

# Using IAR Embedded Workbench for Freescale Kinetis MCU

*IAR Systems, Shanghai*  
*ryan.sheng@iar.com*



- Established in 1983
- Headquarter: Uppsala, Sweden
- 160+ employees
- Support for 8000+ devices
  - 2000+ ARM devices
- A world-leading embedded development tools vendor
- Main products
  - IAR Embedded Workbench: C/C++ Compiler & Debugger Tools
  - IAR visualSTATE: State-Machine Modeling & Software Design Tools
  - IAR I-jet / I-scope / JTAGjet: Debugging & Trace Probes
- China office
  - Shanghai, 021-63758658

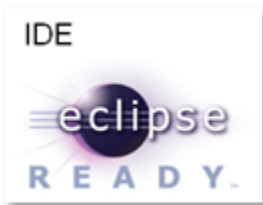
# Strategic collaboration with Freescale

- Long partnership with Freescale
- Initiated close cooperation around HC12 & S12
- EWCF: released on 2007
- EWS08: released on 2008
- EWARM is the most widely used commercial tool chain for ARM-based MCU/MPU
- Expand the Freescale ecosystem
- IAR Embedded Workbench

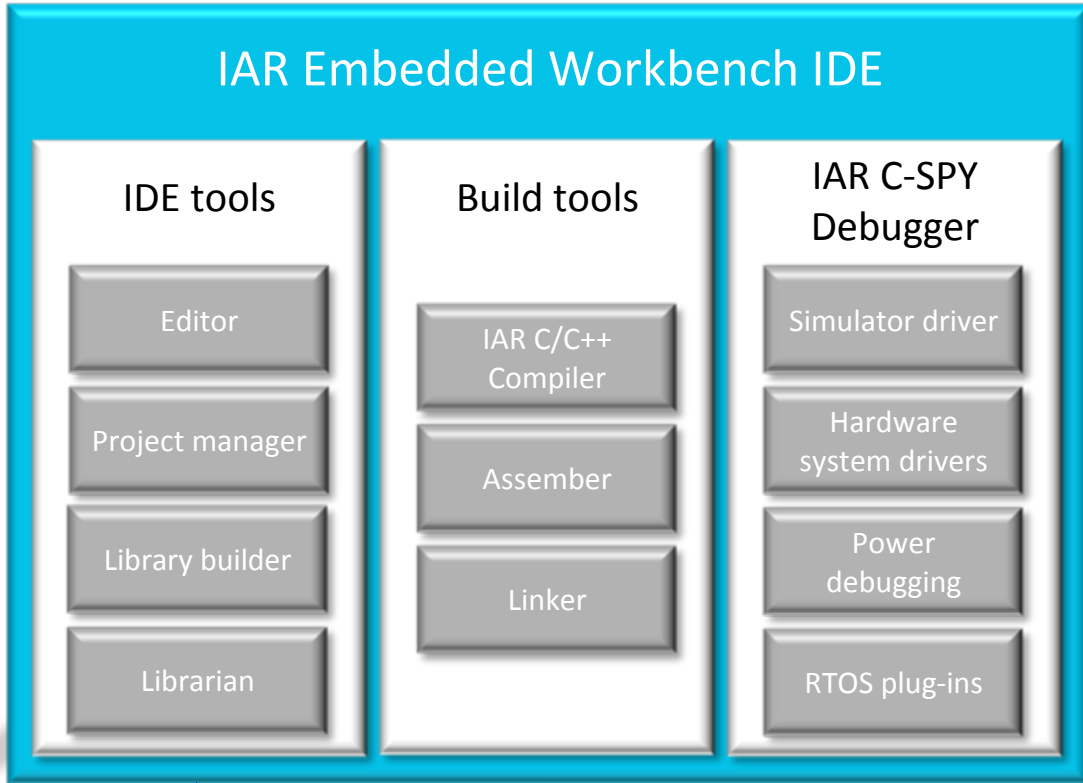


- EWHCS12: HC12 & S12 MCU
- EWCF: ColdFire & ColdFire+ MCU/MPU
- EWS08: S08 MCU
- EWARM: Kinetis, i.MX, Vybrid, MC1322x, ...

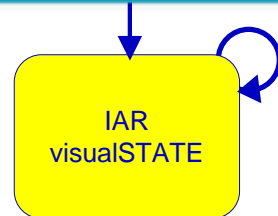
# IAR Embedded Workbench for ARM



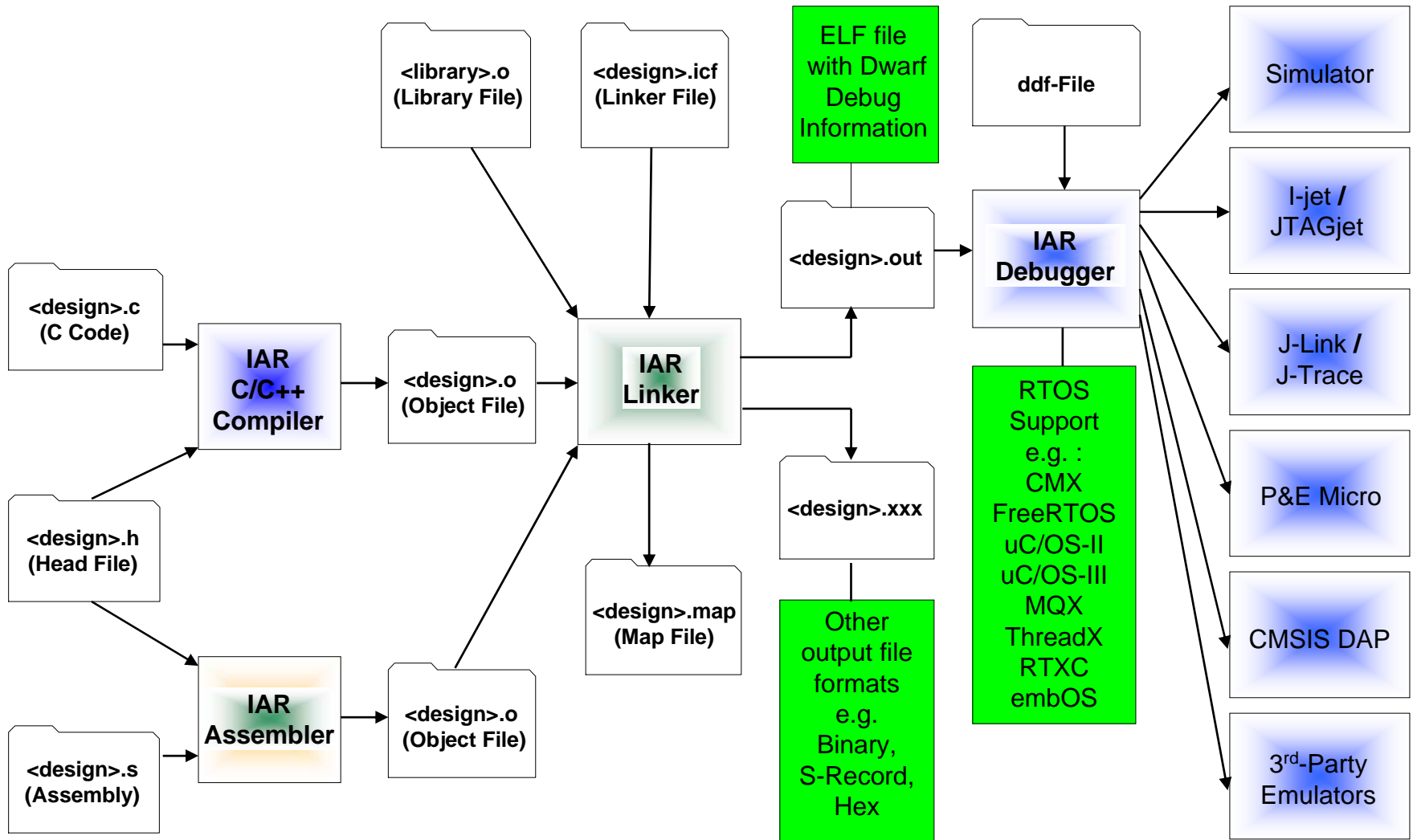
P L U G I N S



Configuration tools



# IAR Embedded Workbench for ARM

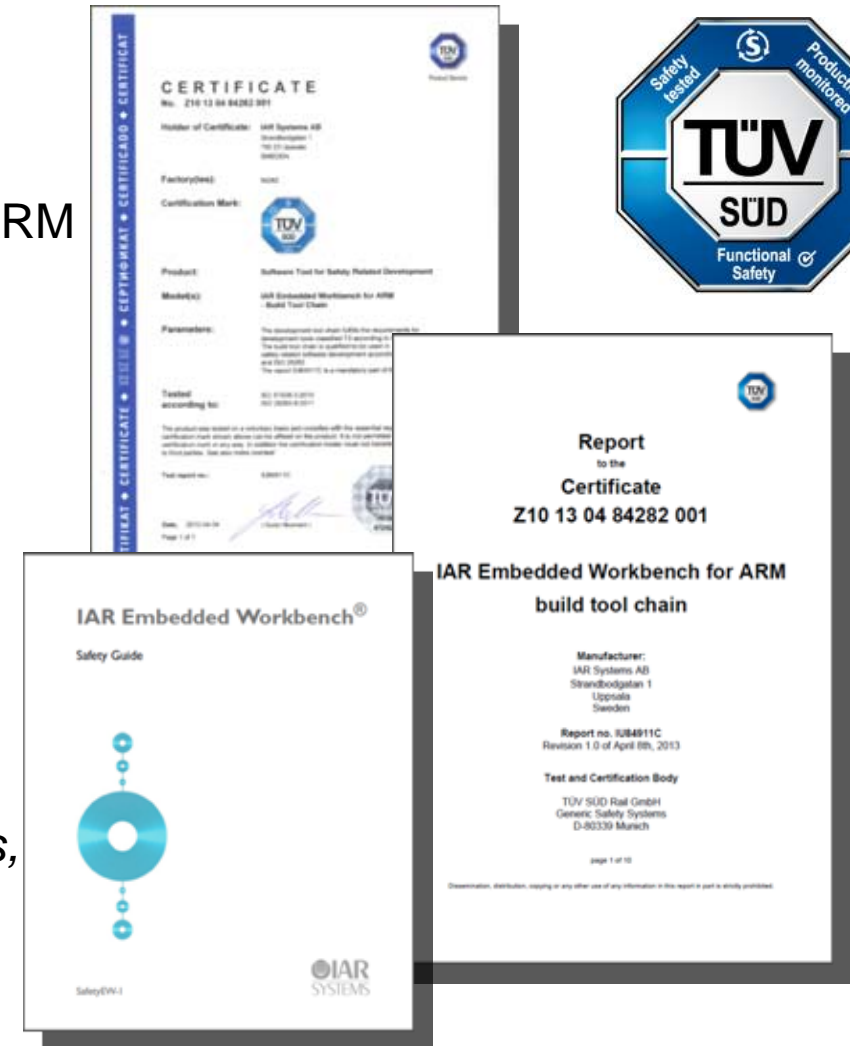


# EWARM: Product variants

- EWARM -- *Standard edition*
- EWARM-BL -- *Baseline edition*
  - 256KB code size limitation
- EWARM-CM -- *Cortex-M edition*
  - Only for Cortex-M0/M0+/M1/M3/M4
- EWARM-CM0 -- *CM0/CM1 only*
  - Only for Cortex-M0/M0+/M1
- EWARMFS -- *Functional Safety*
  - IEC-61508 and ISO-26262 Certificated

