Freescale **MQX™ RTOS** Introduction

APF-DES-T1635

Stanley Huang | Sr. MCU FAE

AUG.2015

freescale™

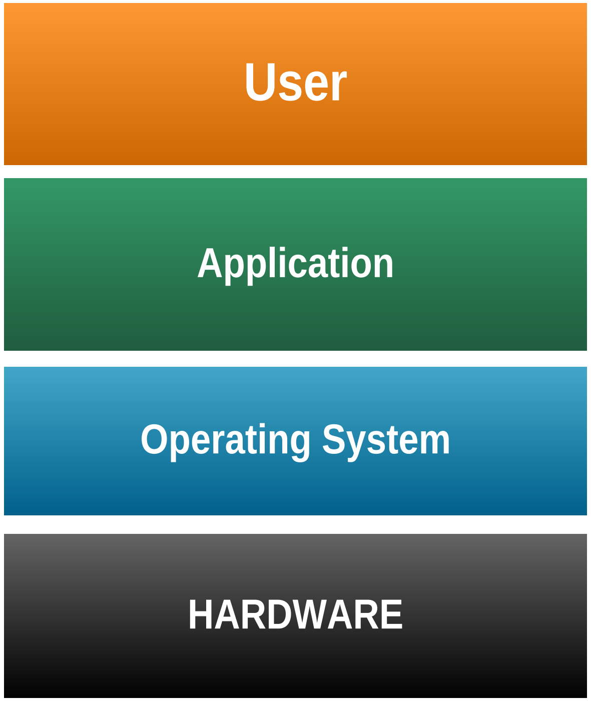
External Use

Agenda

- **What is an RTOS?**
- **MQX Basics: Tasks**
- **MQX Basics: Scheduling**
- **MQX Basics: Task Synchronization**
  - Semaphores
  - Events and Messages
- **MQX Intermediate**
  - Libraries
  - Interrupts
  - BSP
- **Additional Resources**
- **Review**

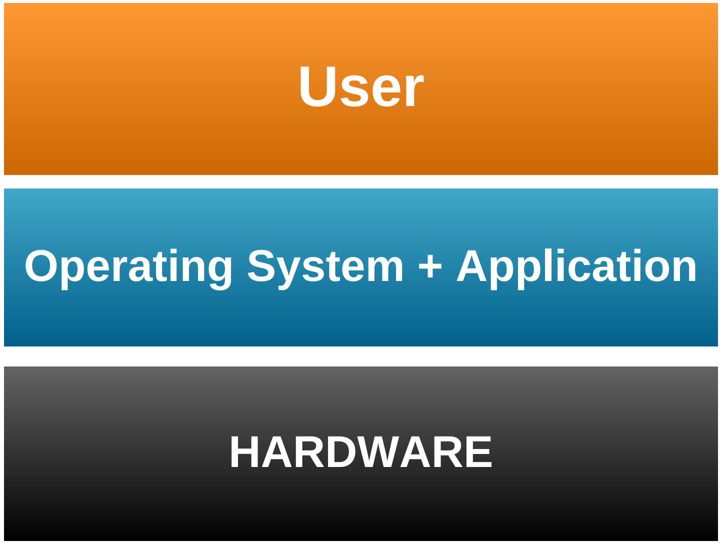
Operating Systems

- **The term “operating system” can be used to describe the collection of software that manages a system’s hardware resources**
- **This software might include a file system module, a GUI and other components**
- **Often times, a “kernel” is understood to be a subset of such a collection**
- **Characteristics**
  - ❑ Resource management
  - ❑ Interface between application and hardware
  - ❑ Library of functions for the application



Real Time Operating Systems

- **Fusion of the application and the OS to one unit.**
- **Code of the OS and the application mostly reside in ROM.**
- **A real-time operating system (RTOS) manages the time of a microprocessor or microcontroller.**
- **Features of an RTOS:**
  - ❑Allows multi-tasking
  - ❑Scheduling of the tasks with priorities
  - ❑Synchronization of the resource access
  - ❑Inter-task communication
  - ❑Time predictable
  - ❑Interrupt handling



User

Operating System + Application

HARDWARE

Why use an RTOS?

- Plan to use drivers that are available with an RTOS
- Would like to spend your time developing application code and not creating or maintaining a scheduling system
- Multi-thread support with synchronization
- Portability of application code to other CPUs
- Resource handling
- Add new features without affecting higher priority functions
- Support for upper layer protocols such as:
  - ❑ TCP/IP, USB, Flash Systems, Web Servers,
  - ❑ CAN protocols, Embedded GUI, SSL, SNMP

Freescale MQX

- We will be using Freescale MQX to demonstrate these RTOS concepts.

- Freescale MQX Software can be downloaded:
  - ❑ http://www.freescale.com/mqx

- Default Freescale MQX folder:
  - ❑ C:\Freescale\Freescale_MQX_4_2

Freescale MQX RTOS 4.2

Released
May 6, 2015

New Board Support Packages Added

- ***Latest Kinetis K-Series MCUs such as K22, K24, K65***

New Features and Updates

- ***MFS Updates*** – Multiple read/write support, improvements in directory & file search/seek, and general speed and code-size optimizations
- ***New USB Stack*** *(Select MCUs only)* – Simplified API, improved performance, reduced memory footprint, composite device support, more robust Hub support
- ***Driver Updates and Other Features***
  - New TLSF memory allocator (optional) for higher determinism / lower fragmentation
- ***RTCS Updates***
  - New Features – Websockets, Link-Local Multicast Name Resolution (LLMNR), GPRS modem example, iperf performance example, Secure webserver (HTTPs) example
  - Sockets API updated for improved Berkeley Sockets compatibility
  - New IPv6 protocols – DHCPv6 Client, Telnet Client, TFTP Client/Server, FTP Client [add-on for purchase]
  - CyaSSL TLS/SSL [free add-on for evaluation]

Freescale MQX RTOS 4.2

- **22 *Complimentary* BSPs** covering all *Kinetis K-Series and Vybrid*
- **Windows & Linux installers**
- **Tools support**
  - KDS, CW, IAR, Keil, ARM® DS-5™, and GNU tools for ARM® (Windows and Linux)

Numerous additional BSPs for legacy devices available free of charge in earlier MQX versions, or for purchase

PLATFORM	MQX 4.2
VYBRID	
TWR-VF65GS10 (M4&A5)	✓
EVB-VF522R3 (M4&A5)	✓
KINETIS	
TWR-K20D50M	✓
TWR-K20D72M	✓
TWR-K21D50M	✓
TWR-K21F120M	✓
FRDM-K22F ***NEW***	✓
TWR-K22F120M ***NEW***	✓
TWR-K24F120M ***NEW***	✓
TWR-K40X256	✓
TWR-K40D100M	✓
KWIKSTICK (K40)	✓
TWR-K53N512	✓
TWR-K60D100M	✓
TWR-K60F120M	✓
TWR-K60N512	✓
TWR-K64F120M	✓
FRDM-K64F	✓
TWR-K65F180M ***NEW***	✓
TWR-K70120M	✓

MQX RTOS for Kinetis KDS 3.0.0

Released
May 6, 2015

New Features and Updates

- ***General Updates***
  - RAM footprint optimizations
  - New MQX Lite application examples

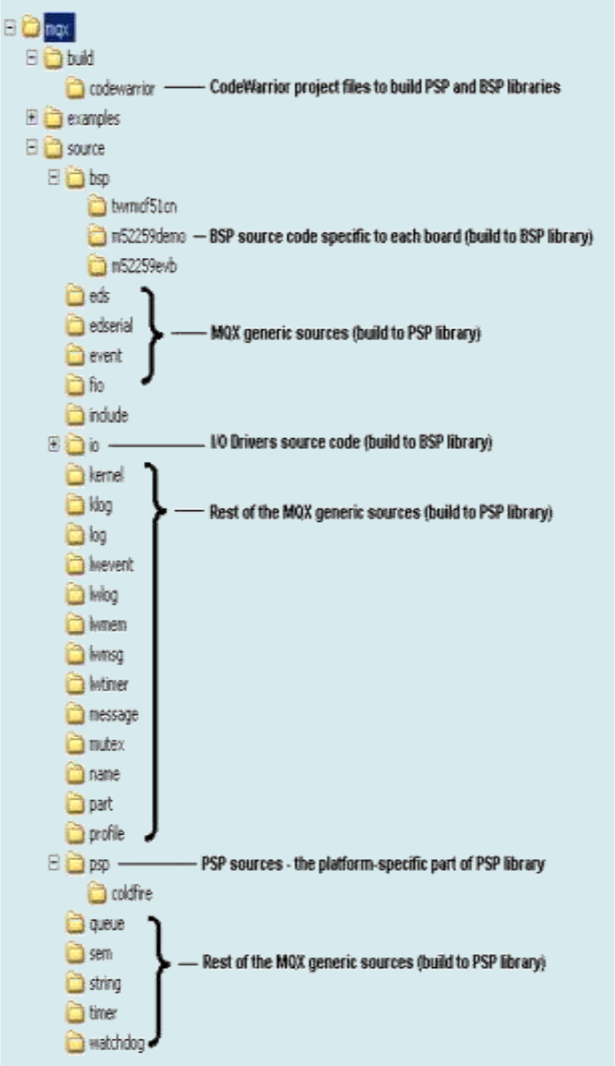
- ***MFS Updates*** – Multiple read/write support, improvements in directory & file search/seek, and general speed and code-size optimizations

- ***RTCS Updates***
  - New Features – Websockets, Link-Local Multicast Name Resolution (LLMNR), Secure webserver (HTTPs) example
  - Sockets API updated for improved Berkeley Sockets compatibility
  - New IPv6 protocols – DHCPv6 Client, Telnet Client, TFTP Client/Server, FTP Client [add-on for purchase]
  - CyaSSL TLS/SSL [free add-on for evaluation]

MQX Directory Structure

- **Described in the MQX Release Notes**

- **Folders are:**
  - ❑ config
  - ❑ demo
  - ❑ doc
  - ❑ lib
  - ❑ mqx
  - ❑ tools
  - ❑ And then the RTCS, USB, and MFS stacks

