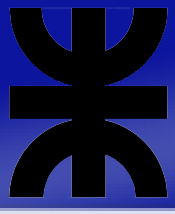


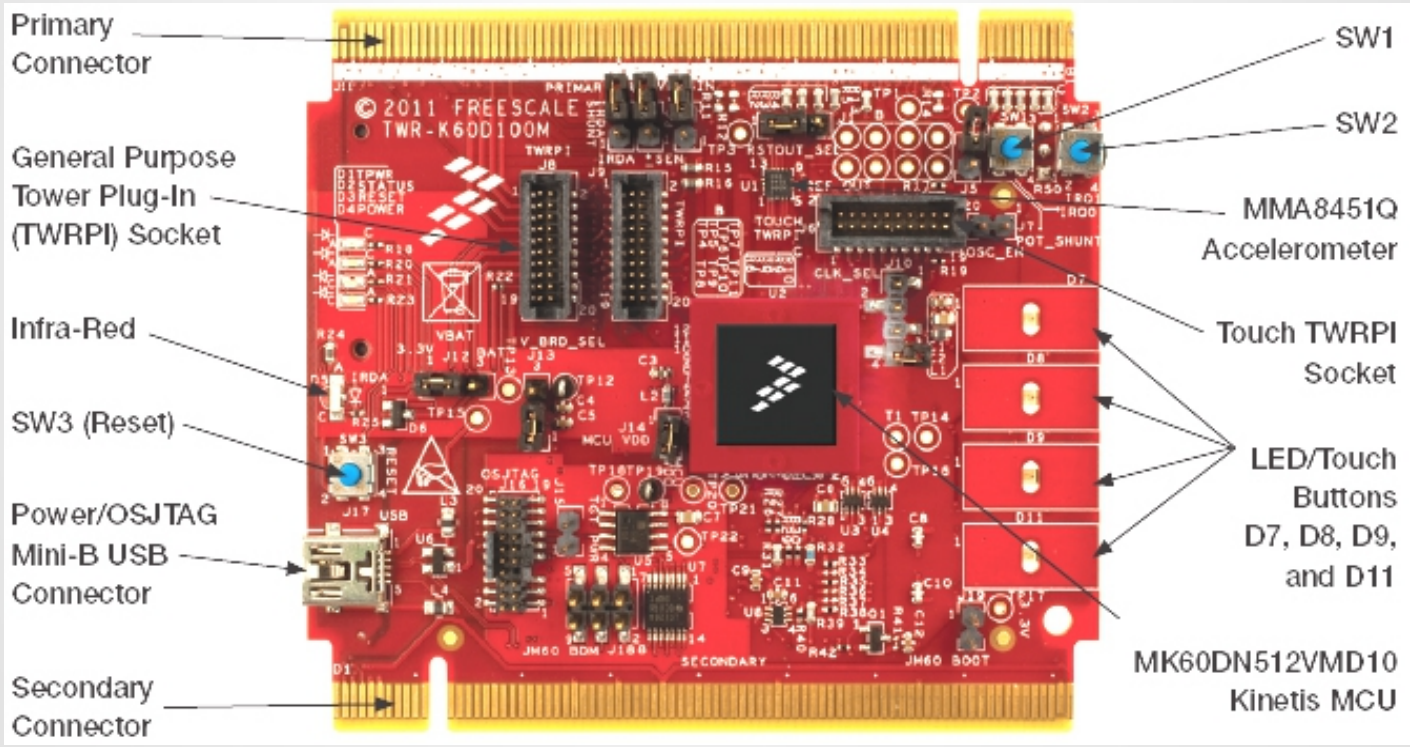
MQX RTOS
Módulo TWR-K60D100M
Freescale

Módulo TWR-K60D100M

TWR-K60D100M

El TWR-K60D100M es un módulo basado en los dispositivos kinetis, de la familia de microcontroladores K60.

Cuenta con un microcontrolador de baja potencia Kinetis K60 basado en la arquitectura ARM ®.