# EWARM: Functional Safety certificate

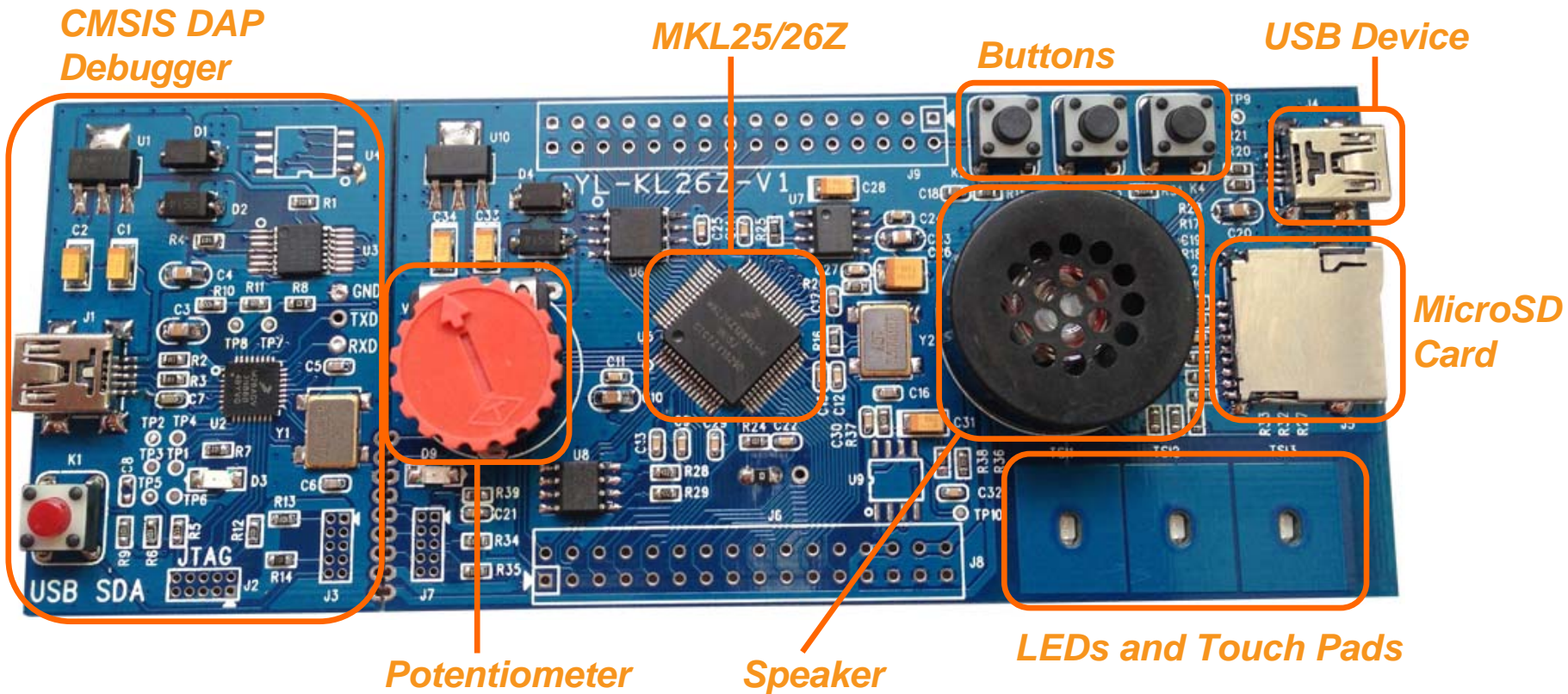
- EWARMFS
  - The Functional Safety edition of IAR Embedded Workbench for ARM
  - Current edition: 6.50.4
- Certified by TÜV SÜD
- Functional Safety Standards
  - IEC 61508-3:2010 (SIL 3)  
*For electrical, electronic and programmable systems in all kinds of industry.*
  - ISO 26262-8:2011 (ASIL D)  
*Safety standard for road vehicles, derived from IEC 61508.*
- [www.iar.com/safety](http://www.iar.com/safety)



# Project Connection with Freescale Processor Expert



# YL-KL25Z development board



# Processor Expert Software

PEx



## Target System

Kinetis L MCU

On-Chip Peripherals

On-Chip Flash

On-Chip RAM

Cortex-M0+

JTAG/SWD

Code

User-Written Code

Task1

Task2

Task3

.....

Auto-generated device drivers

MQX-Lite Kernel

## Emulator

JTAG/SWD

On-Board CMSIS DAP

IAR

Compiler / Assembler

C/C++ Library

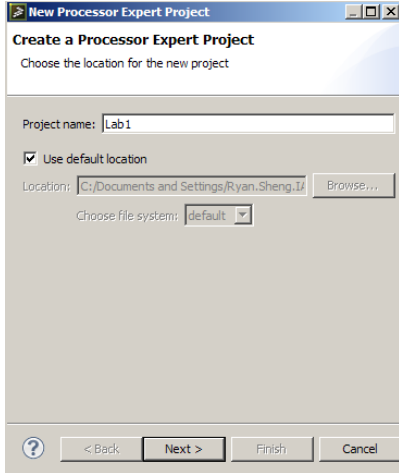
Linker

Project Manager

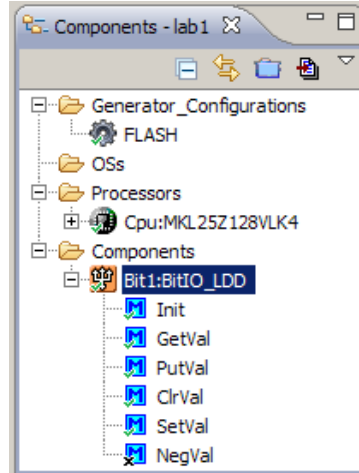
EWARM

C-SPY Debugger

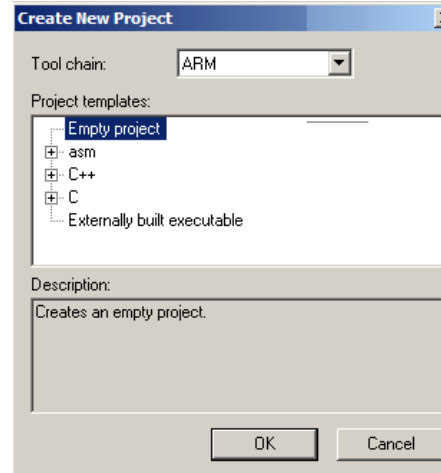
## PEX Create project



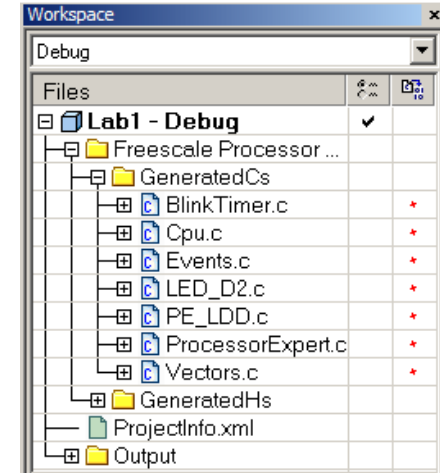
## PEX Configure components



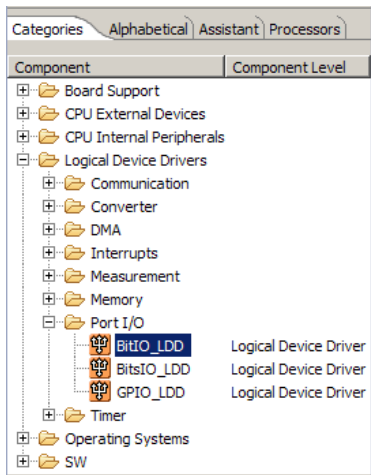
## IAR EWARM Create project



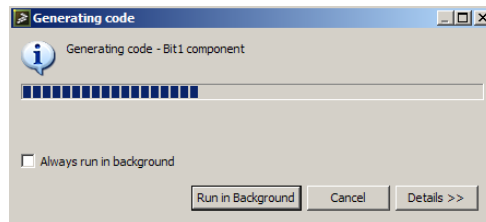
## IAR EWARM Configure project options



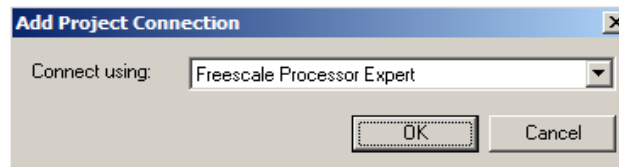
## PEX Add components



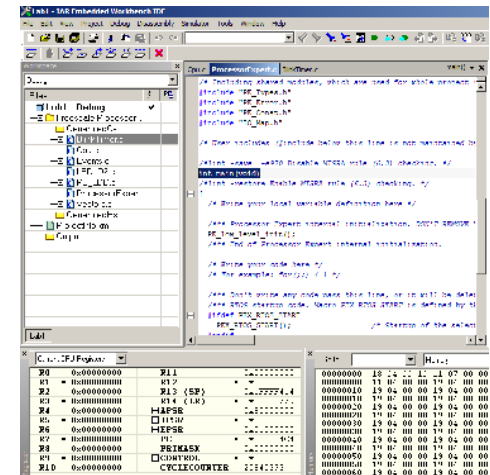
## PEX Generate code



## IAR EWARM Connect to PEX project



## IAR EWARM Build and Debug



# Processor Expert overview

**C/C++ Perspective**

**File Explorer**

**Component Inspector - Cpu**

Name	Value	Details
CPU type	MKL26Z128VMC4	
<b>Clock settings</b>		
Initialization priority	interrupts enabled	1
Watchdog disable	yes	
<b>CPU interrupts/resets</b>		
<b>Clock configurations</b>	1	
<b>Clock configuration 0</b>		
<b>Clock source setting</b>	configuration 0	
MCG mode	FEI	
<b>System clocks</b>		
Core clock	20.97152	20.97152 MHz
Bus clock	20.97152	20.97152 MHz
<b>TPM clock selection</b>	Auto select	PLL/FLL clock
Clock frequency [MH	20.97152	20.97152 MHz

**Component Inspector**

**Components Library**

- CPU External Devices
- CPU Internal Peripherals
- Logical Device Drivers
  - Communication
  - Converter
  - DMA
  - Interrupts
  - Measurement
  - Memory
  - Port I/O
    - BitIO\_LDD Logical Device ...
    - BitsIO\_LDD Logical Device ...
    - GPIO\_LDD Logical Device ...
  - Timer
- Operating Systems
- SW

**Source Editor**

```
#include "Cpu.h"
#include "Events.h"

#ifdef __cplusplus
extern "C" {
#endif

/* User includes (#include below this line is not maintained by Processor Expert) */
```

**Components**

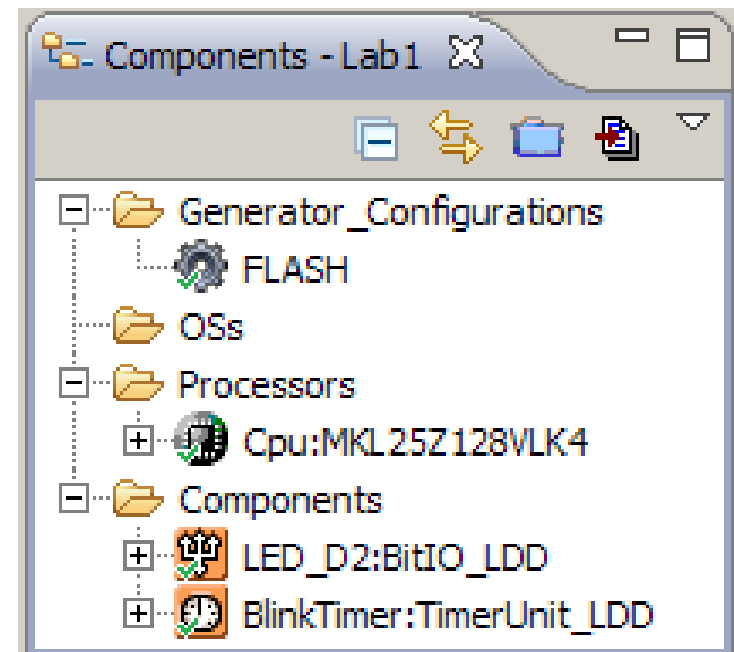
- Generator\_Configurations
  - FLASH
  - OSs
- Processors
  - Cpu:MKL26Z128VMC4
- Components