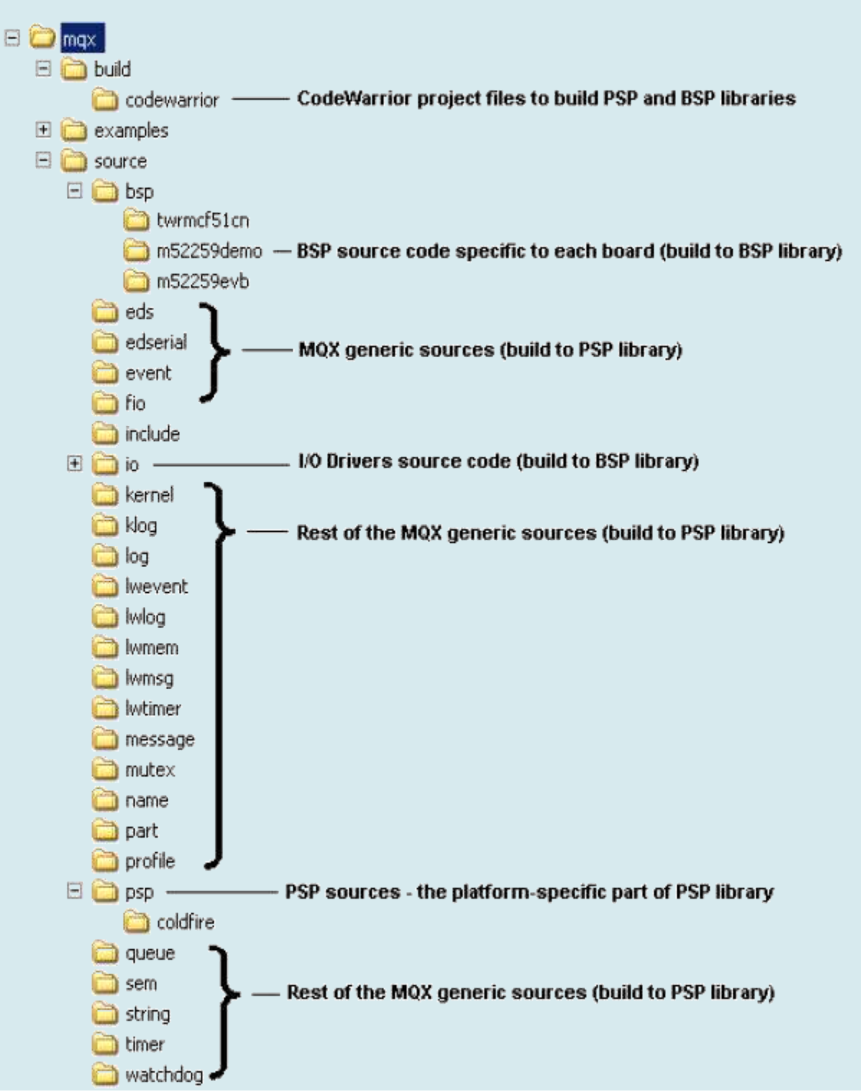


```
⊟ mqx
  ⊟ build
      codewarrior —— CodeWarrior project files to build PSP and BSP libraries
  ⊞ examples
  ⊟ source
    ⊟ bsp
        twrmcf51cn
        m52259demo — BSP source code specific to each board (build to BSP library)
        m52259evb
      eds      }
      edserial }
      event    }—— MQX generic sources (build to PSP library)
      fio      }
      include
    ⊞ io —————— I/O Drivers source code (build to BSP library)
      kernel   }
      klog     }—— Rest of the MQX generic sources (build to PSP library)
      log      }
      lwevent  }
      lwlog    }
      lwmem    }
      lwmsg    }
      lwtimer  }
      message  }
      mutex    }
      name     }
      part     }
      profile  }
    ⊟ psp —————— PSP sources - the platform-specific part of PSP library
        coldfire
      queue    }
      sem      }
      string   }— Rest of the MQX generic sources (build to PSP library)
      timer    }
      watchdog }
```

MQX Directory Structure (Cont.)

- **The “mqx” directory is heart of MQX**

- **Folders are:**
  - ❑ build
  - ❑ examples
  - ❑ source
    - ▪ bsp
    - ▪ io
    - ▪ psp
    - ▪ MQX API source

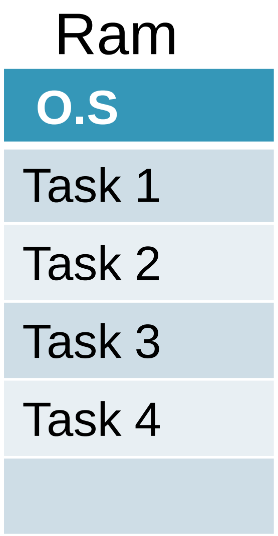


```
⊟ mqx
  ⊟ build
      codewarrior ——— CodeWarrior project files to build PSP and BSP libraries
  ⊞ examples
  ⊟ source
    ⊟ bsp
        twrmcf51cn
        m52259demo — BSP source code specific to each board (build to BSP library)
        m52259evb
      eds       }
      edserial  }
      event     } ——— MQX generic sources (build to PSP library)
      fio       }
      include
    ⊞ io ——————————— I/O Drivers source code (build to BSP library)
      kernel    }
      klog      } ——— Rest of the MQX generic sources (build to PSP library)
      log       }
      lwevent   }
      lwlog     }
      lwmem     }
      lwmsg     }
      lwtimer   }
      message   }
      mutex     }
      name      }
      part      }
      profile   }
    ⊟ psp —————————— PSP sources - the platform-specific part of PSP library
        coldfire
      queue     }
      sem       }
      string    } ——— Rest of the MQX generic sources (build to PSP library)
      timer     }
      watchdog  }
```

Agenda

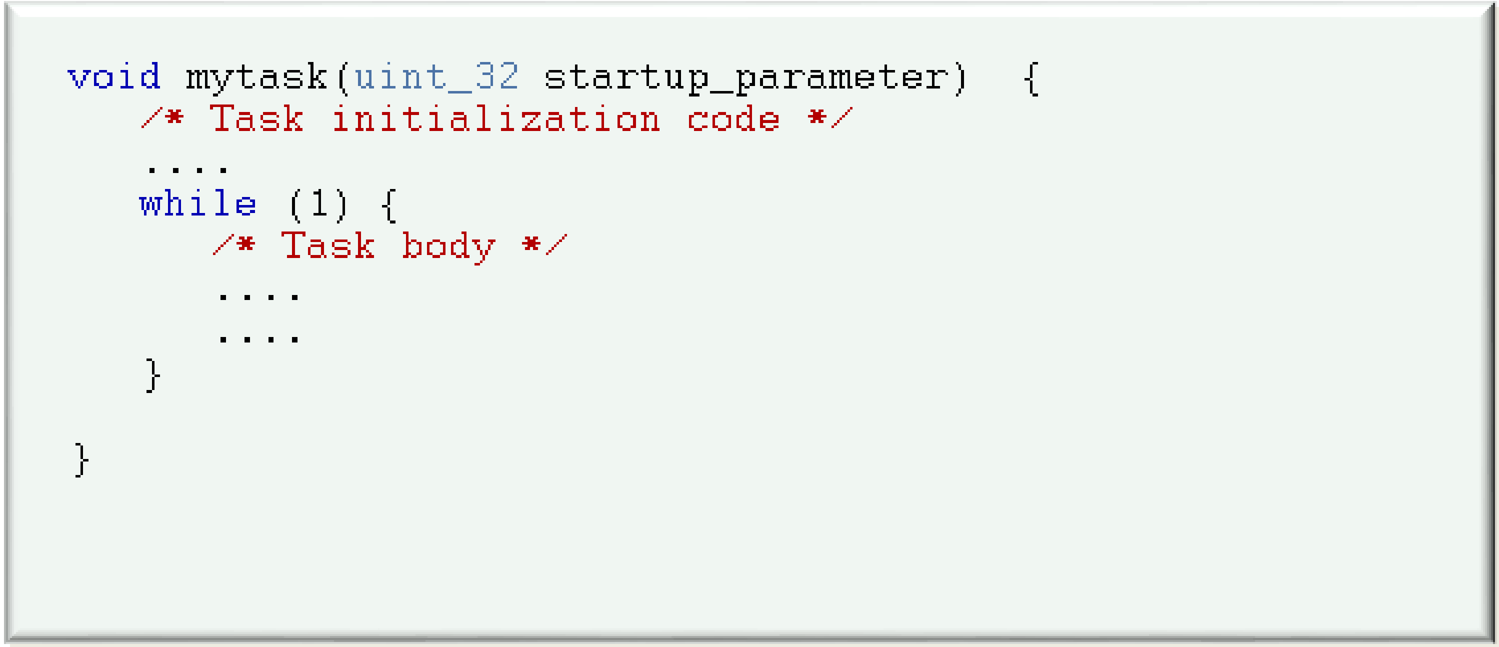
- **What is an RTOS?**
- **MQX Basics: Tasks**
- **MQX Basics: Scheduling**
- **MQX Basics: Task Synchronization**
  - Semaphores
  - Events and Messages
- **MQX Intermediate**
  - Libraries
  - Interrupts
  - BSP
- **Additional Resources**
- **Review**

MQX RTOS Tasks



- A system consists of multiple tasks
- Tasks take turns running
- Only one task is active (has the processor) at any given time
- MQX manages how the tasks share the processor (context switching)
- Task Context
  - Data structure stored for each task, including registers and a list of owned resources

Typical Task Coding Structure

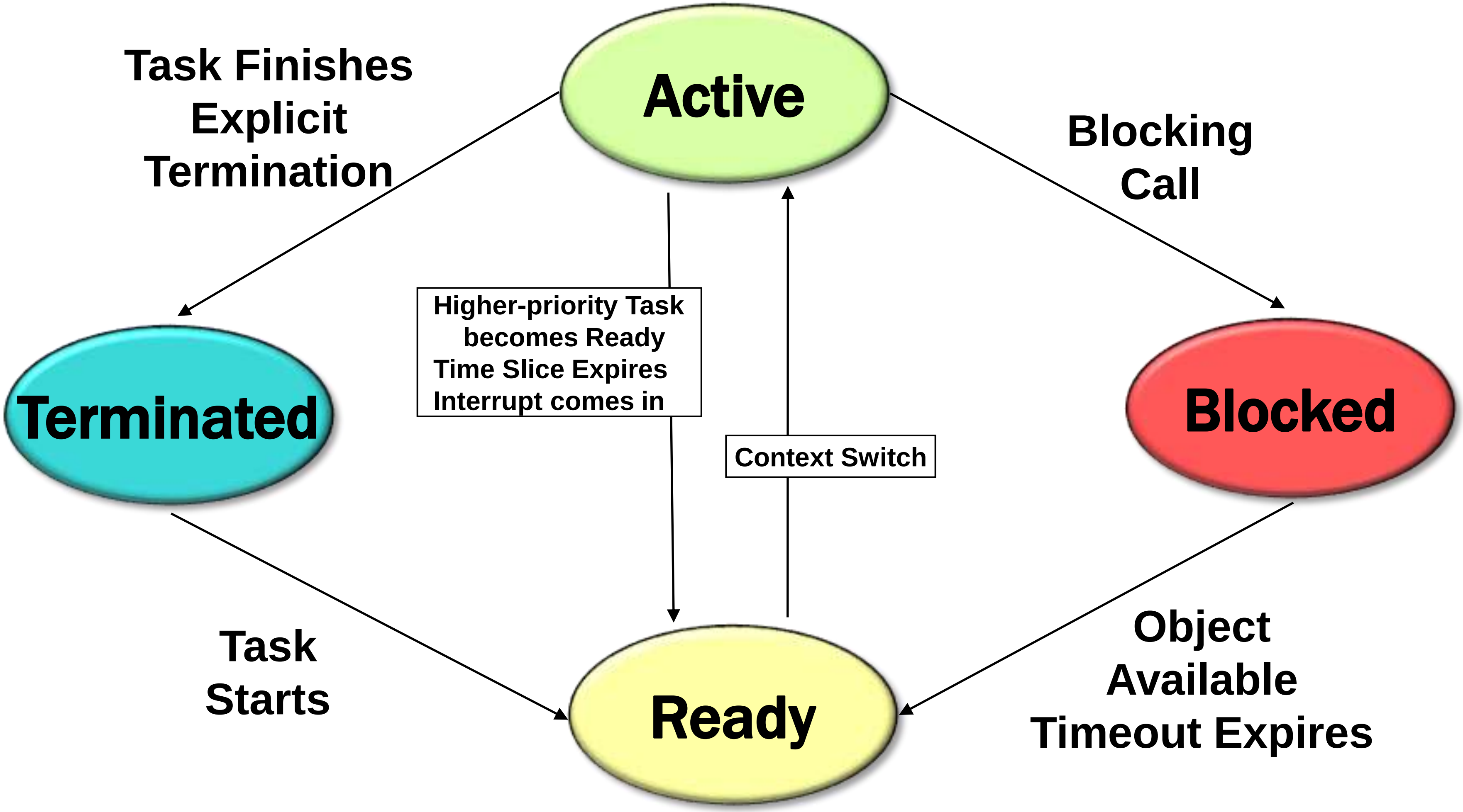


```
void mytask(uint_32 startup_parameter)  {
    /* Task initialization code */
    ....
    while (1) {
        /* Task body */
        ....
        ....
    }

}
```

