 0u

#if OS\_FLAG\_DEL\_EN > 0

#error "OS\_CFG.H, OS\_FLAG\_DEL\_EN must be disabled for safety-critical release code"

#endif

#endif

#if OS\_MBOX\_EN > 0u

#if OS\_MBOX\_DEL\_EN > 0u

#error "OS\_CFG.H, OS\_MBOX\_DEL\_EN must be disabled for safety-critical release code"

#endif

#endif

#if OS\_MUTEX\_EN > 0u

#if OS\_MUTEX\_DEL\_EN > 0u

#error "OS\_CFG.H, OS\_MUTEX\_DEL\_EN must be disabled for safety-critical release code"

#endif

#endif

#if OS\_Q\_EN > 0u

#if OS\_Q\_DEL\_EN > 0u

#error "OS\_CFG.H, OS\_Q\_DEL\_EN must be disabled for safety-critical release code"

#endif

#endif

#if OS\_SEM\_EN > 0u

#if OS\_SEM\_DEL\_EN > 0u

#error "OS\_CFG.H, OS\_SEM\_DEL\_EN must be disabled for safety-critical release code"

#endif

#endif

#if OS\_TASK\_EN > 0u

#if OS\_TASK\_DEL\_EN > 0u

#error "OS\_CFG.H, OS\_TASK\_DEL\_EN must be disabled for safety-critical release code"

#endif

#endif

#if OS\_CRITICAL\_METHOD != 3u

#error "OS\_CPU.H, OS\_CRITICAL\_METHOD must be type 3 for safety-critical release code"

#endif

#endif /\* ------------------------ SAFETY\_CRITICAL\_RELEASE ------------------------ \*/

#ifdef \_\_cplusplus

}

#endif

#endif