**Problems View**

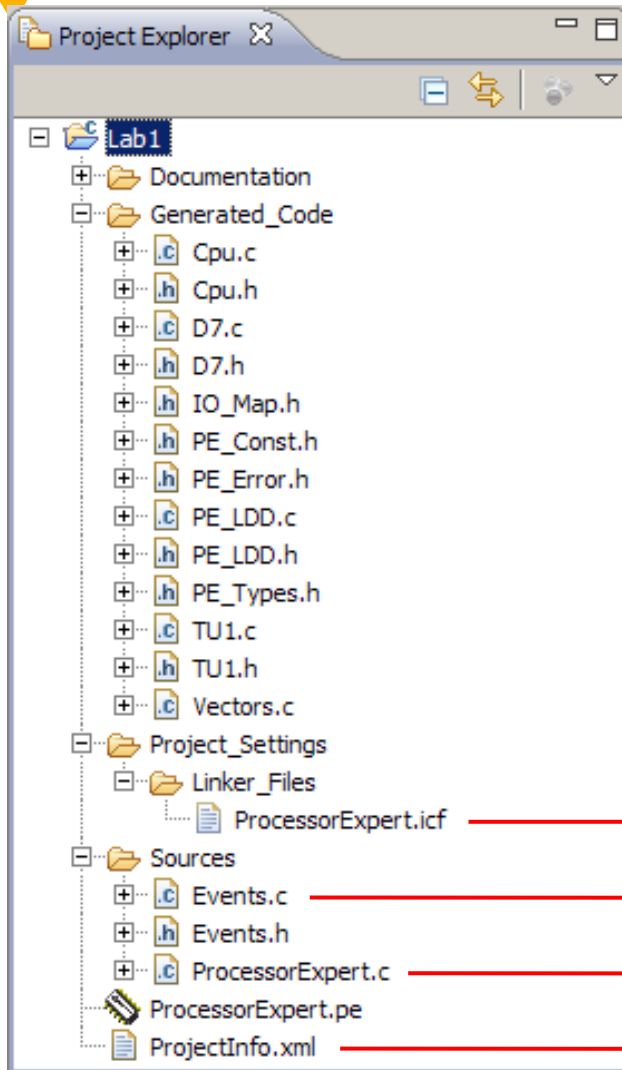
Description	Resource	Path	Location	Type
0 items				

# Example 1

- Blink a LED in the timer interrupt
- PEx Components
  - CPU: MKL25Z
  - GPIO: BitIO\_LDD
  - Timer: TimerUnit\_LDD



# Processor Expert generated code



*Auto-generated device drivers, initialization code and the interrupt vector table. Do not modify these files because they might be overwritten by the next code generation.*

*Linker configuration file for IAR EWARM*

*User's event handler module*

*User's application code, e.g. main()*

*Project information (XML format)*

# ProcessorExpert.c

```
int main (void)
{
    /* Write your local variable definition here */

    /* Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! */
    PE_low_level_init();
    /* End of Processor Expert internal initialization. */

    /* Write your code here */
    /* For example: for(;;) { } */

    /* Don't write any code pass this line, or it will be deleted during code generation. */
    /* RTOS startup code. Macro PEX_RTOS_START is defined by the RTOS component. DON'T MODIFY THIS CODE! */
    #ifdef PEX_RTOS_START
        PEX_RTOS_START(); /* Startup of the selected RTOS. Macro is defined by the RTOS component. */
    #endif
    /* End of RTOS startup code. */
    /* Processor Expert end of main routine. DON'T MODIFY THIS CODE!!! */
    for(;;) {}
    /* Processor Expert end of main routine. DON'T WRITE CODE BELOW!!! */
} /* End of main routine. DO NOT MODIFY THIS TEXT!!! */
```

# Events.c

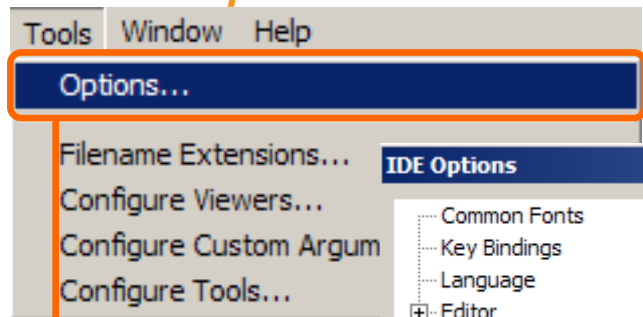
```
/*
** =====
**      Event      :  Cpu_OnNMIINT (module Events)
**      Component   :  Cpu [MKL26Z256MC4]
** =====
*/
void Cpu_OnNMIINT (void)
{
    /* Write your code here ... */
}

/*
** =====
**      Event      :  TU1_OnCounterRestart (module Events)
**      Component   :  TU1 [TimerUnit_LDD]
** =====
*/
void TU1_OnCounterRestart (LDD_UserData *UserDataPtr)
{
    /* Write your code here ... */
    LED_NegVal(NULL);
}
```

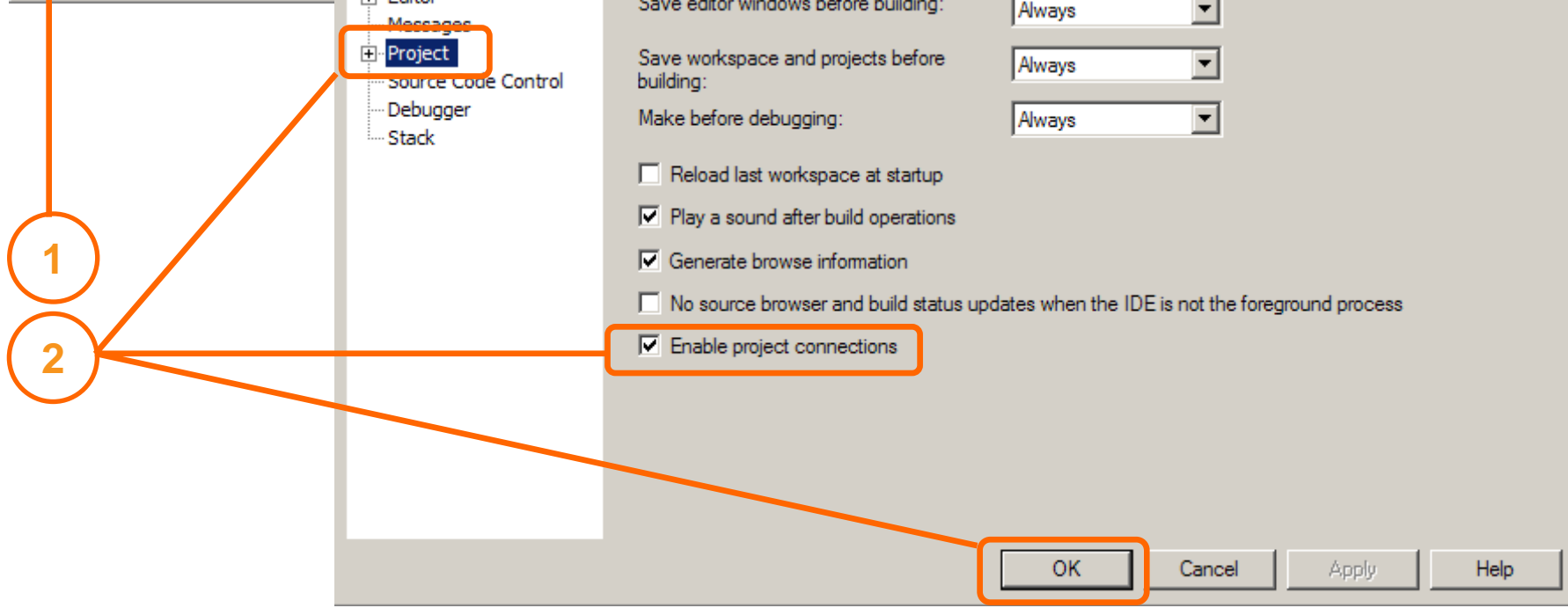


# Enable project connections in EWARM

*Tools → Options...*

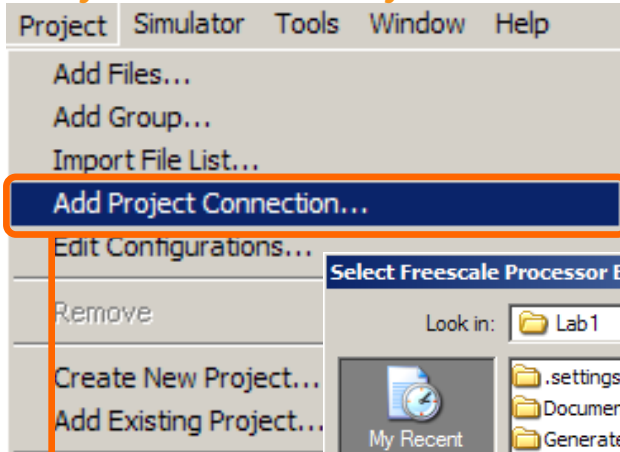


*Project → Enable project connections*



# Add project connection to PEx

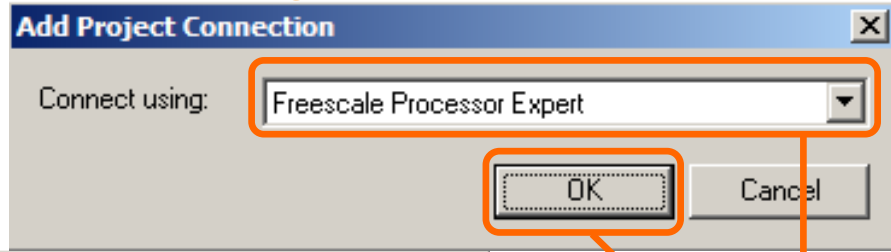
## Project → Add Project Connection...