Task States

- **A task is in one of these logical states:**
  - ❑ blocked
    - the task is blocked and therefore not ready
    - it’s waiting for a condition to be true
  - ❑ active
    - the task is ready and is running because it’s the highest-priority ready task
  - ❑ ready
    - the task is ready, but it’s not running because it isn’t the highest-priority ready task
  - ❑ terminated
    - the task has finished all its work, or was explicitly destroyed

Task States


Task Finishes
Explicit
Termination
Active
Blocking
Call
Higher-priority Task
becomes Ready
Time Slice Expires
Interrupt comes in
Terminated
Blocked
Context Switch
Task
Starts
Ready
Object
Available
Timeout Expires

Agenda

- **What is an RTOS?**
- **MQX Basics: Tasks**
- **MQX Basics: Scheduling**
- **MQX Basics: Task Synchronization**
  - Semaphores
  - Events and Messages
- **MQX Intermediate**
  - Libraries
  - Interrupts
  - BSP
- **Additional Resources**
- **Review**

MQX Basics: Scheduling

Priorities

8

9

10

11

idle

- Priorities run from 0 to N
  - Priority 0 means interrupts disabled, 1 is most important task
- N(11) is set by the highest priority number in the MQX_Template_List
  - Idle task runs at N+1
- MQX creates one ready queue for each priority up to the lowest priority (highest number)
  - So must make sure priorities are consecutive
- Able to change priority of a task during runtime
  - **`_task_set_priority()`**
- Any tasks at priority below 7 means it masks certain levels of interrupts. So user tasks should start at 8 or above.

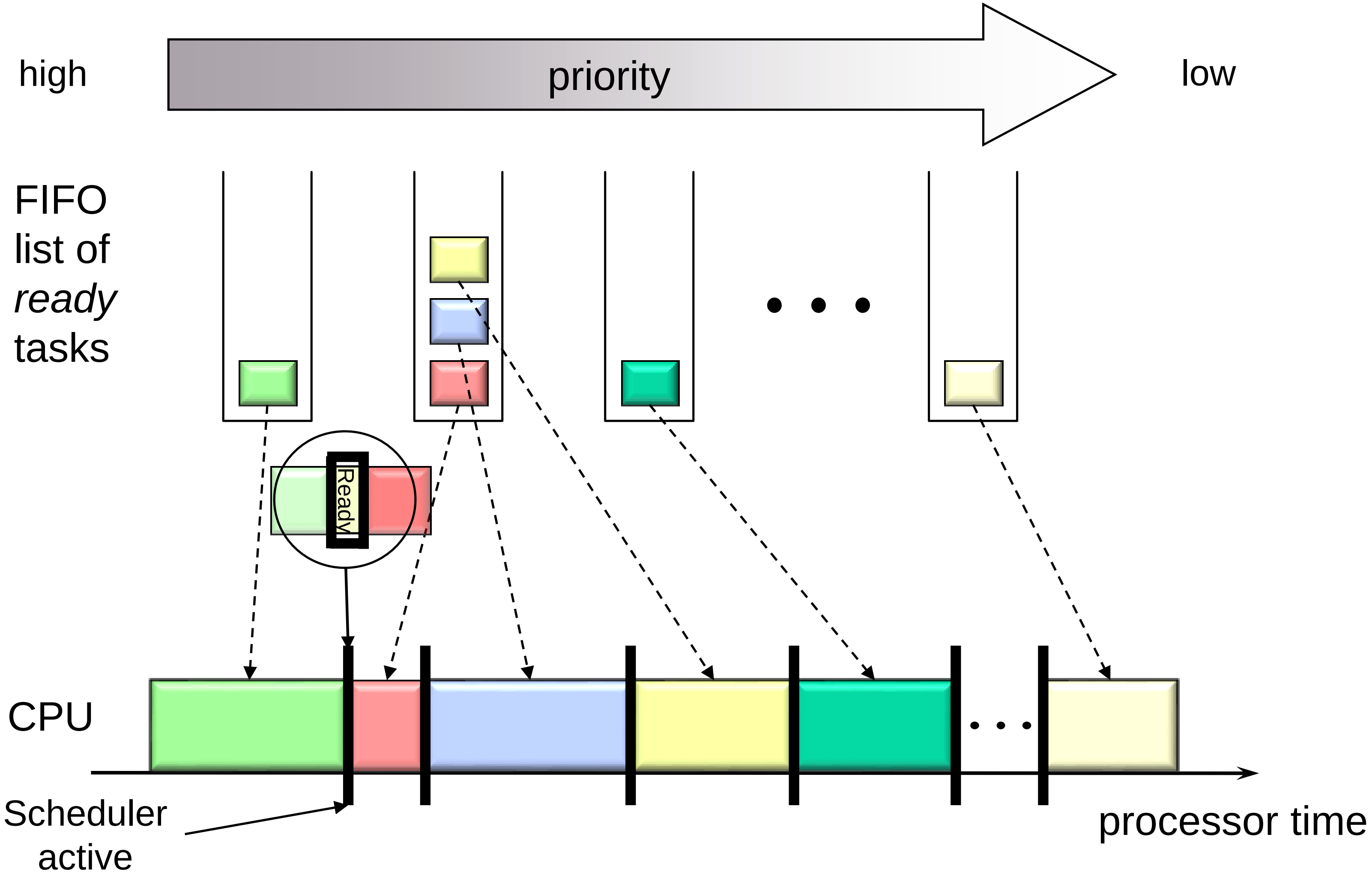
freescale™

Scheduler

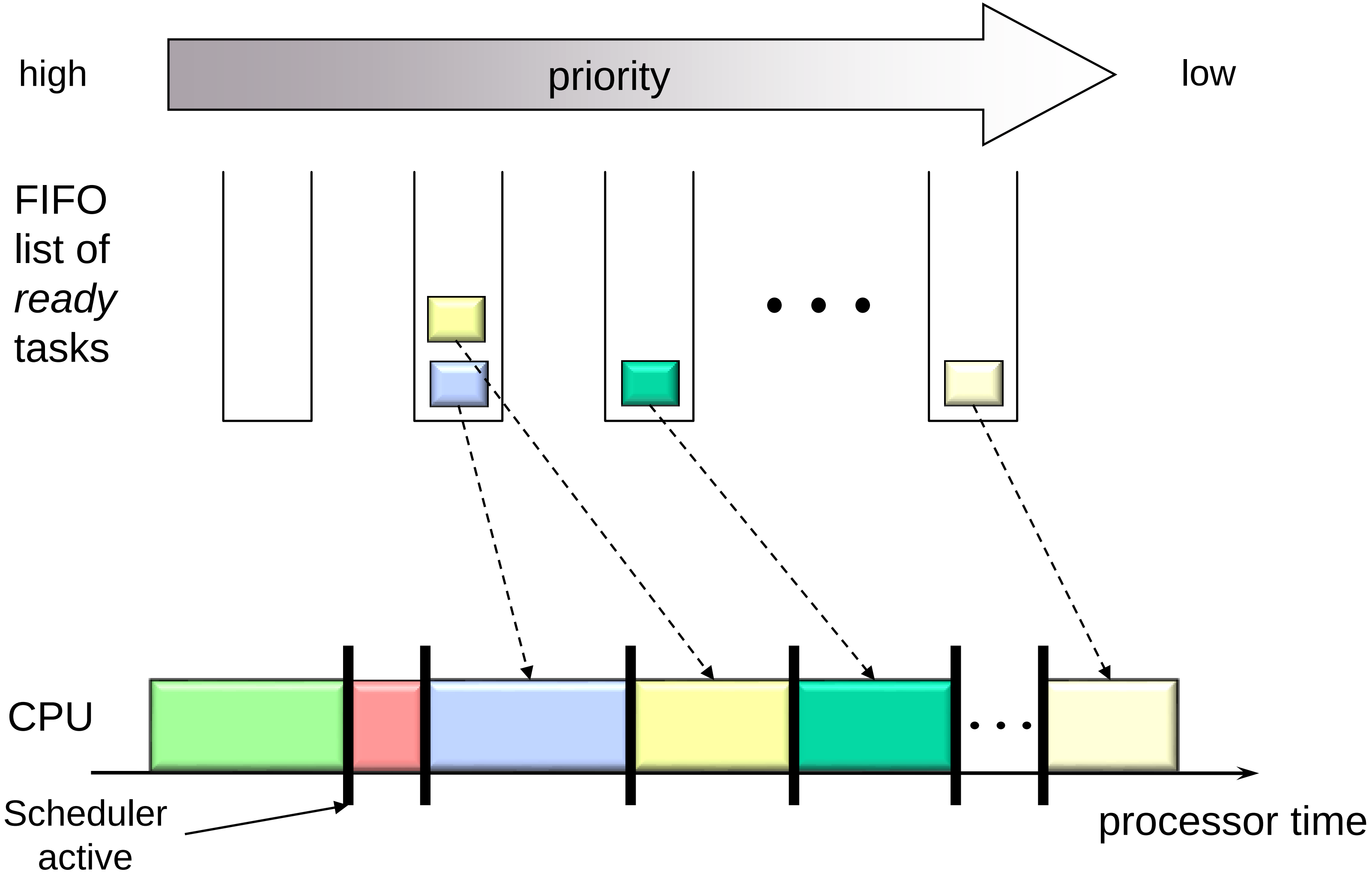
- There are several scheduling policies that MQX supports.