MQX RTOS

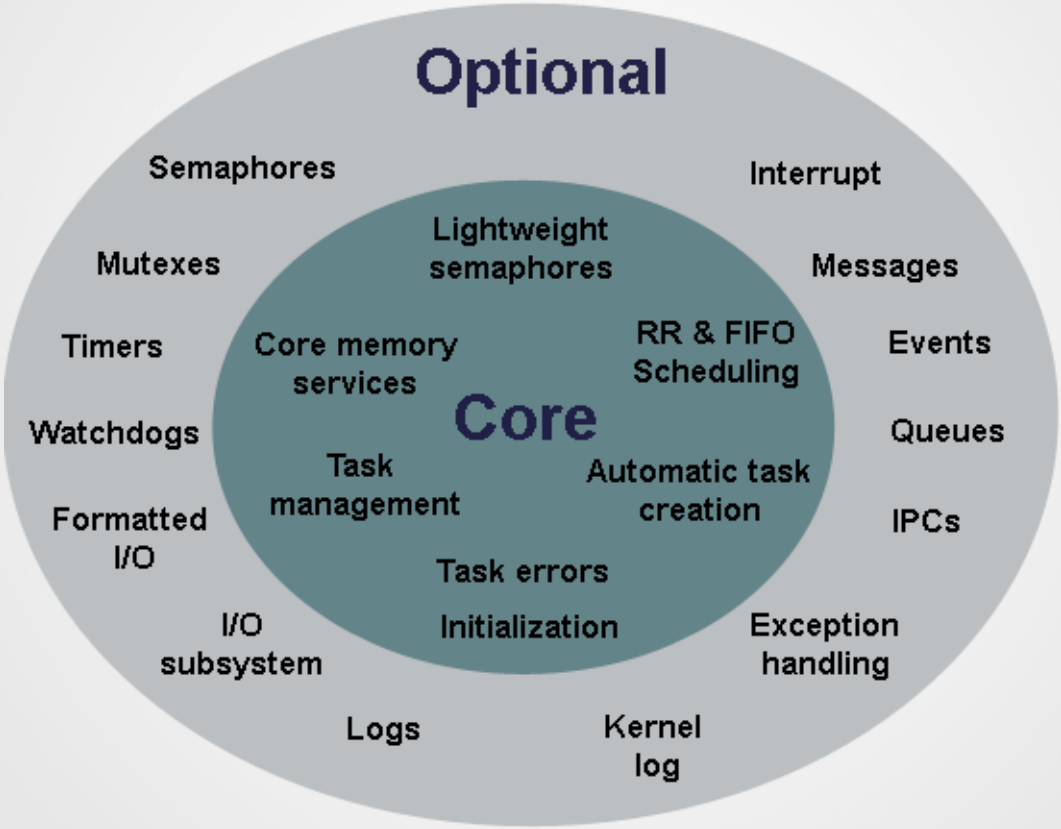
MQX es el sistema operativo de tiempo real que Freescale adoptó para su plataforma de placas kinetis.

Esta diseñado para plataformas de un solo procesador o multiprocesador.

MQX RTOS

Organización de MQX RTOS

Consta de núcleo (no opcional) y de componentes opcionales.



Planificador MQX RTOS

El planificador cumple con POSIX.4. Ofrece estas políticas de planificación:

• FIFO (basado en prioridades) la tarea activa es la de mayor prioridad que ha estado más tiempo en la lista.

• Round robin RR (con intervalos de tiempo). Es como una FIFO pero con ranuras de tiempo, es decir, cada tarea tiene una cantidad máxima de tiempo (ticks) durante el cual puede estar activa.

FIFO es la planificación por defecto de MQX RTOS.

Planificador MQX RTOS

Con el planificador FIFO, la tarea activa se ejecuta, hasta que se presenta cualquiera de las siguientes situaciones:

• La tarea activa abandona voluntariamente el uso del procesador, ya que llama a una función que bloquea la tarea. Por ejemplo _lwsem_wait().

• Se presenta una interrupción que tiene mayor prioridad que la tarea activa.

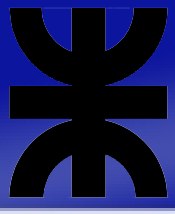
• Una tarea que tiene mayor prioridad que la tarea activa, está lista.

Inicialización MQX RTOS

La aplicación se inicia cuando _mqx() se ejecuta. La función inicializa el hardware y comienza MQX. Cuando MQX se inicia, crea tareas que la aplicación ha definido como tareas de inicio automático.

Una aplicación también puede crear, gestionar y terminar las tareas. La aplicación puede cambiar dinámicamente los atributos de cualquier tarea. MQX libera los recursos de trabajo, cuando una tarea finaliza.

Para cada tarea se puede especificar:
- Una función de salida, que MQX llama cuando se termina la tarea.
- Un controlador de excepciones, que llama MQX si se produce una excepción, mientras que la tarea está activa.



MQX RTOS

Inicio MQX RTOS

MQX RTOS se inicia con _mqx(), uno de los argumentos es la estructura de inicialización MQX RTOS. Basándose en los valores en la estructura, MQX RTOS hace lo siguiente:

- Establece e inicializa los datos que MQX RTOS utiliza internamente (colas de lista, la pila de interrupciones).
- Se inicia el hardware.
- Se inician los temporizadores.
- Se establece el valor de intervalo de tiempo predeterminado.
- Se crea la tarea de inicio, que estará activa si no hay otra tarea de mayor prioridad lista.
- Se crean tareas que la lista de plantillas de tareas define como tareas de arranque automático.
- Se inicia de planificación de las tareas.