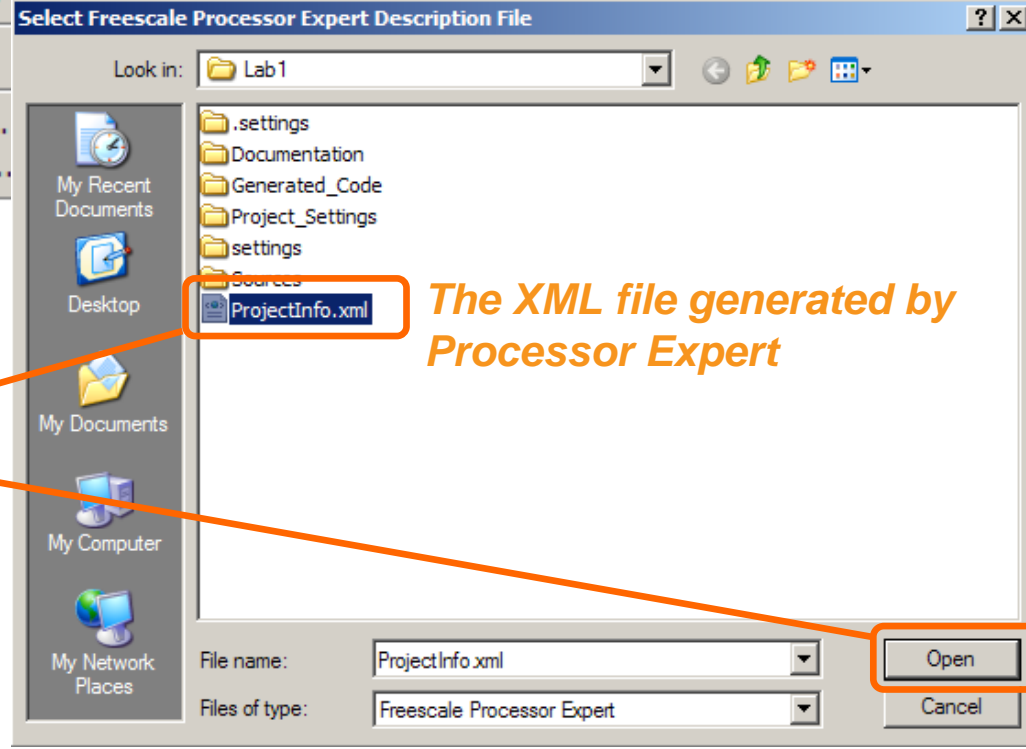
1

3

## Connect using Freescale Processor Expert

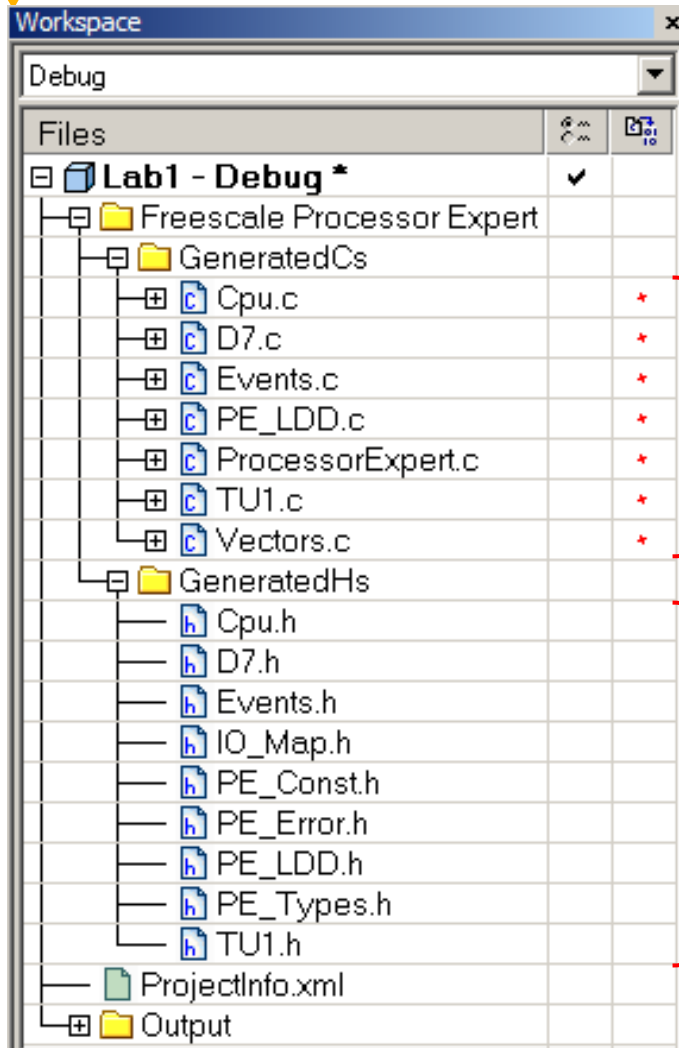


2



The XML file generated by Processor Expert

# What did Project Connection do?



## *Add Files and Groups:*

*The C source files and header files generated by Freescal Processor Expert have been automatically added into the project of IAR Embedded Workbench, in the group “GeneratedCs” and “GeneratedHs” respectively.*

# What did Project Connection do?

## Type of device:

Category:

General Options

C/C++ Compiler

Assembler

Output Converter

Custom Build

Build Actions

Linker

Debugger

Simulator

Angel

CMSIS DAP

GDB Server

Target | Output | Library Configuration | Library Options | M

Processor variant

Core Cortex-M0+

Device Freescale MKL26Z128xxx4

Category:

General Options

C/C++ Compiler

Assembler

Output Converter

Custom Build

Build Actions

Linker

Debugger

Simulator

Angel

CMSIS DAP

## Include directories of compiler:

Factory Settings

Multi-file Compilation

Discard Unused Publics

Code | Optimizations | Output | List | Preprocessor | Diagnostics

Ignore standard include directories

Additional include directories: (one per line)

C:\Freescale\PExDrv v10.2\eclipse\ProcessorExpert\lib\Kinetis\

C:\Freescale\PExDrv v10.2\eclipse\ProcessorExpert\lib\Kinetis\

C:\Documents and Settings\Ryan.Sheng.IARSYSTEMS\Desktop

C:\Documents and Settings\Ryan.Sheng.IARSYSTEMS\Desktop

## Include directories of assembler:

Category:

General Options

C/C++ Compiler

Assembler

Output Converter

Custom Build

Build Actions

Linker

Debugger

Simulator

Angel

CMSIS DAP

GDB Server

IAR ROM-monitor

Language | Output | List | Preprocessor | Diagnostics

Ignore standard include directories

Additional include directories: (one per line)

C:\Freescale\PExDrv v10.2\eclipse\ProcessorExpert\lib\

C:\Freescale\PExDrv v10.2\eclipse\ProcessorExpert\lib\

C:\Documents and Settings\Ryan.Sheng.IARSYSTEMS\Desktop

C:\Documents and Settings\Ryan.Sheng.IARSYSTEMS\Desktop

Category:

General Options

C/C++ Compiler

Assembler

Output Converter

Custom Build

Build Actions

Linker

Debugger

Simulator

Angel

CMSIS DAP

GDB Server

## Linker configuration file:

Factory Settings

Config | Library | Input | Optimizations | Advanced | Output | List

Linker configuration file

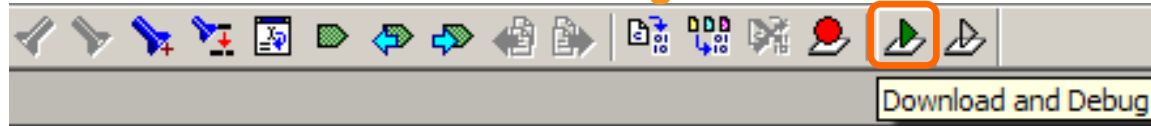
Override default

\_lab\Lab1\Project\_Settings\Linker\_Files\ProcessorExpert.icf

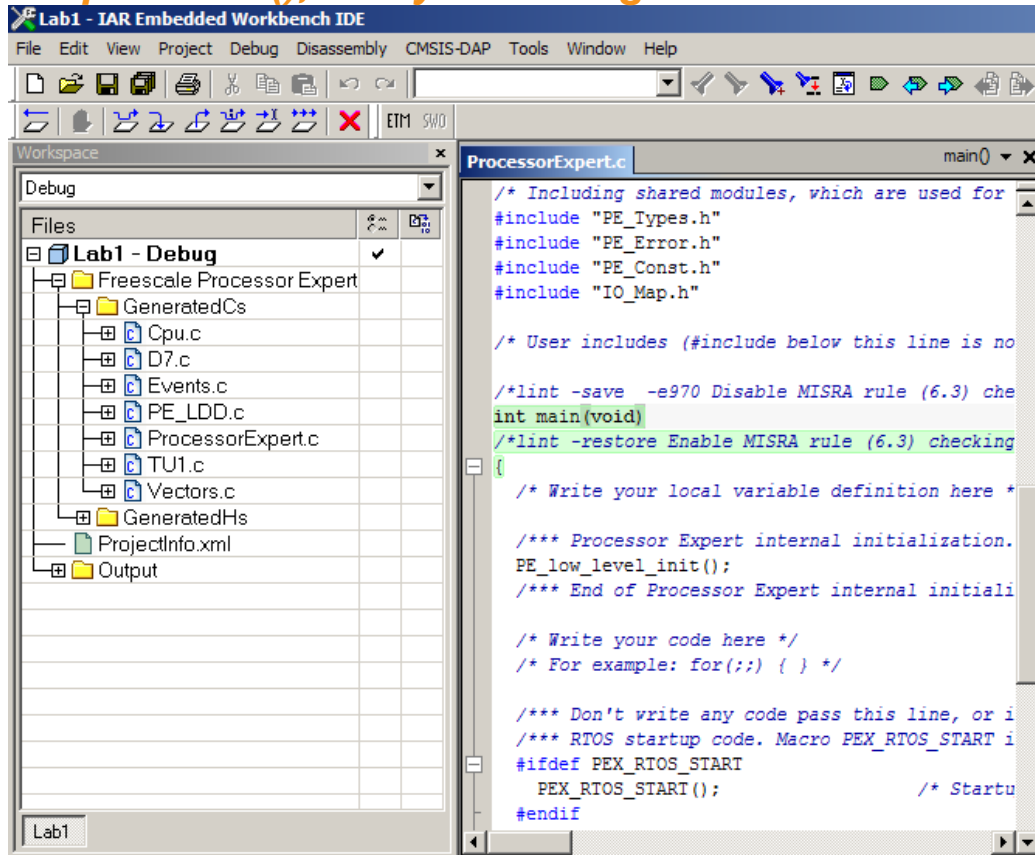
Edit...

# Download and debug

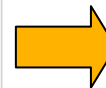
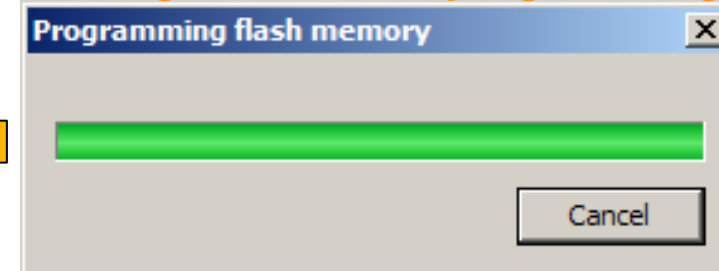
Press the "Download and Debug" button on the toolbar