- **Common Scheduling Configurations:**
  - ❑ FIFO (also called priority-based preemptive)
    - ▪ The active task is the highest-priority task that has been ready the longest
  - ❑ Round Robin
    - ▪ The active task is the highest-priority task that has been ready the longest without consuming its time slice

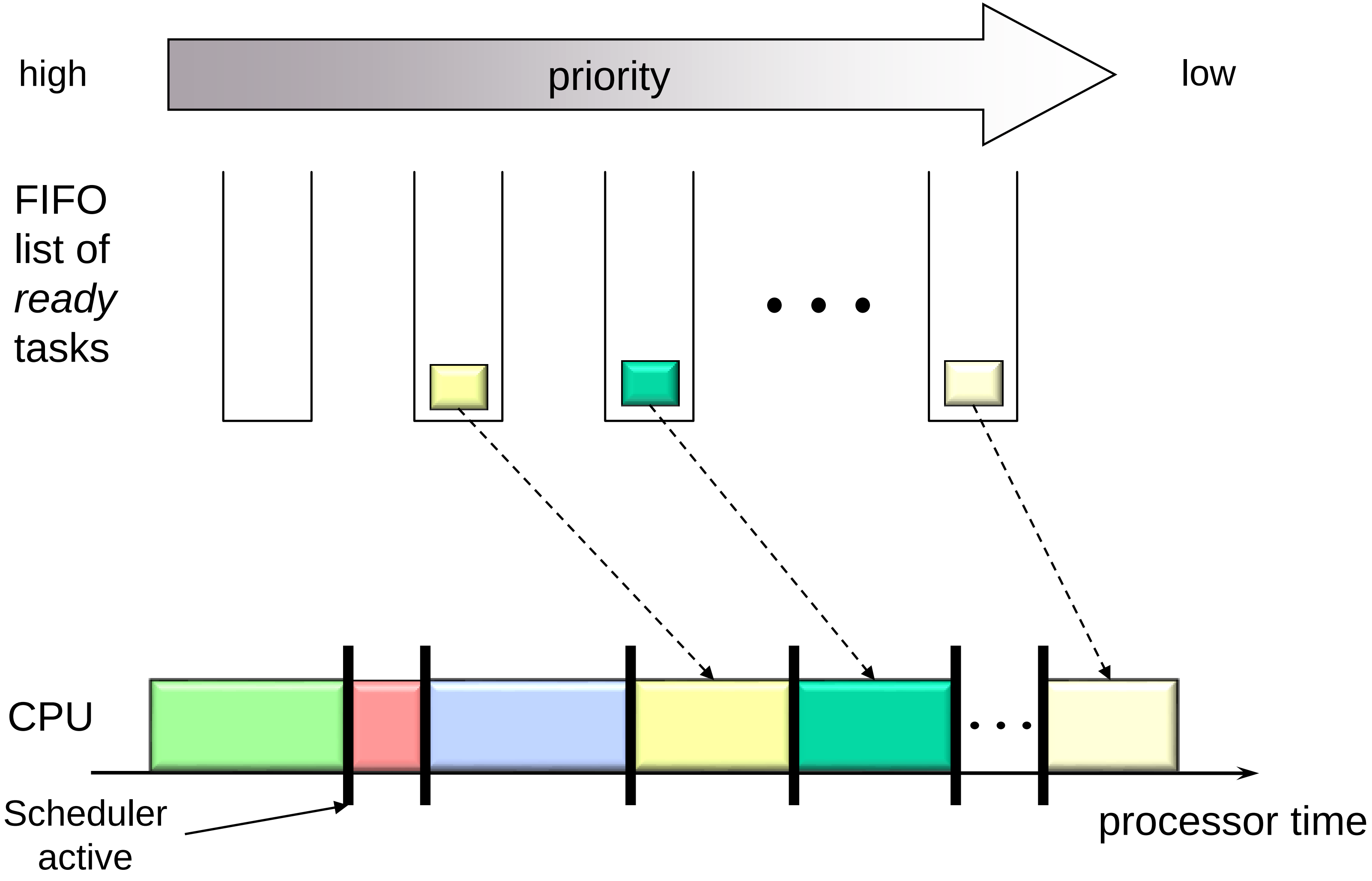
Priority Based FIFO Scheduling



Priority Based FIFO Scheduling



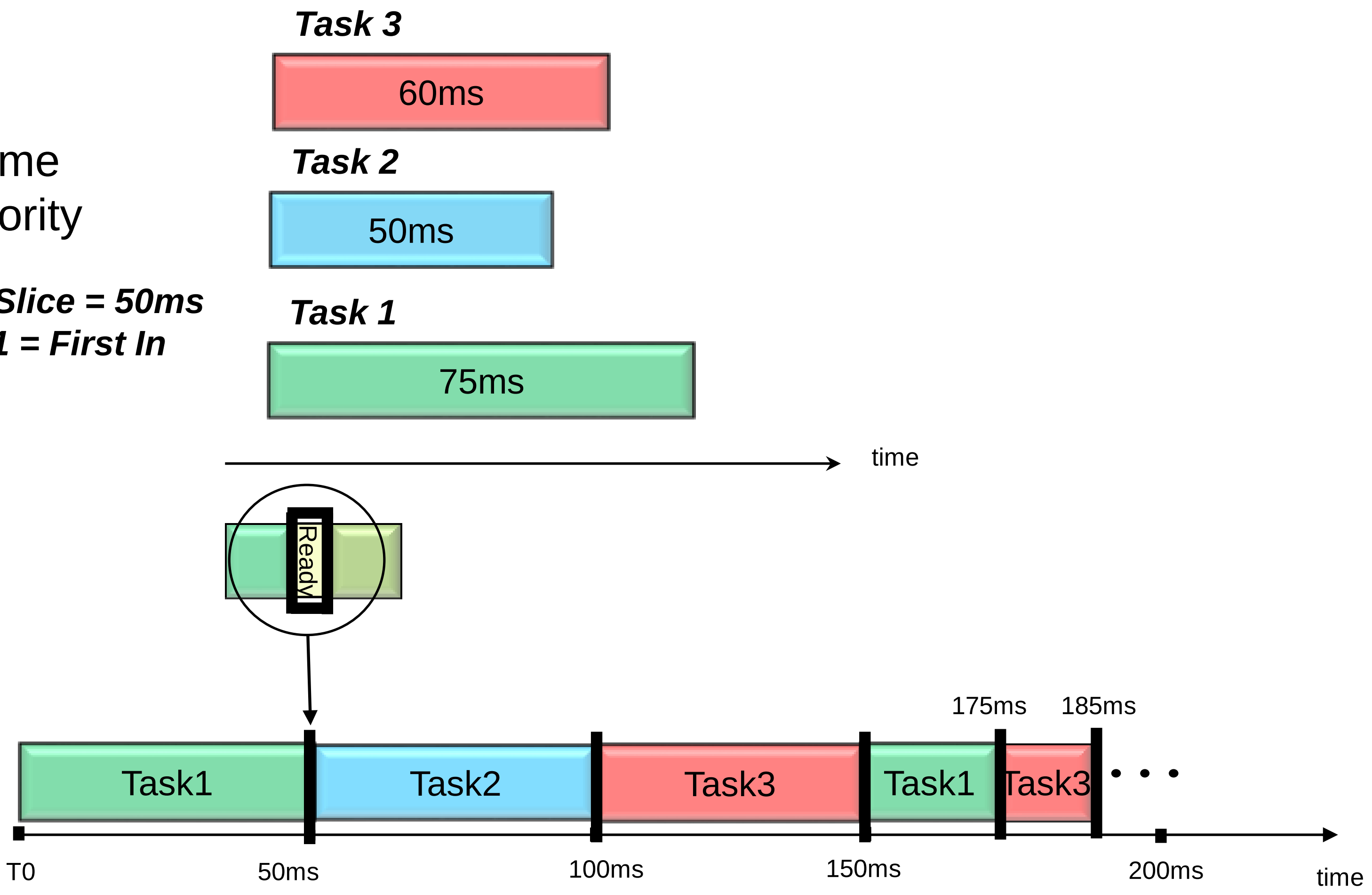
Priority Based FIFO Scheduling



Round-Robin Scheduling

Same
Priority

Time Slice = 50ms
Task 1 = First In

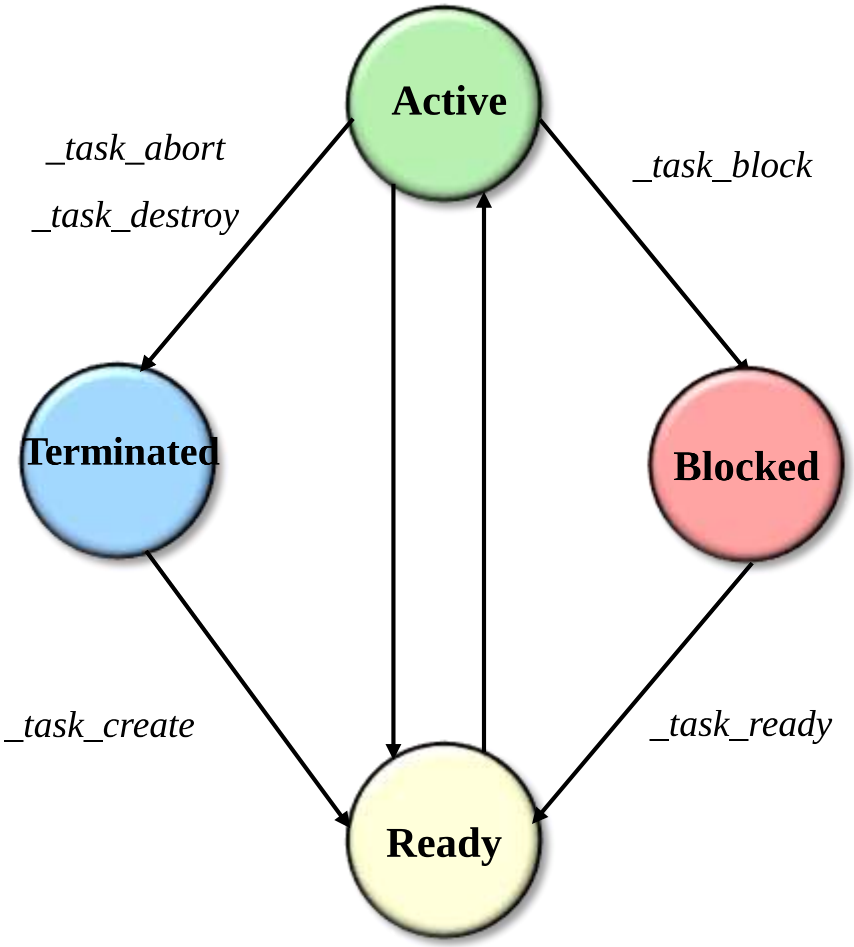


MQX Tasks

- Tasks can be automatically created when MQX Starts; also, any task can create another task by calling `_task_create()` or `_task_create_blocked()`

- The function `_task_create()` puts the child task in the ready state and the scheduler puts the higher priority task to run

- If `_task_create_blocked` is used the task is not ready until `_task_ready()` is called



Creating a Task

- **When creating a task you have to:**
  - ❑ Make the task prototype and index definition

```
#define INIT_TASK 5
extern void init_task(uint_32);
```

  - ❑ Add the task in the Task Template List

```
TASK_TEMPLATE_STRUCT MQX_template_list[] =
{
     { TASK_INDEX, TASK, STACK, TASK_PRIORITY,
TASK_NAME, TASK_ATTRIBUTES, CREATION_PARAMETER,
TIME_SLICE}
}
```

Using the init_task example:

```
TASK_TEMPLATE_STRUCT MQX_template_list[] =
{
     {INIT_TASK, init_task, 100, 9, "init",
       MQX_AUTO_START_TASK, 0, 0},
}
```

Creating a Task (Continue)

```
TASK_TEMPLATE_STRUCT MQX_template_list[] =
{
   {TASK_INDEX, TASK, STACK, TASK_PRIORITY, TASK_NAME,
      TASK_ATTRIBUTES, CREATION_PARAMETER, TIME_SLICE}
}
```

- TASK_INDEX: is usually a Define with an index number.
- TASK: Refers to the function name; C compiler takes the address pointer of the function name.
- STACK is the defines stack size.
- TASK_PRIORITY; the lower number, the higher priority. Task with priority 0 disables all the interrupts ,Priorities 0 to 7 are used by the OS Kernel.