Plantilla de la lista de tareas TASK_TEMPLATE_STRUCT

En la inicialización, MQX RTOS crea una petición para cada tarea, que se definió como una tarea de inicio automático en la plantilla.
Mientras se ejecuta una tarea, ésta puede crear otras tareas, que estén definidas en la plantilla de la lista de tareas, sin inicio automático.

```
#include <mqx.h>
const TASK_TEMPLATE_STRUCT  MQX_template_list[] =
{
//Task Index,   Function,   Stack,  Priority,  Name,          Attributes,     Param, Time Slice
{ MAIN_TASK, Main_task, 1500,     6,       "main",MQX_AUTO_START_TASK,  0,    0 },
{    TAREA1,      tarea_1,    1500,     7,     "tarea1",              0,              0,     0 },
{    TAREA2,      tarea_2,    1500,     7,     "tarea2",              0,              0,     0 },
{        0,              0,          0,       0,          0,                0,              0,     0 },
};
void Main_task(uint32_t initial_data) {
   printf("Main, tarea de mayor prioridad, es la primera que va a ejecutarse\n");
   _task_block();    }
```

MQX RTOS

Main_task

Es la tarea de prioridad 6, es la tarea de mas alta prioridad en la lista.

Es una tarea de inicio automático (MQX_AUTO_START_TASK).

La aplicación define el índice de plantilla de tarea (MAIN_TASK).

La función Main_task,() es el punto de código de entrada de la tarea.

El tamaño de la pila es 0x1500.

Creación de tareas

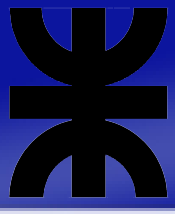
Cualquier tarea puede crear otra tarea llamando a la función _task_create()

_task_id ***_task_create***(_processor_number processor_number,_mqx_uint template_index, uint32_t parameter)

task_id1= _task_create(0, TAREA_PR, (uint32_t)strings1)

processor_number: número del procesador , 0 procesador local.
template_index: Índice de la plantilla de la lista de tarea.
parameter: normalmente se utiliza para proporcionar información de inicialización para la tarea creada.

Devuelve el ID de la tarea creada en caso de éxito o MQX_NULL_TASK_ID en caso de error.



MQX RTOS

Terminación de Tareas

Una tarea puede terminarse a si misma o cualquier otra tarea, si conoce el ID de la tarea.
Una aplicación puede terminar una tarea inmediatamente con la función _task_destroy() o terminar con elegancia con la función _task_abort().

_mqx_uint **_task_destroy**(_task_id task_id)

Devuelve MQX_OK en caso de éxito o MQX_INVALID_TASK_ID en caso de error (task_id no representa una tarea válida)

_mqx_uint **_task_abort**(_task_id task_id)

Devuelve MQX_OK en caso de éxito o MQX_INVALID_TASK_ID en caso de error (task_id no representa una tarea válida)

Bloqueo de tareas

La funcion ***_task_block()*** bloquea la tarea activa.

La función elimina la tarea activa de la cola de listos y setea el bit BLOQUEADO en el registro STATE del descriptor de tarea.

Función _task_ready

Hace que una tarea bloqueada pase a estado lista y quede en la cola de tareas listas:

void ***_task_ready***(void *td_ptr)
td_ptr [IN]: Puntero al descriptor de tarea

Códigos de error:
MQX_INVALID_TASK_ID: task_id no es un descriptor válido
MQX_INVALID_TASK_STATE: La tarea ya está en la cola de tareas listas.

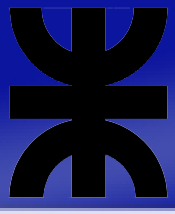
Función _task_get_td()

Devuelve el puntero del descriptor de la tarea

void ***_task_get_td***(_task_id task_id)

task_id [IN] es el ID de la tarea

Devuelve
- El puntero descriptor de la tarea si tuvo éxito
- NULL en caso de error



MQX RTOS

Función _mqx_exit(0)

Termina la aplicación MQX RTOS y vuelve al entorno que comenzó la aplicación.

void ***_mqx_exit***(_mqx_uint error_code);

error_code: código de error para la función _mqx().

Módulo TWR-K60D100M

TWR-K60D100M

TWR-K60D100M solo soporta la política de planificación FIFO.

Bibliografía

Freescale MQX™ RTOS Reference Manual. 2014.
Freescale MQX™ RTOS User's Guide. 2014.
TWR-K60D100M Tower Module User's Manual. 2011.