Stop at main(), ready for debug



Waiting for the flash programming

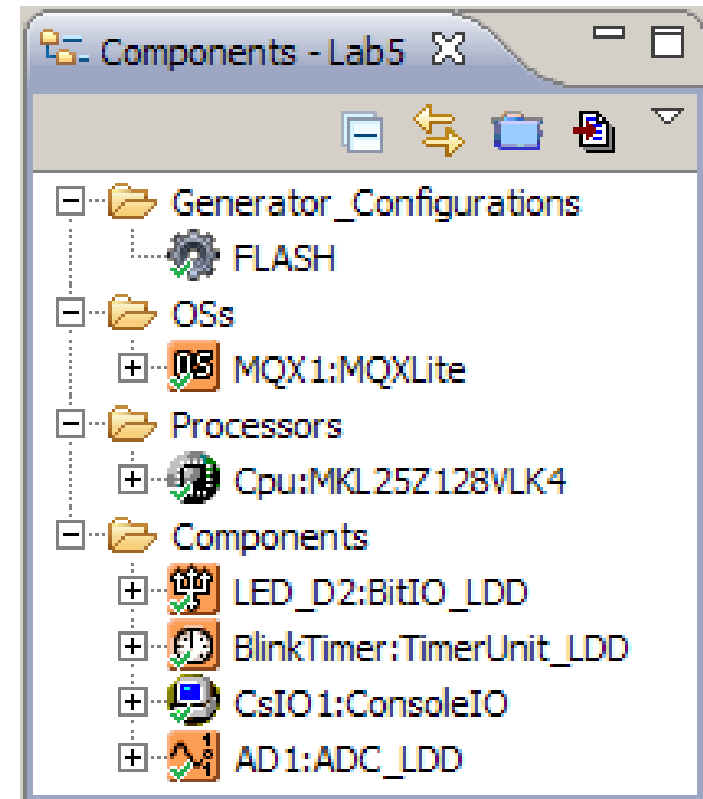


Press F5 to execute the program

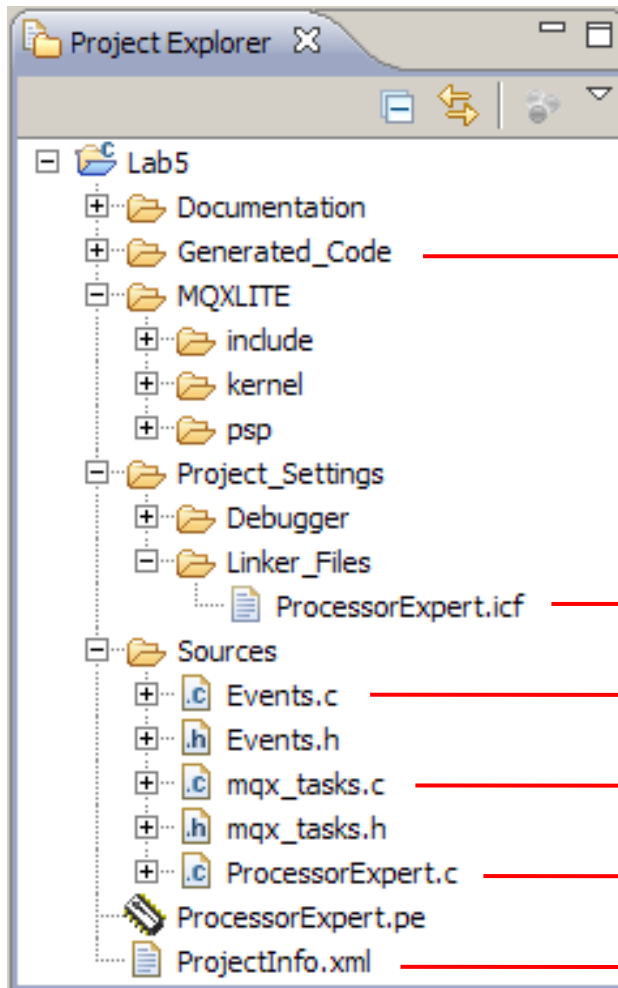
# Debug a Multi-Task Application by the MQX Kernel Awareness Plugin

## Example 2

- Implement a multi-task application by the MQX-Lite RTOS
  - Timer interrupt
    - Blink a LED
    - Start an A/D convert
  - A/D interrupt
    - Post semaphore 1
  - Task 1
    - Wait for semaphore 1
    - Read the result of A/D convert
    - Post semaphore 2
  - Task 2
    - Wait for semaphore 2
    - Output A/D convert results to the serial console



# Processor Expert generated code



*Drivers, initialization code and vector table*

*MQX-Lite kernel and PSP modules*

*Linker configuration file for IAR EWARM*

*User's event handler module*

*Tasks implementation*

*User's application code, e.g. main()*

*Project information (XML format)*



# ProcessorExpert.c

```
/* #####  
**      Filename : ProcessorExpert.c  
**      #####*/  
  
/* User includes (#include below this line is not maintained by Processor Expert) */  
LDD_TDeviceData *pAD1_dev;          /* pointer to the A/D converter device structure */  
LWSEM_STRUCT sem1, sem2;           /* light-weight semaphores */  
  
int main (void)  
{  
    .....  
    /* Write your code here */  
    _lwsem_create(&sem1, 0);        /* create light-weight semaphore 1 */  
    _lwsem_create(&sem2, 0);        /* create light-weight semaphore 2 */  
  
    pAD1_dev = AD1_Init(NULL);      /* A/D converter initialization */  
    AD1_SelectSampleGroup(pAD1_dev, 0);  
    .....  
} /* End of main routine. DO NOT MODIFY THIS TEXT!!! */
```

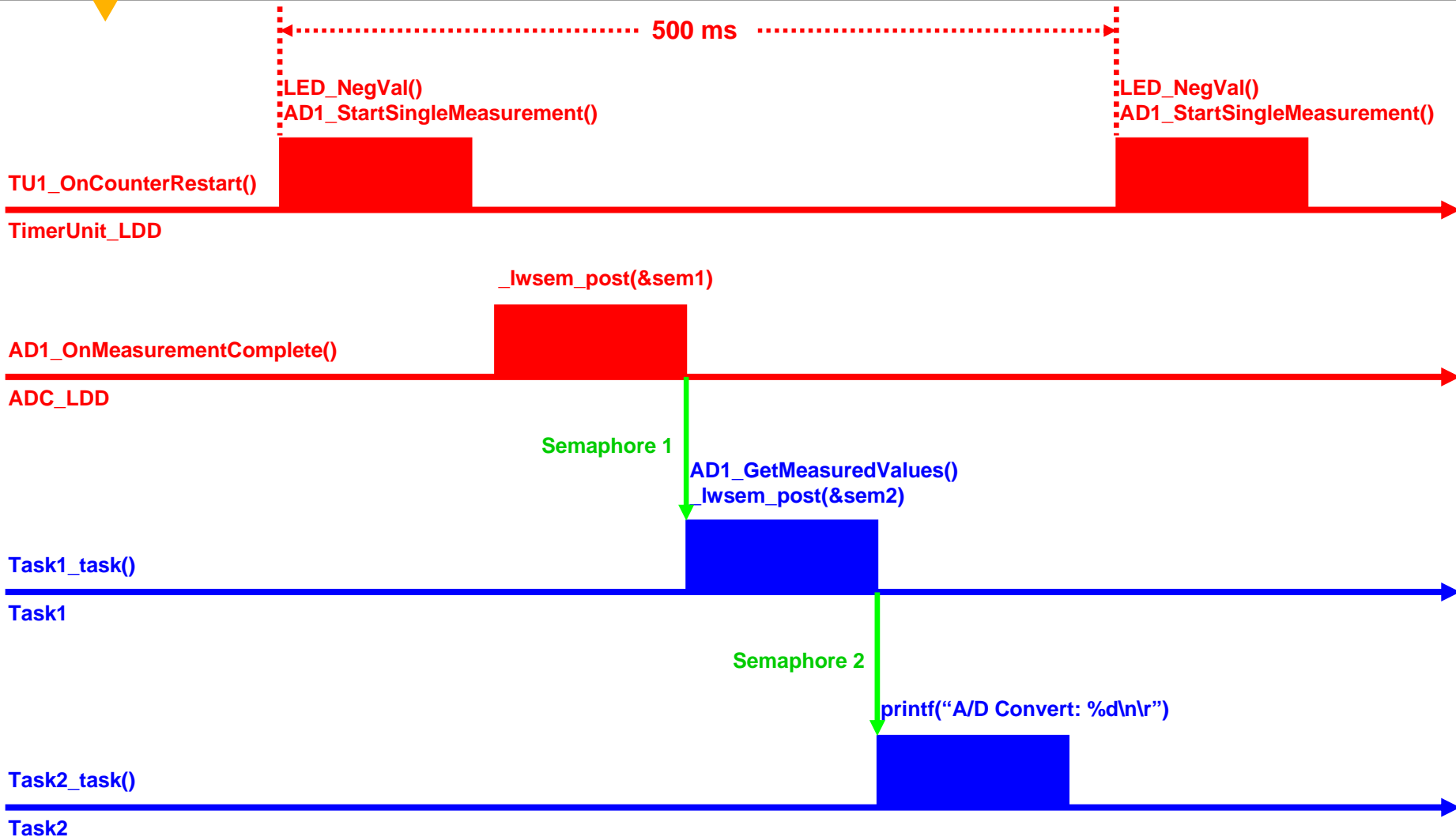
# Events.c

```
/* #####  
**  Filename : Events.c  
**  #####*/  
  
//  Event      : TU1_OnCounterRestart (module Events)  
//  Component   : TU1 [TimerUnit_LDD]  
extern LDD_TDeviceData *pAD1_dev;  
void TU1_OnCounterRestart(LDD_TUserData *UserDataPtr)  
{  
    /* Write your code here ... */  
    LED_NegVal(NULL);                /* Toggle the LED */  
    AD1_StartSingleMeasurement(pAD1_dev); /* Start a single A/D measurement */  
}  
  
//  Event      : AD1_OnMeasurementComplete (module Events)  
//  Component   : AD1 [ADC_LDD]  
extern LWSEM_STRUCT sem1;  
void AD1_OnMeasurementComplete(LDD_TUserData *UserDataPtr)  
{  
    /* Write your code here ... */  
    _lwssem_post(&sem1);            /* Post semaphore 1 to active task 1 */  
}
```

# mqx\_tasks.c

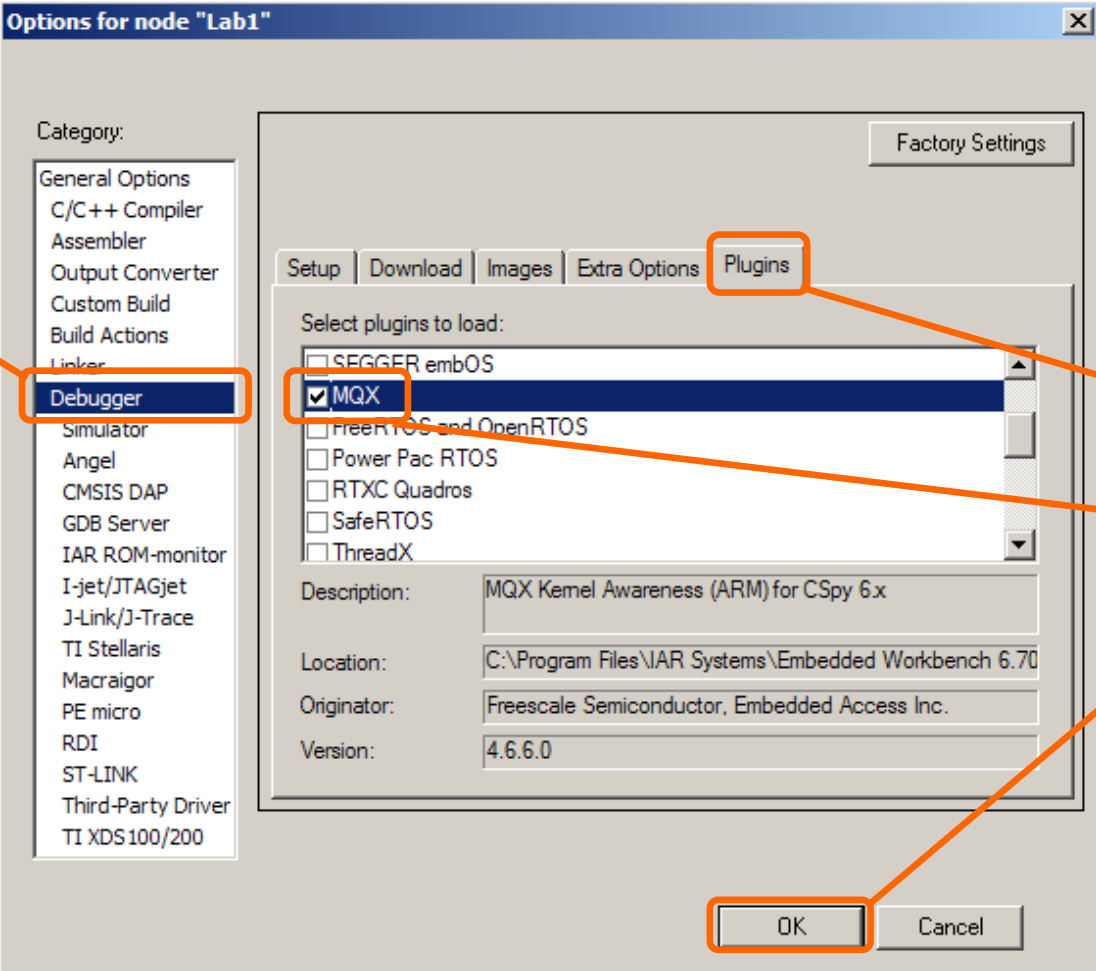
```
/* #####  
**      Filename : mqx_tasks.c  
**      #####*/  
  
/* User includes (#include below this line is not maintained by Processor Expert) */  
#include <stdio.h>                                /* declaration of printf() */  
extern LDD_TDeviceData *pAD1_dev;  
extern LWSEM_STRUCT sem1, sem2;  
volatile unsigned short AD1_data;                /* A/D convert value */  
//Event : Task1_task (module mqx_tasks)  
void Task1_task (uint32_t task_init_data) {  
    while(1) {  
        _lwsem_wait(&sem1);                        /* Wait for semaphore 1 */  
        AD1_GetMeasuredValues(pAD1_dev, (void *)&AD1_data); /* Read A/D convert value */  
        _lwsem_post(&sem2);                        /* Post semaphore 2 to active task 2 */  
    }  
}  
//Event : Task2_task (module mqx_tasks)  
void Task2_task (uint32_t task_init_data) {  
    while(1) {  
        _lwsem_wait(&sem2);                        /* Wait for semaphore 2 */  
        printf("AD Convert: %d\n\r", AD1_data);    /* Print A/D convert value to serial console */  
    }  
}
```