Creating a Task (Continue)

```
TASK_TEMPLATE_STRUCT MQX_template_list[] =
{
   {TASK_INDEX, TASK, STACK, TASK_PRIORITY, TASK_NAME,
      TASK_ATTRIBUTES, CREATION_PARAMETER, TIME_SLICE}
}
```

- TASK_NAME is a string that helps to identify the task. It is also used to get the task ID.
- TASK_ATTRIBUTES.
  - Auto start — when MQX starts, it creates one instance of the task.
  - DSP — MQX saves the DSP co-processor registers as part of the task’s context.
  - Floating point — MQX saves floating-point registers as part of the task’s context.
  - Time slice — MQX uses round robin scheduling for the task. Default is FIFO.

- CREATION_PARAMETER: is the parameter to be passed to this task, when created.
- TIME_SLICE: Time slice (in milliseconds) used for the task when using round-robin scheduling. Ex:150 ms.

Creating a Task (Continue)

- **When creating a task you have to:**

  □ Make the task definition

```
void init_task(void)
{
    /* Put the Task Code here */
}
```

  □ During execution time, create the task using

```
task_create()
```

(if it is not an auto start task)

MQX_Template_List

```
{ WORLD_ID, world_task, 150, 9,
  "world_task",
   MQX_AUTO_START_TASK, 0, 0},
```

```
{ HELLO_ID, hello_task, 100, 8,
  “hello_task",
   MQX_TIME_SLICE_TASK, 0, 100},
```

```
{ LED_ID, led_task, 125, 10,
   “LED Task",
   MQX_AUTO_START_TASK |
     MQX_TIME_SLICE_TASK, 0, 50},
```

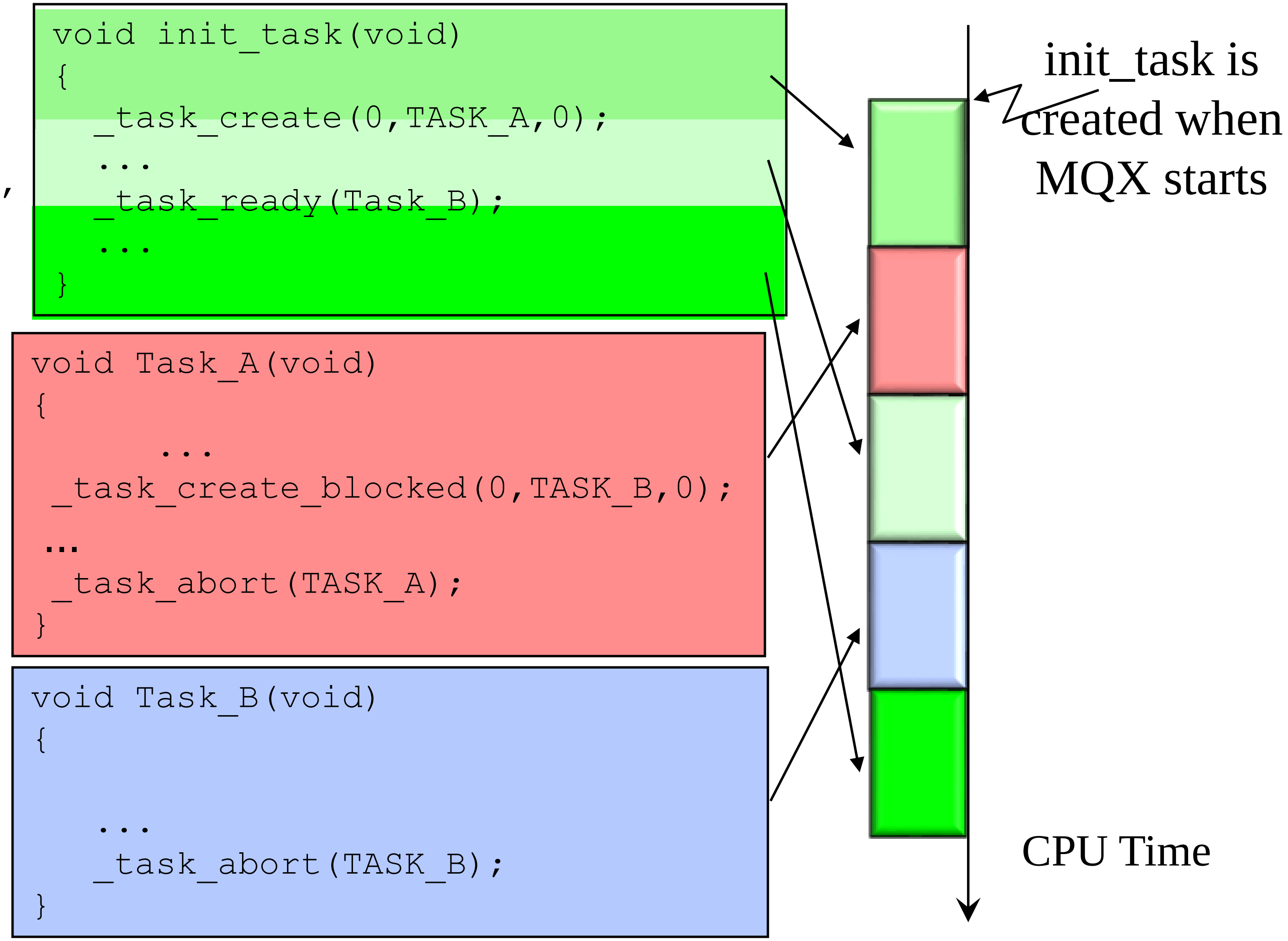
At least one task must be set to MQX_AUTO_START_TASK.

MQX - Task Management Example

```
{INIT_TASK,
init_task, 100, 11,
"init",
MQX_AUTO_START_TASK,
0, 0},
```

```
{TASK_A,
 Task_A, 100, 10,
“Task A",
 0,
 0, 0},
```

```
{TASK_B,
 Task_B, 100, 9,
“Task B",
 0,
 0, 0},
```



```
void init_task(void)
{
   _task_create(0,TASK_A,0);
   ...
   _task_ready(Task_B);
   ...
}
```

```
void Task_A(void)
{
       ...
 _task_create_blocked(0,TASK_B,0);
…
 _task_abort(TASK_A);
}
```

```
void Task_B(void)
{


   ...
   _task_abort(TASK_B);
}
```

init_task is created when MQX starts

Agenda

- **What is an RTOS?**
- **MQX Basics: Tasks**
- **MQX Basics: Scheduling**
- → **MQX Basics: Task Synchronization**
  - Semaphores
  - Events and Messages
- **MQX Intermediate**
  - Libraries
  - Interrupts
  - BSP
- **Additional Resources**
- **Review**

Competence Condition

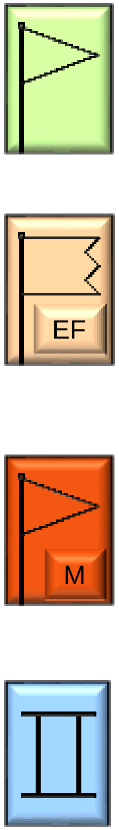
- What happens if two tasks access the same resource at the same time?
  - ❑ We call this “competence condition”. When two or more tasks read or write on share a resource at a certain moment

- Why the “competence condition” can be a problem?
  - ❑ Memory corruption
  - ❑ Wrong results
  - ❑ Unstable application
  - ❑ Device conflicts

Why Synchronization?

- **Synchronization may be used to solve:**
  - ❑ Mutual Exclusion
  - ❑ Control Flow
  - ❑ Data Flow

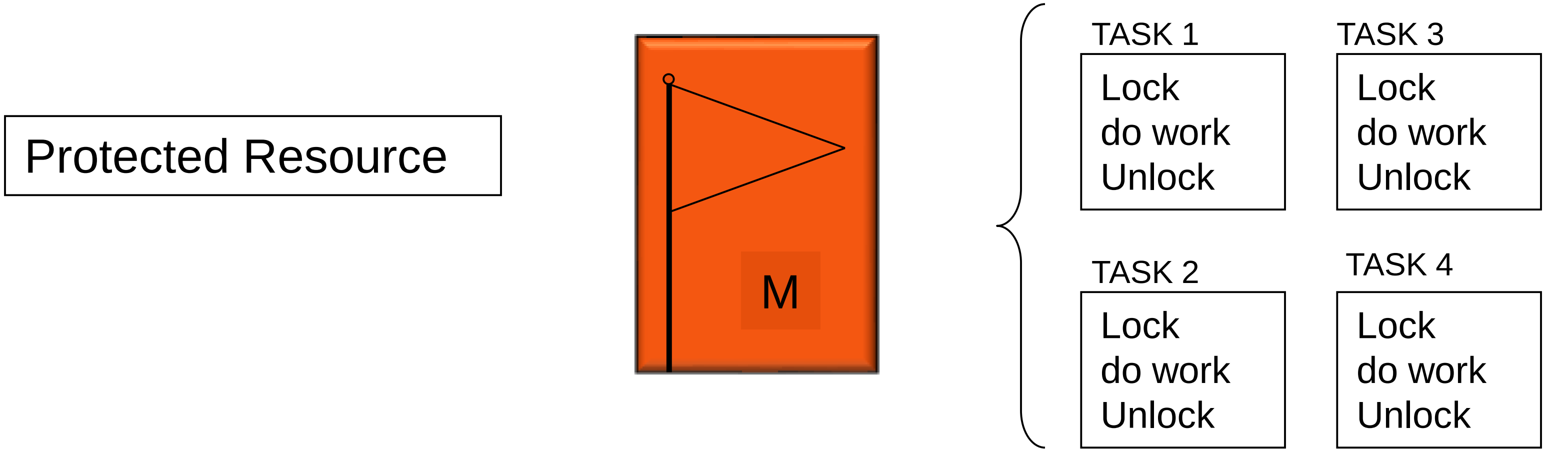
❑ **Synchronization Mechanisms include:**
  - ❑ Semaphores
  - ❑ Events
  - ❑ Mutexs
  - ❑ Message Queues



- **The correct synchronization mechanism depends on the synchronization issue being addressed**

Mutual Exclusion

- Allowing only one task at a time to access a shared resource
- Resource may be devices, files, memory, drivers, code…
- Mutual exclusion locks the resource



Protected Resource

M

TASK 1
Lock
do work
Unlock

TASK 3
Lock
do work
Unlock

TASK 2
Lock
do work
Unlock

TASK 4
Lock
do work
Unlock

Semaphores

- A semaphore is a protocol mechanism offered by most multitasking kernels. Semaphores are used to:
  - ❑ Control access to a shared resource (mutual exclusion)
  - ❑ Signal the occurrence of an event
  - ❑ Allow two tasks to synchronize their activities
- Semaphore has two types

  (a) Binary semaphore, (resource Only one).

  (b) Counting semaphore
- If the semaphore is already in use, the requesting task is suspended until the semaphore is released by its current owner

How Semaphores Work

- **A semaphore has:**
  - counter — maximum number of concurrent accesses
  - queue — for tasks that wait for access
- **If a task waits for a semaphore**

```
if counter > 0
          counter is decremented by 1
          task gets the semaphore and can do work
  else
          task is put in the queue
```

- **If a task releases (post) a semaphore**

```
if at least one task is in the semaphore queue
          appropriate task is readied, according to the queuing policy
  else
          counter is incremented by 1
```

Synchronization Mechanisms

- **Synchronization may be used to solve:**
  - ❑ Mutual Exclusion
  - ❑ Control Flow
  - ❑ Data Flow
- ❑ **Synchronization Mechanisms include:**
  - ❑ Semaphores
  - ❑ Events
  - ❑ Mutexs
  - ❑ Message Queues
- **The correct synchronization mechanism depends on the synchronization issue being addressed**

Events

```
If (EventBit == 0x03)
:
Else
:
```

- Tasks can wait for a combination of event bits to become set. A task can set or clear a combination of event bits.

- Events can be used to synchronize a task with another task or with an ISR.

- The event component consists of event groups, which are groupings of event bits.
  - 32 event bits per group (mqx_unit)

Ex: MotorStarEvent = user_pressed+ Mcurrent zero + Speed 0

- Tasks can wait for all or any set of event bits in an event group (with an *optional timeout*)

- Event groups can be identified by name or by index (fast event groups)

Messages Passing by Message Queue

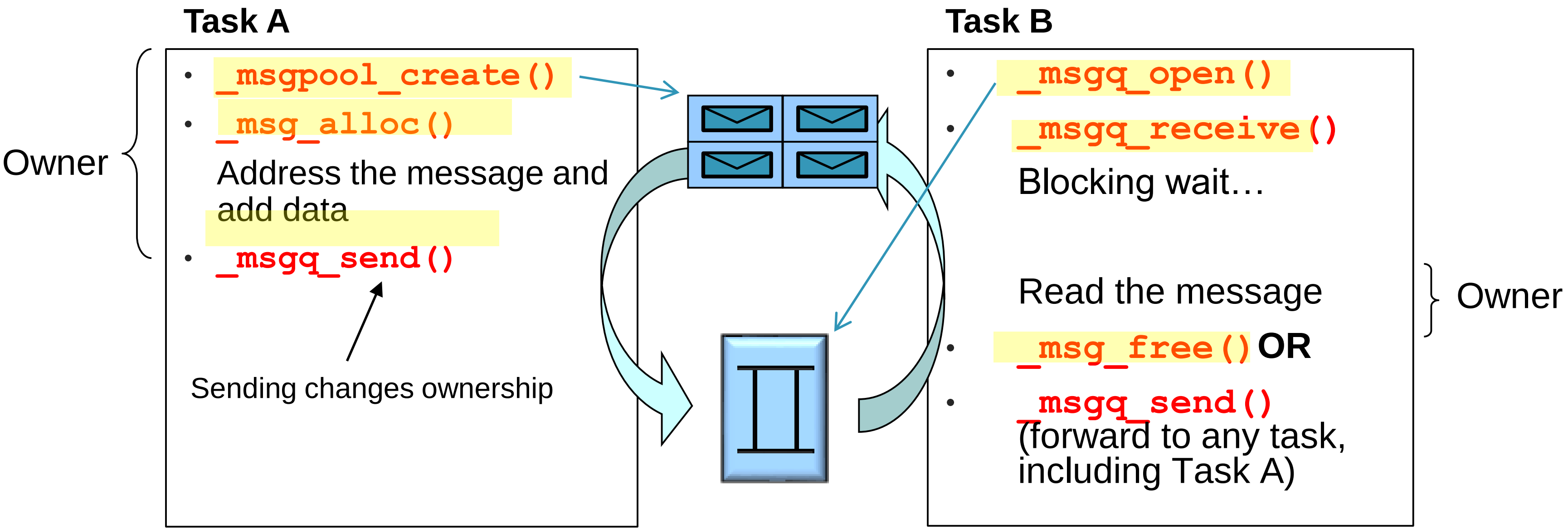
- Tasks can communicate with each other by
  exchanging messages
  (e.g Clipboard in windows, or copy in mobile phone)

- Tasks send messages to queues opened by system (system message pool, broadcast)
  or other tasks,
  Receive messages from owned message queues.

- Messages can be assigned a priority or marked urgent

- Messages are an optional component in MQX.

Message passing example



Task A

Owner

- _msgpool_create()
- _msg_alloc()

  Address the message and add data
- _msgq_send()

Sending changes ownership

Task B

- _msgq_open()
- _msgq_receive()

  Blocking wait…

  Read the message
- _msg_free() **OR**
- _msgq_send()

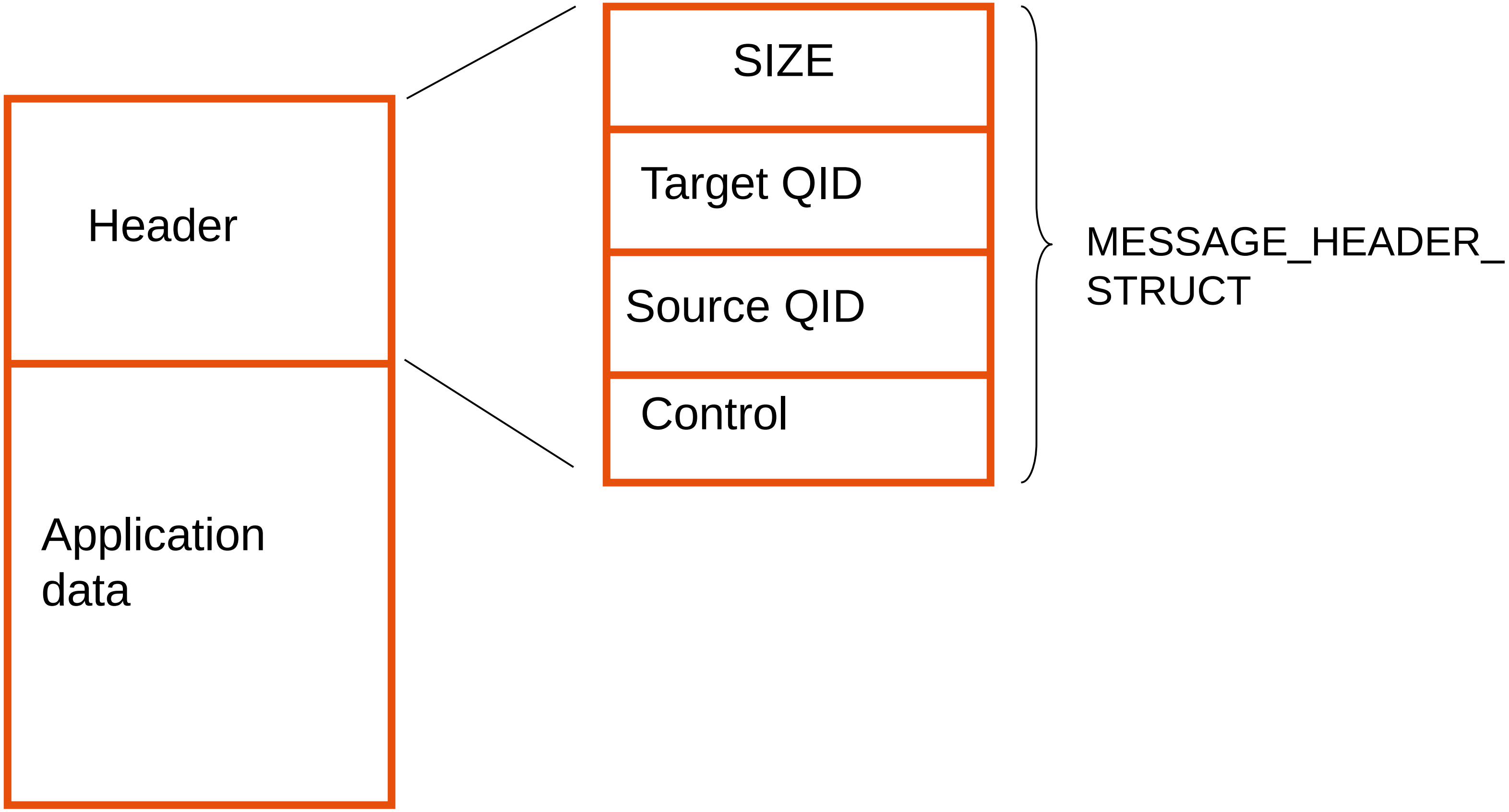
  (forward to any task, including Task A)

Owner

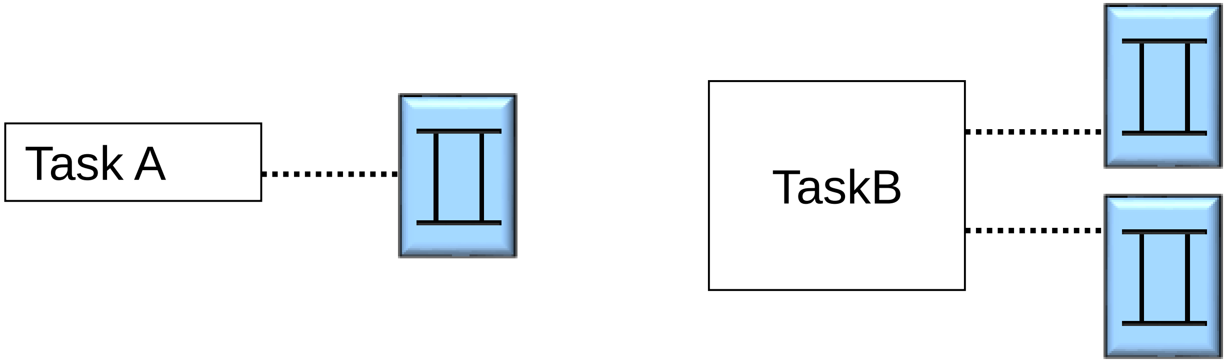
- Message must “travel” in a loop:
  - Allocate it from a pool
  - Use it
  - Return it to pool (i.e. free it)

Message Format

- Messages are areas of memory divided into a header and a data area
- Application data is user-defined



Message Queues



- Each task can have one (or more) messages queues associated with it
- Messages are always addressed to queues, not tasks
- Queues are identified by _queue_id
  - This is a combination of queue number and CPU number
- Create a queue using `_msgq_open()`