# Synchronization of Tasks and Interrupts



# Enable the MQX debugger plugin

*Debugger → Plugins → MQX*



Options for node "Lab1"

Category:

- General Options
- C/C++ Compiler
- Assembler
- Output Converter
- Custom Build
- Build Actions
- Linker
- Debugger**
- Simulator
- Angel
- CMSIS DAP
- GDB Server
- IAR ROM-monitor
- I-jet/JTAGjet
- J-Link/J-Trace
- TI Stellaris
- Macraigor
- PE micro
- RDI
- ST-LINK
- Third-Party Driver
- TI XDS100/200

Factory Settings

Setup | Download | Images | Extra Options | **Plugins**

Select plugins to load:

- SEGGER embOS
- MQX**
- FreeRTOS and OpenRTOS
- Power Pac RTOS
- RTX C Quadros
- SafeRTOS
- ThreadX

Description: MQX Kernel Awareness (ARM) for CSpy 6.x

Location: C:\Program Files\IAR Systems\Embedded Workbench 6.70

Originator: Freescale Semiconductor, Embedded Access Inc.

Version: 4.6.6.0

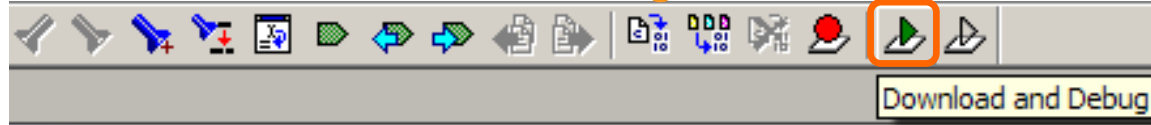
**OK** Cancel

1 2 3 4

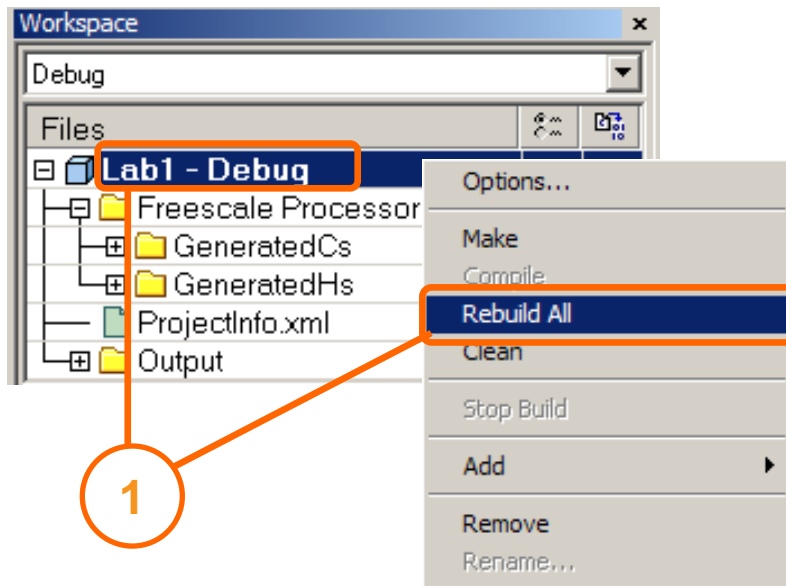
# Rebuild, download and debug

2

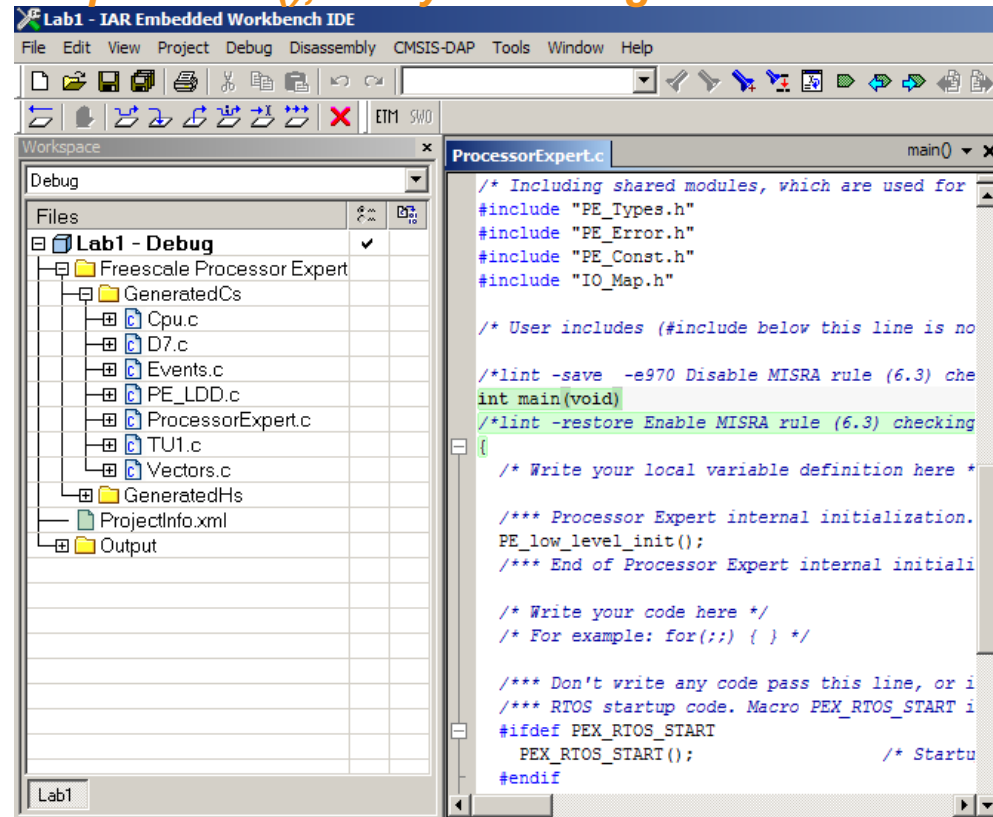
Press the “Download and Debug” button on the toolbar



Right-click on the project name and select “Rebuild All” in the context menu.

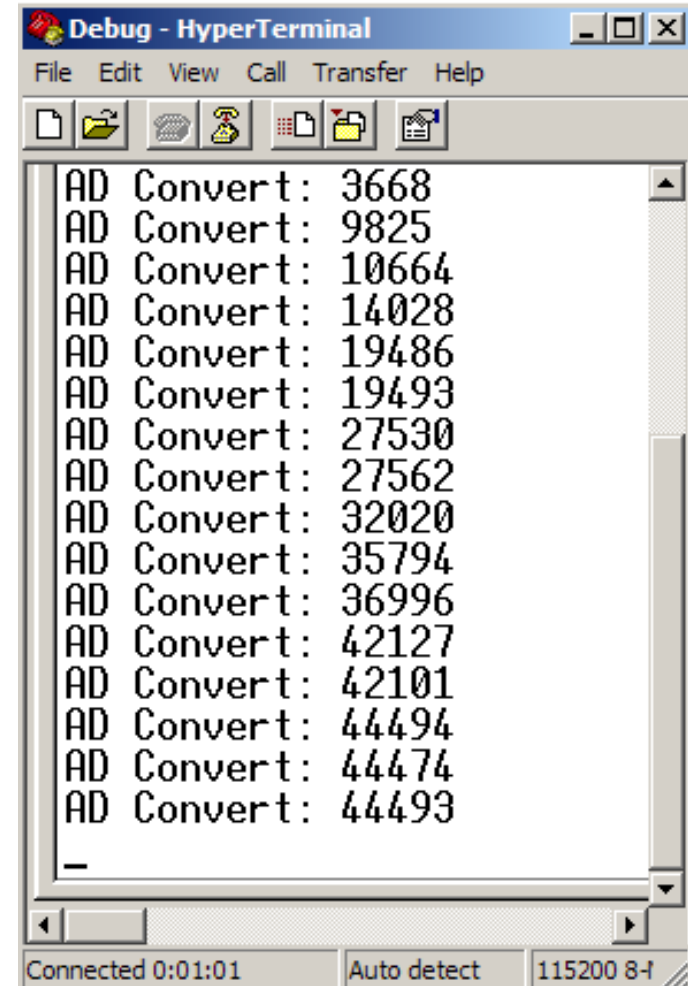
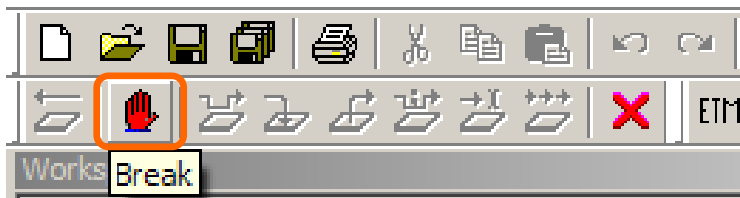


Stop at main(), ready for debug



# Execute the program

- Press F5 to execute the program
  - The LED **D7** should blink
  - Turn the potentiometer **VR1** to check the change on A/D convert values in the serial console.
- Press the “Break” button to stop the execution of program



# MQX kernel awareness

MQX	RTCS	Tools	Window	Help
Task List				
Kernel Data				
Check for Errors				
Task Summary				
Ready Queues				
Stack Usage				
Memory Pools				
Memory Blocks				
Memory Extension Blocks				
Lightweight Memory Pools				
Lightweight Memory Blocks				
Partition Summary				
Message Queues				
Message Pools				
Lightweight Message Queues				
Lightweight Events				
Lightweight Semaphores				
Events				
Mutexes				
Semaphores				
Task Queues				

## Task List

A	Name	ID	TD	Priority	State
→	_mqx_idle_task	0x10001	0x1fff8b8	9	Active
	task1	0x10002	0x1fff010	8	LW Semaphore Blocked
	task2	0x10003	0x1fff290	8	LW Semaphore Blocked
	NO TASK				

## Light-weight Semaphores

LWSem	Valid	Value	Waiting#	Symbol
0x1fff684	Yes	1	0	_mqx_kernel_data_struct.
0x1fff604	Yes	1	0	_mqx_kernel_data_struct.
0x1fffc78	Yes	0	1	sem1
0x1fffc94	Yes	0	1	sem2