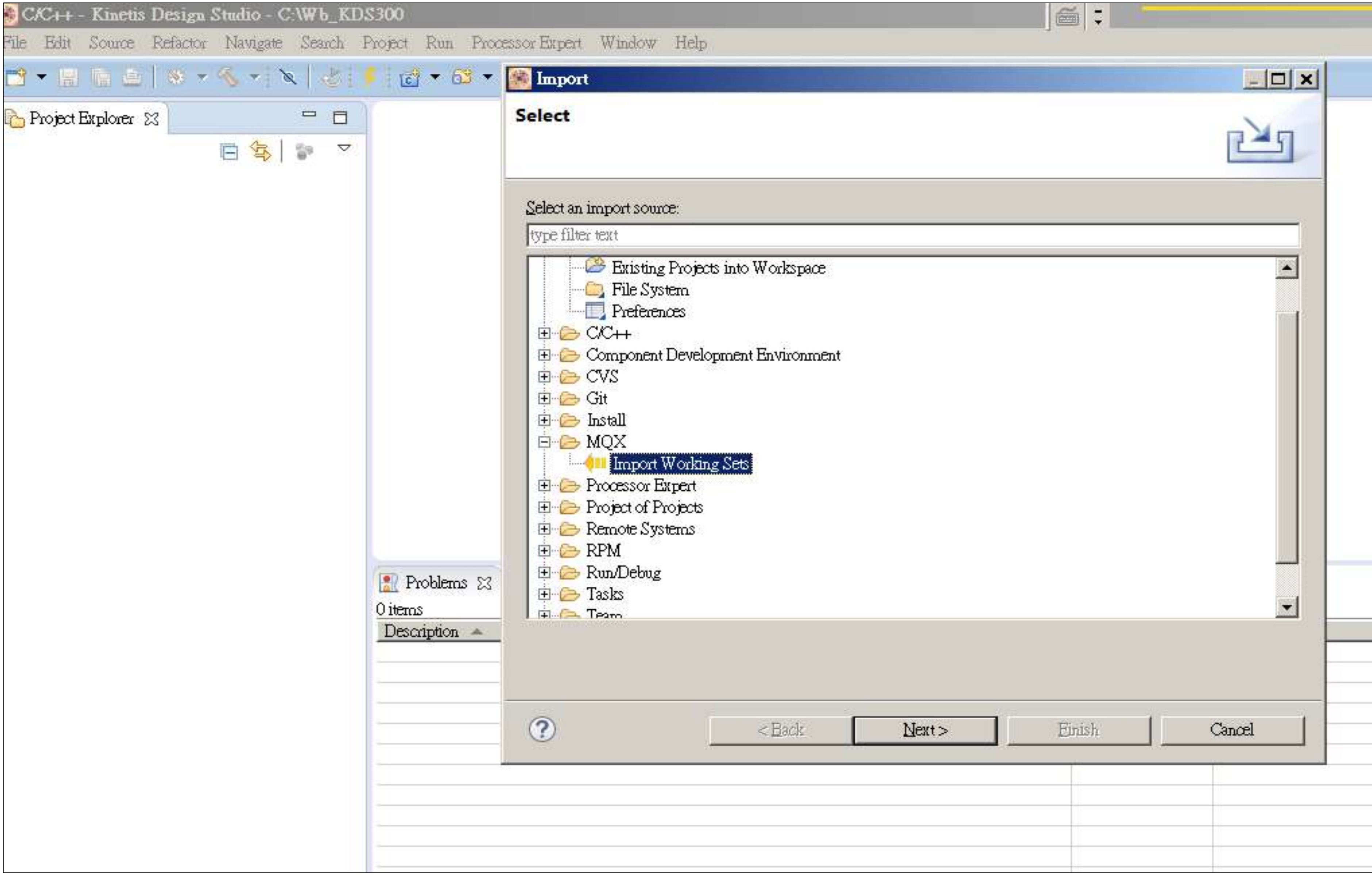
MQX Interrupts

- Embedded systems are based on ISR
- Usually an ISR is used for signal an event
- The most common actions on an ISR are:
  - Post a semaphore
  - Send a message
  - Set an event
  - Clear an error condition
- **Important**: ISRs are not tasks
- **Remember:** ISR should be short and should not use blocking functions.

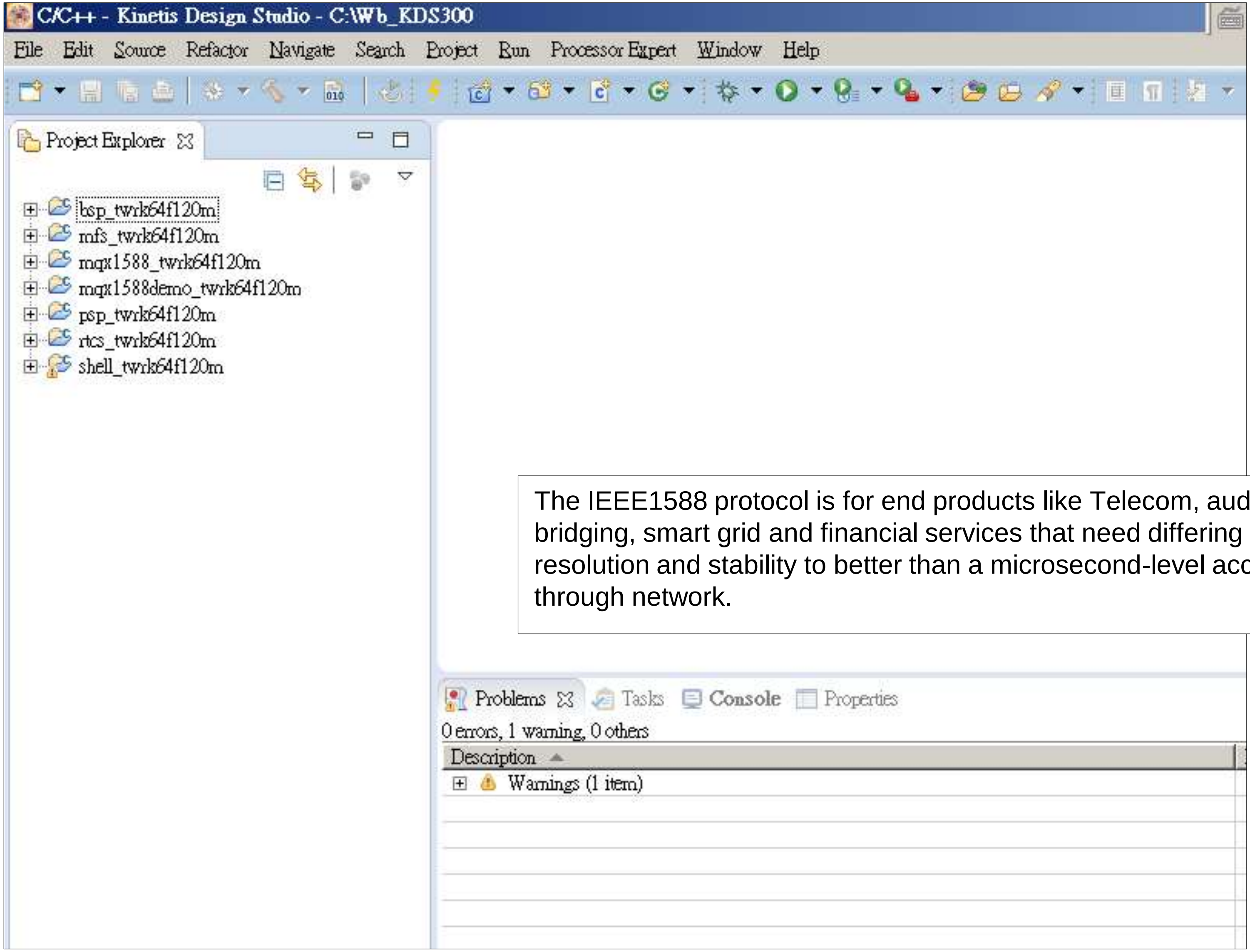
Agenda

- **What is an RTOS?**
- **MQX Basics: Tasks**
- **MQX Basics: Scheduling**
- **MQX Basics: Task Synchronization**
  - Semaphores
  - Events and Messages
- **MQX Intermediate**
  - Working Sets
  - Recompile
  - Clone Wizard
- **Additional Resources**
- **Review**

Import Whole Working Sets (*.wsd)

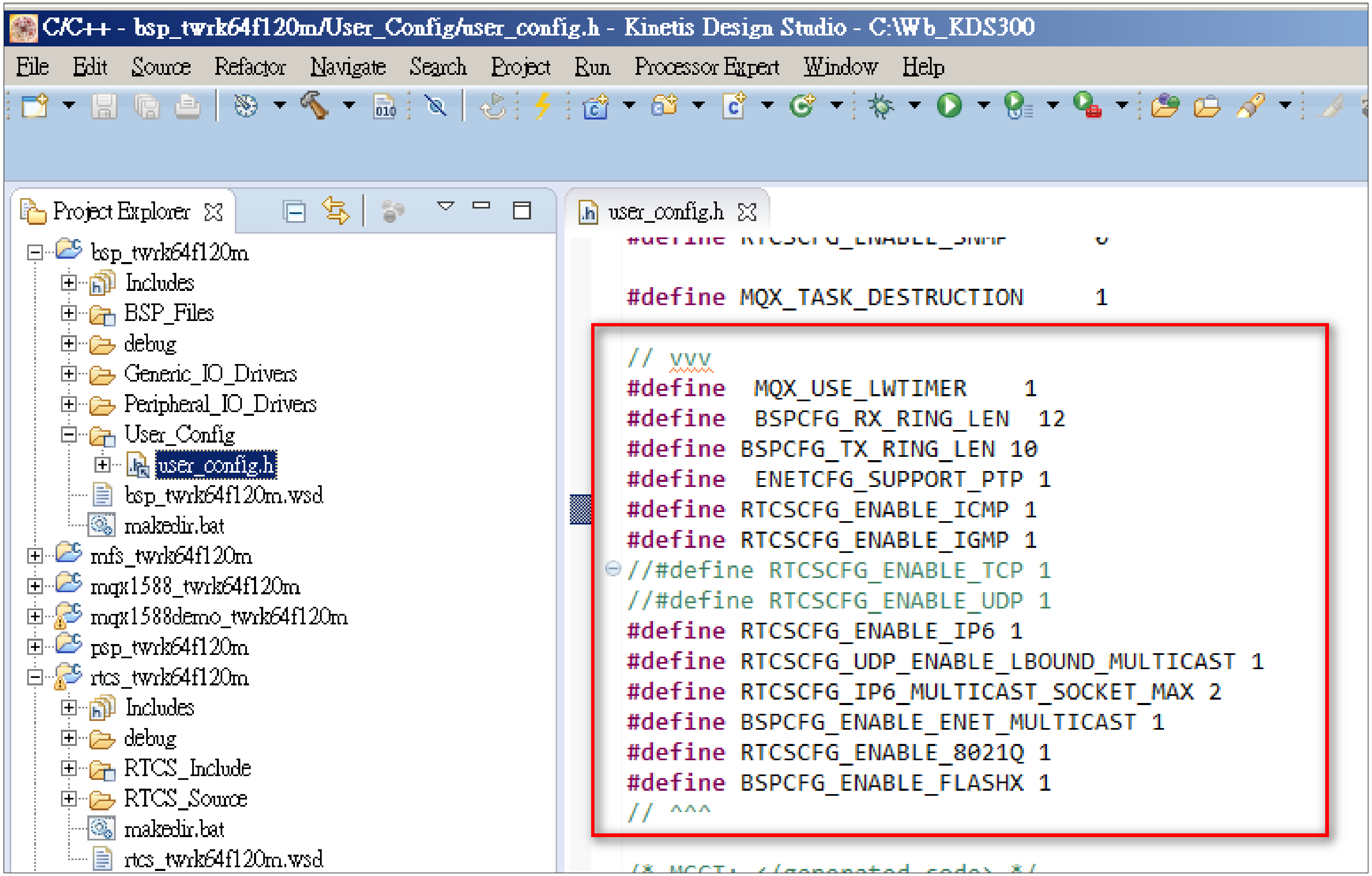


Import Whole Working Sets (*.wsd)



The IEEE1588 protocol is for end products like Telecom, audio video bridging, smart grid and financial services that need differing precision, resolution and stability to better than a microsecond-level accuracy through network.