## Kernel Data

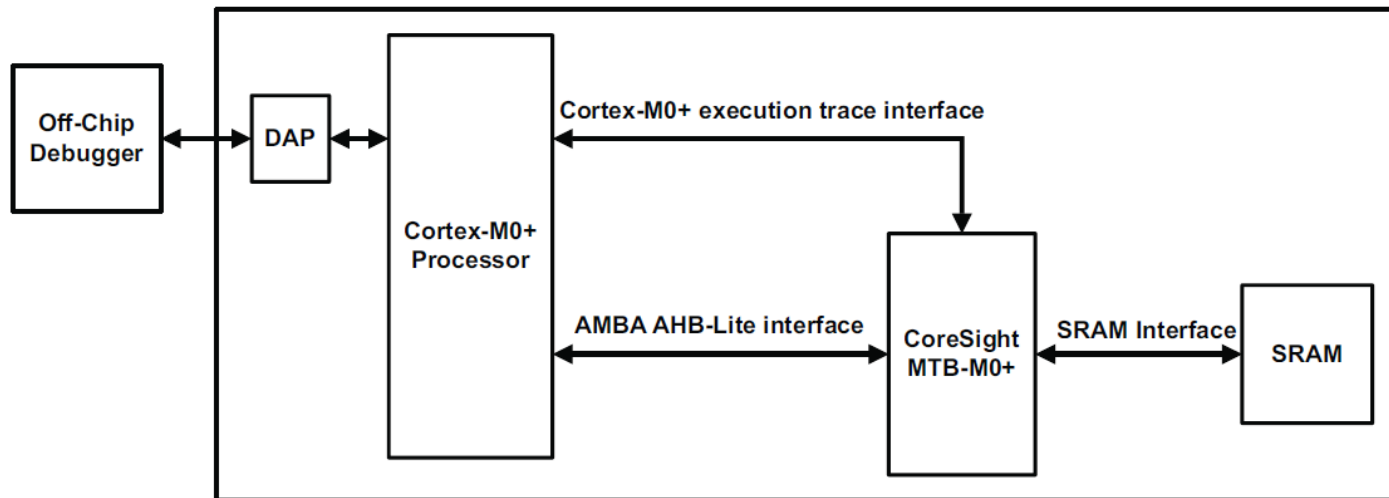
Version:	4.0.0
Active Task:	_mqx_idle_task, ID:0x10001, TD:0x1fff8b8
CPU Type:	MKLXX (MKL2X)
In an ISR?	No
Scheduler Policy:	FIFO
Current Time:	4.239 seconds
Time Offset:	0.000 seconds
Embedded IO:	I/O Subsystem, RTCS, SNMP, MFS, USB, Shell, EDS



# Instruction Trace Debugging by the Micro Trace Buffer of Kinetis L MCU

# Instruction trace techniques

- ETM (Embedded Trace Macrocell)
  - Off-chip trace buffer (in emulator, 2~16 MB)
  - 4~32-bit data bus
  - High performance but expensive for both processor and emulator
- ETB (Embedded Trace Buffer)
  - On-chip dedicated trace buffer, no extra pins
- MTB (Micro Trace Buffer)
  - On-chip configurable (shared) trace buffer, no extra pins



# MTB debugging on Kinetis L

Workspace

Debug

Files

- int\_gkis.c
- int\_inst.c
- int\_kisr.c
- int\_pvt.a.c
- int\_unx.c
- int\_vtab.c
- int\_xcpt.c
- IO1.c
- klog.c
- lwevent.c
- lwlog.c
- lwmsgq.c
- lwsem.c
- lwtimer.c
- mem\_zero.c
- MQX1.c
- mqx\_tasks.c**
- mqx\_utils.c
- mqxlite.c
- mutex.c
- PE\_LDD.c
- Processor...

mqx\_tasks.c

```
** =====  
**      Event      : Task2_task (module mqx_tasks)  
**  
**      Component  : Task2 [MQXLite_task]  
**      Description:  
**          MQX task routine. The routine is generated into mqx_tasks.  
**          file.  
**      Parameters :  
**          NAME          - DESCRIPTION  
**          task_init_data -  
**      Returns      : Nothing  
** =====  
*/  
void Task2_task(uint32_t task_init_data)  
{  
    int counter = 0;  
    while(1) {  
        counter++;  
  
        /* Write your code here ... */  
        _lwsem_wait(&sem2);  
        printf("AD Convert: %d\n\r", AD1_data);  
    }  
}
```

Task2\_task(uint32\_t)

1

2

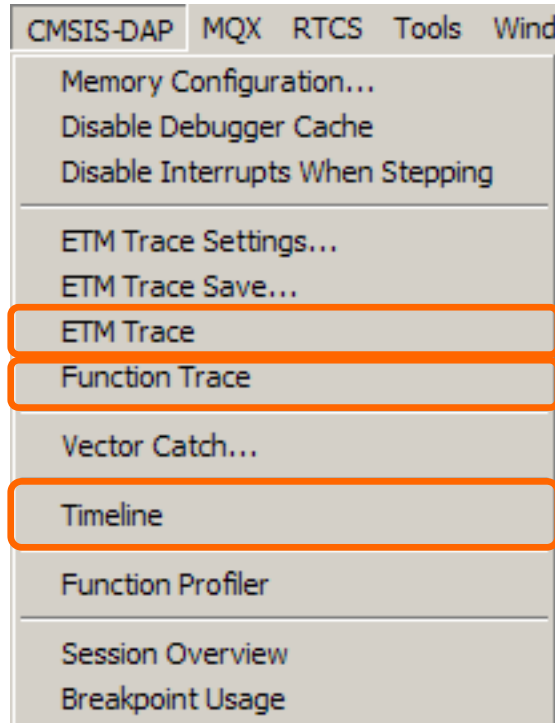
3

Find the function Task2\_task(...).

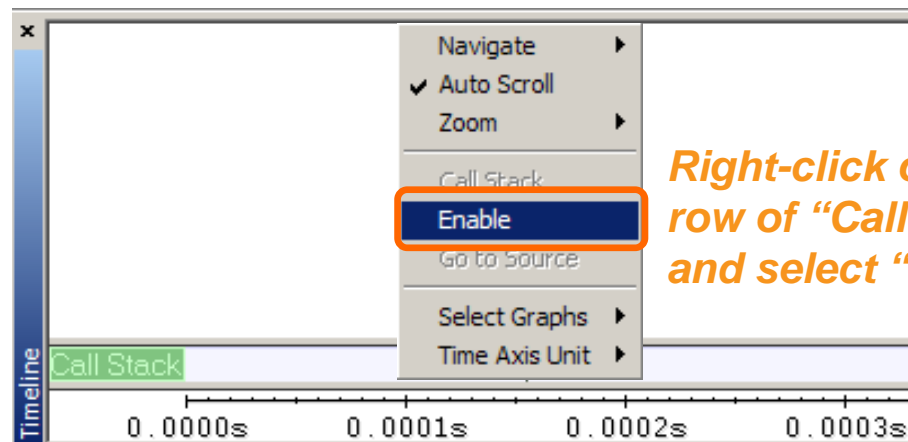
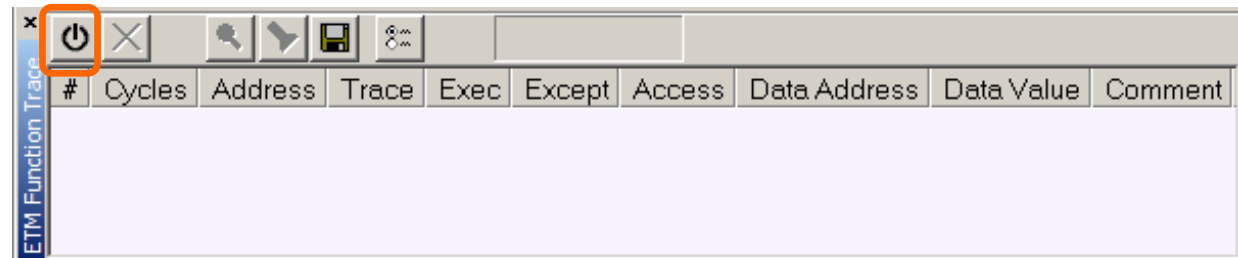
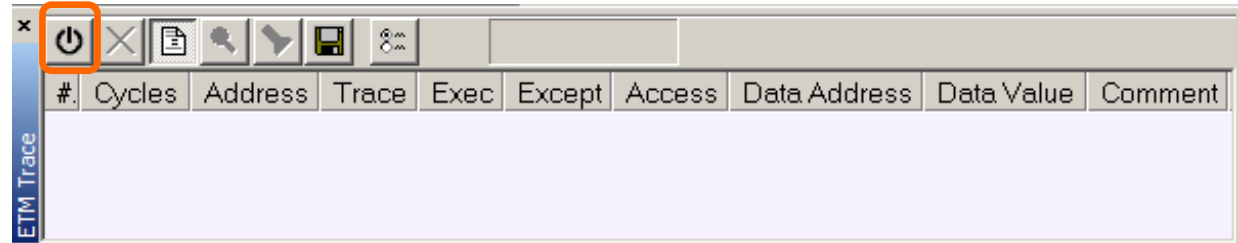
Click on the line of printf(...) and press F9 to set a breakpoint on it.