Recompile

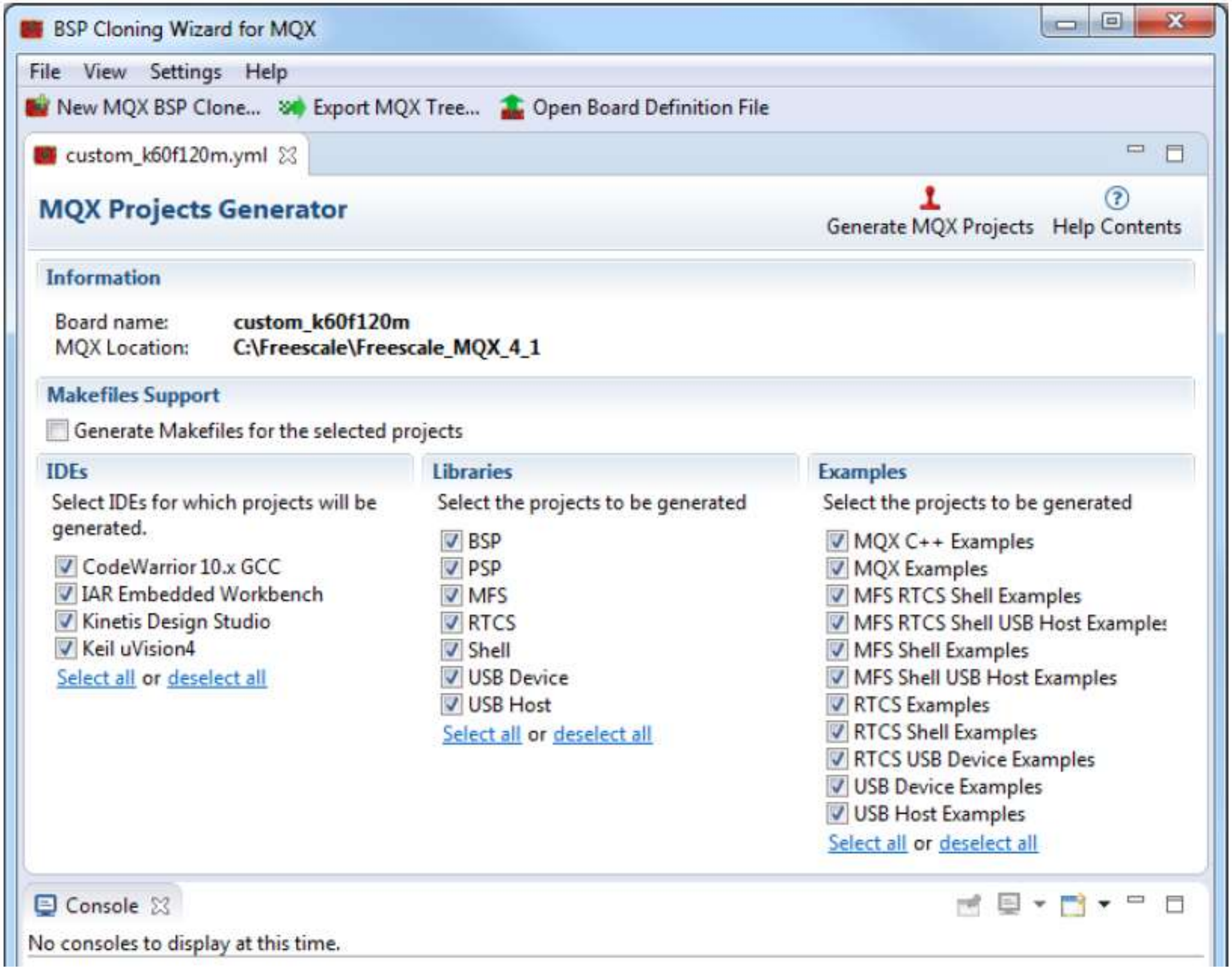


```
#define RTCSCFG_ENABLE_SNMP          0

#define MQX_TASK_DESTRUCTION        1

// vvv
#define  MQX_USE_LWTIMER     1
#define  BSPCFG_RX_RING_LEN   12
#define BSPCFG_TX_RING_LEN 10
#define  ENETCFG_SUPPORT_PTP 1
#define RTCSCFG_ENABLE_ICMP 1
#define RTCSCFG_ENABLE_IGMP 1
//#define RTCSCFG_ENABLE_TCP 1
//#define RTCSCFG_ENABLE_UDP 1
#define RTCSCFG_ENABLE_IP6 1
#define RTCSCFG_UDP_ENABLE_LBOUND_MULTICAST 1
#define RTCSCFG_IP6_MULTICAST_SOCKET_MAX 2
#define BSPCFG_ENABLE_ENET_MULTICAST 1
#define RTCSCFG_ENABLE_8021Q 1
#define BSPCFG_ENABLE_FLASHX 1
// ^^^

/* MGCT: </generated_code> */
```

Re-Compiling

- Anytime a change is made to **user_config.h** the libraries should be re-compiled. This over-writes all the files in lib for that board.
- Anytime a change is made in the library source code, the library should be re-compiled.
- To re-compile the libraries, open up the library projects for the board.

Clone wizard



BSP Cloning Wizard for MQX
File View Settings Help
New MQX BSP Clone... Export MQX Tree... Open Board Definition File
custom_k60f120m.yml
Generate MQX Projects
Help Contents
Console
No consoles to display at this time.

MQX Projects Generator

Information

Board name: **custom_k60f120m**
MQX Location: **C:\Freescale\Freescale_MQX_4_1**

Makefiles Support

☐ Generate Makefiles for the selected projects

IDEs

Select IDEs for which projects will be generated.

- ☑ CodeWarrior 10.x GCC
- ☑ IAR Embedded Workbench
- ☑ Kinetis Design Studio
- ☑ Keil uVision4

Select all or deselect all

Libraries

Select the projects to be generated

- ☑ BSP
- ☑ PSP
- ☑ MFS
- ☑ RTCS
- ☑ Shell
- ☑ USB Device
- ☑ USB Host

Select all or deselect all

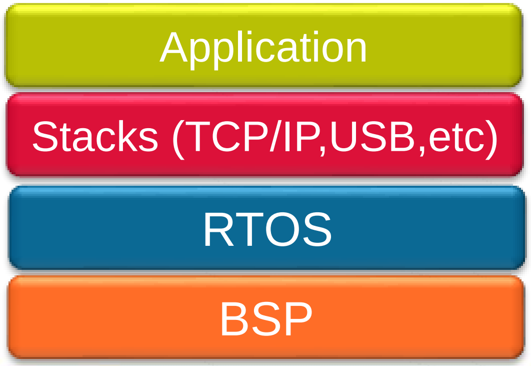
Examples

Select the projects to be generated

- ☑ MQX C++ Examples
- ☑ MQX Examples
- ☑ MFS RTCS Shell Examples
- ☑ MFS RTCS Shell USB Host Example:
- ☑ MFS Shell Examples
- ☑ MFS Shell USB Host Examples
- ☑ RTCS Examples
- ☑ RTCS Shell Examples
- ☑ RTCS USB Device Examples
- ☑ USB Device Examples
- ☑ USB Host Examples

Select all or deselect all

MQX Board Support Package



Application

Stacks (TCP/IP,USB,etc)

RTOS

BSP

- Initializes microprocessor and board
  - PLL and clocks, memory interface, core registers
- Defines board specific parameters
  - Clocks, memory parameters, interrupt usage, driver parameters/enabling, MQX limits, IO pin definitions, ENET interfaces, etc.
- Presents board-specific API to I/O drivers and appli
  - Timer ISR functions used by MQX scheduler, I/O pin initializations
- Installs and initializes device drivers (selected by *user_config.h)*

Agenda

- **What is an RTOS?**
- **MQX Basics: Tasks**
- **MQX Basics: Scheduling**
- **MQX Basics: Task Synchronization**
  - Semaphores
  - Events and Messages
- **MQX Intermediate**
  - Libraries
  - Interrupts
  - BSP
- **Additional Resources**
- **Review**

Additional Resources

Kinetis MQX Quick Start Demos

- Source code and lab guide available online for both K40 SLCD and K60 Web server demos (IAR and CW10.1)
  - http://freescale.com/twr-k40x256
  - http://freescale.com/twr-k60n512

- Showcases Ethernet, SLCD, SD Card, USB, I2C, ADC, TSI, RNG, UART, RTC, Flash, and GPIO features on Kinetis.

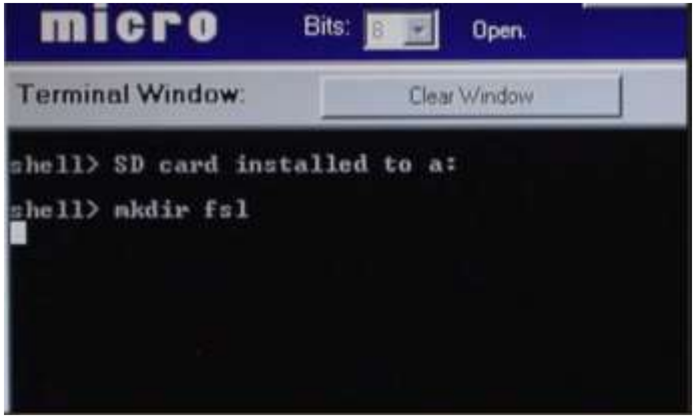
- TWR-K40X256
  - Display seconds, hours and minutes, potentiometer, and temperature
  - http://youtu.be/4sSRHyYyilA

- TWR-K60N512
  - Interactive web server and touch memory game
  - http://youtu.be/gkL4n2b5RU4

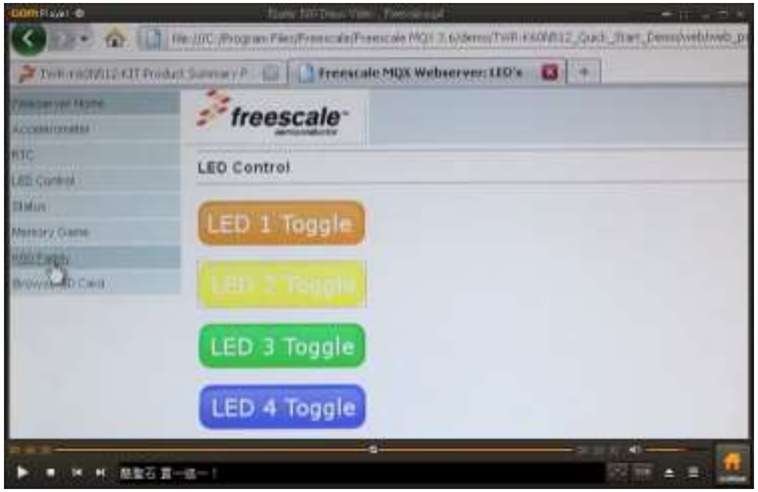
Learn more at: www.freescale.com/MQX

Watch the K60 quick start video

- (a) Assembly
- (b) OS console
- (c) SD card access & File system.
- (d) USB mouse
- (e) Ethernet Web server



```
shell> SD card installed to a:
shell> mkdir fsl
```



Further Reading and Training

- **Webinnar at www.freescale.com/tower**
  - Introduction to Tower, CodeWarrior 10, and MQX
  - TWR-K60N512 and TWR-K40X256 Quick Start Demos

- **Videos: www.freescale.com/mqx**
  - Getting started with MQX
  - And more

- **vFTF technical session videos www.freescale.com/vftf**
  - Introducing a modular system, Serial-to-Ethernet V1 ColdFire® MCU and Complimentary MQX™ RTOS
  - Writing First MQX Application
  - Implementing Ethernet Connectivity with the complimentary Freescale MQX™ RTOS

Further Reading and Training (Continue)

- **MQX Release Notes**
- **MQX User’s Guide**
- **Writing First MQX Application (AN3905)**
- **Using MQX: RTCS, USB, and MFS (AN3907)**
- **How to Develop I/O Drivers for MQX (AN3902)**
- **IP Camera and USB Snapshot with MQX (AN4022)**
- **Supporting New Toolchains with Freescale MQX RTOS (4190)**
- **Motor Control Under the Freescale MQX Operating System (AN4254)**
- **MQX Board Support Package Porting Guide (AN4287)**

www.Freescale.com