Lab1

# Open trace windows



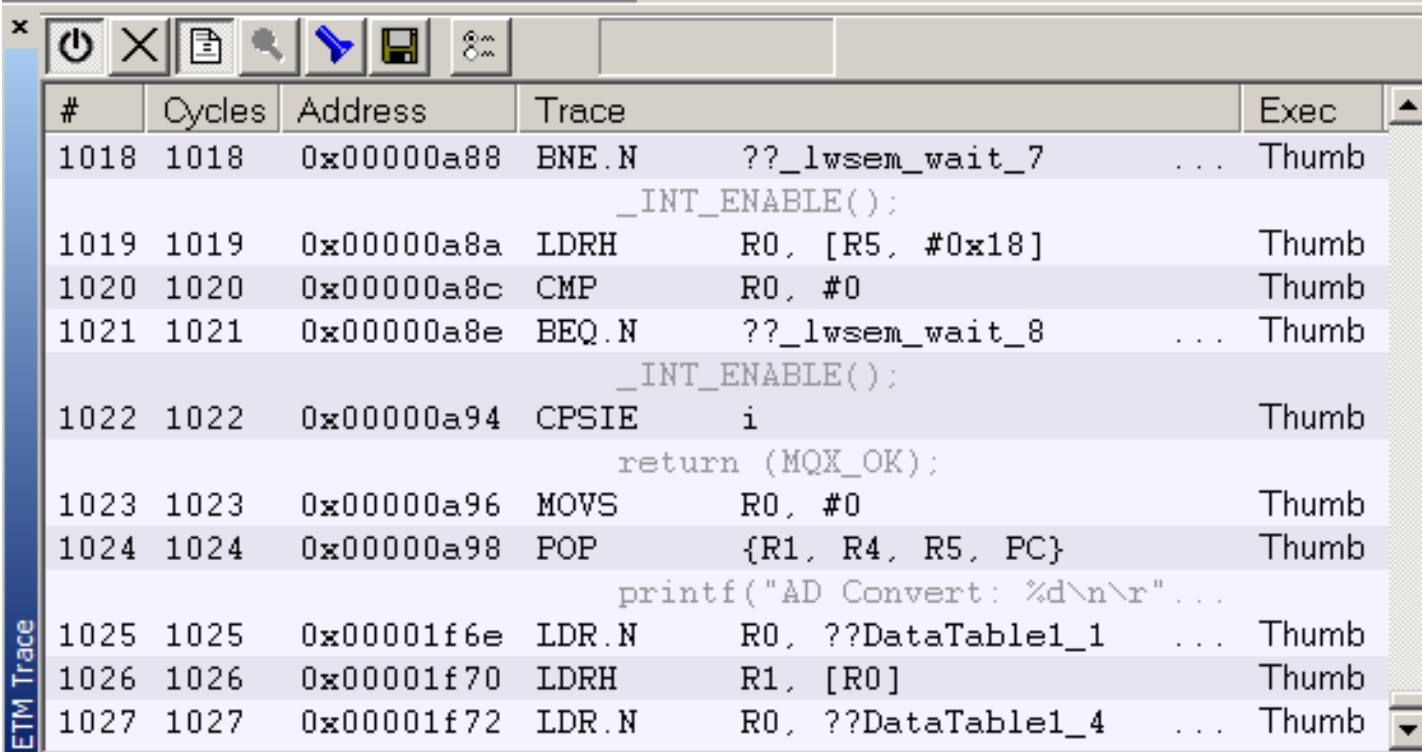
## Enable



*Right-click on the row of "Call Stack" and select "Enable"*

# Collect executed instructions

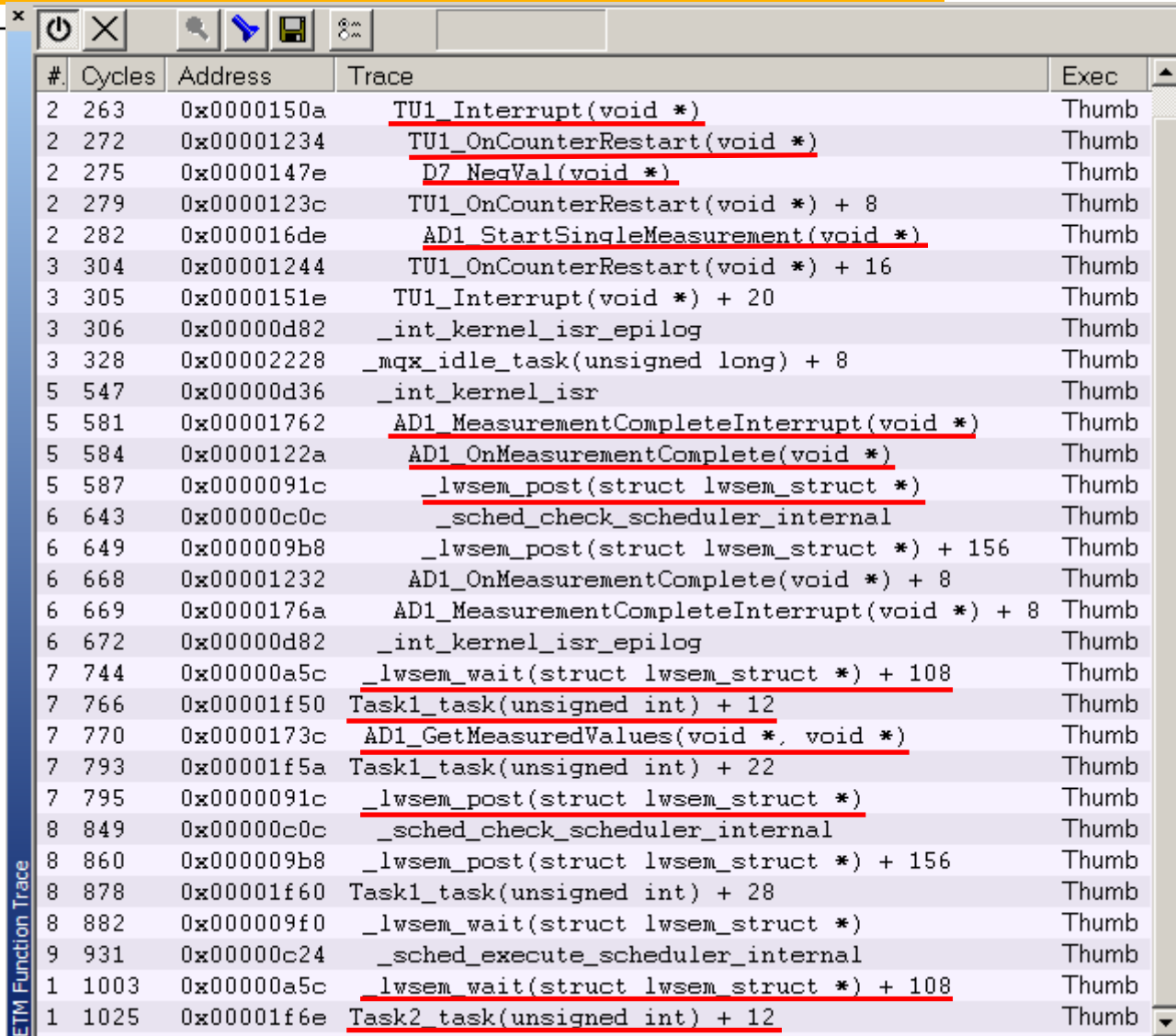
- Press F5 to execute the program, until the breakpoint is hit.
- Go to the bottom of “ETM Trace” window to check the recorded instructions together with mixed C source.



#	Cycles	Address	Trace	Exec
1018	1018	0x00000a88	BNE.N ??_lwsem_wait_7 ... _INT_ENABLE();	Thumb
1019	1019	0x00000a8a	LDRH R0, [R5, #0x18]	Thumb
1020	1020	0x00000a8c	CMP R0, #0	Thumb
1021	1021	0x00000a8e	BEQ.N ??_lwsem_wait_8 ... _INT_ENABLE();	Thumb
1022	1022	0x00000a94	CPSIE i return (MOX_OK);	Thumb
1023	1023	0x00000a96	MOVS R0, #0	Thumb
1024	1024	0x00000a98	POP {R1, R4, R5, PC} printf("AD Convert: %d\n\r"...	Thumb
1025	1025	0x00001f6e	LDR.N R0, ??DataTable1_1 ...	Thumb
1026	1026	0x00001f70	LDRH R1, [R0]	Thumb
1027	1027	0x00001f72	LDR.N R0, ??DataTable1_4 ...	Thumb

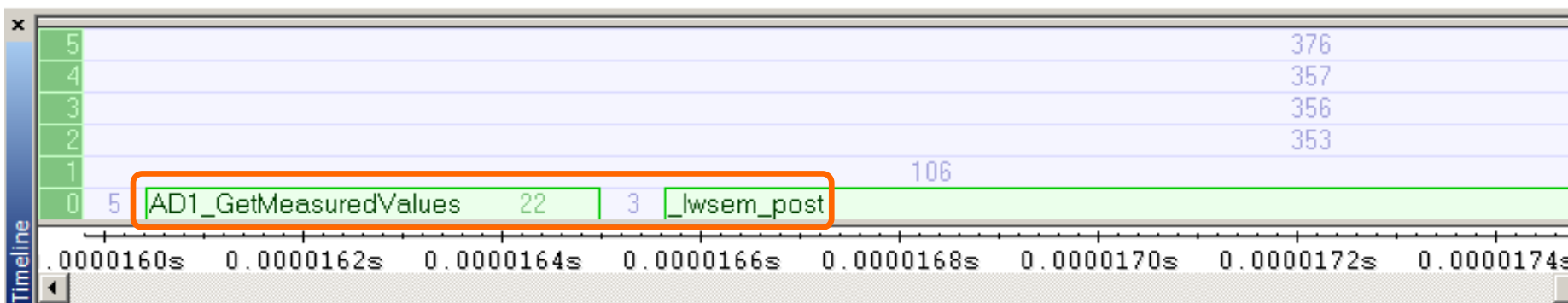
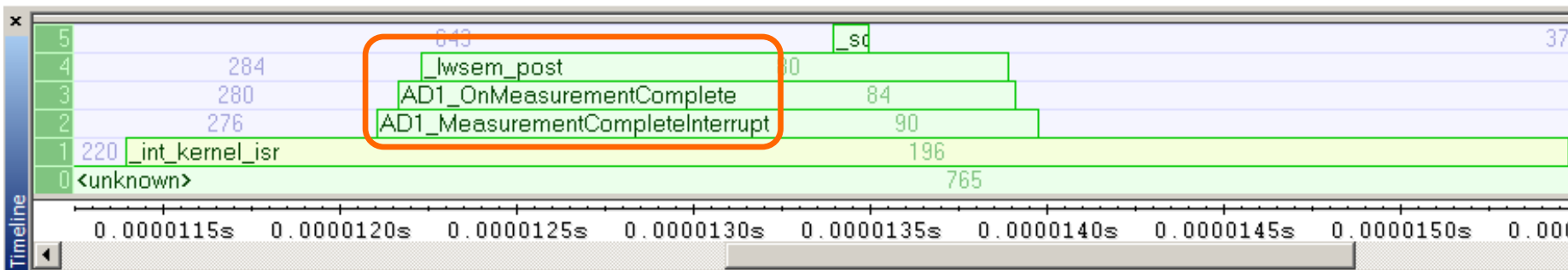
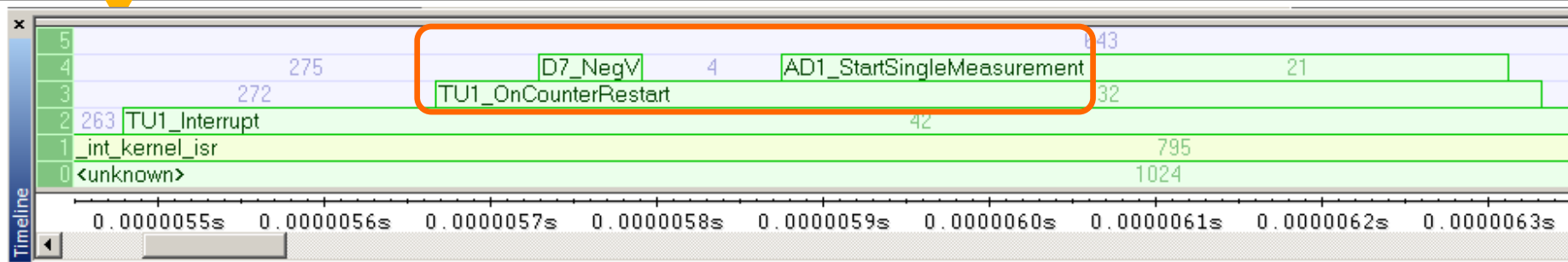
# View the trace data at function-level

- Go to the bottom of “ETM Function Trace” window to find the trace data in function-level.
- Check the actual sequence of ISRs and tasks during execution.



#	Cycles	Address	Trace	Exec
2	263	0x0000150a	<u>TU1_Interrupt(void *)</u>	Thumb
2	272	0x00001234	<u>TU1_OnCounterRestart(void *)</u>	Thumb
2	275	0x0000147e	<u>D7_NeqVal(void *)</u>	Thumb
2	279	0x0000123c	TU1_OnCounterRestart(void *) + 8	Thumb
2	282	0x000016de	<u>AD1_StartSingleMeasurement(void *)</u>	Thumb
3	304	0x00001244	TU1_OnCounterRestart(void *) + 16	Thumb
3	305	0x0000151e	TU1_Interrupt(void *) + 20	Thumb
3	306	0x00000d82	_int_kernel_isr_epilog	Thumb
3	328	0x00002228	_mqx_idle_task(unsigned long) + 8	Thumb
5	547	0x00000d36	_int_kernel_isr	Thumb
5	581	0x00001762	<u>AD1_MeasurementCompleteInterrupt(void *)</u>	Thumb
5	584	0x0000122a	<u>AD1_OnMeasurementComplete(void *)</u>	Thumb
5	587	0x0000091c	<u>_lwsem_post(struct lwsem_struct *)</u>	Thumb
6	643	0x00000c0c	_sched_check_scheduler_internal	Thumb
6	649	0x000009b8	_lwsem_post(struct lwsem_struct *) + 156	Thumb
6	668	0x00001232	AD1_OnMeasurementComplete(void *) + 8	Thumb
6	669	0x0000176a	AD1_MeasurementCompleteInterrupt(void *) + 8	Thumb
6	672	0x00000d82	_int_kernel_isr_epilog	Thumb
7	744	0x00000a5c	<u>_lwsem_wait(struct lwsem_struct *) + 108</u>	Thumb
7	766	0x00001f50	<u>Task1_task(unsigned int) + 12</u>	Thumb
7	770	0x0000173c	<u>AD1_GetMeasuredValues(void *, void *)</u>	Thumb
7	793	0x00001f5a	Task1_task(unsigned int) + 22	Thumb
7	795	0x0000091c	<u>_lwsem_post(struct lwsem_struct *)</u>	Thumb
8	849	0x00000c0c	_sched_check_scheduler_internal	Thumb
8	860	0x000009b8	_lwsem_post(struct lwsem_struct *) + 156	Thumb
8	878	0x00001f60	Task1_task(unsigned int) + 28	Thumb
8	882	0x000009f0	_lwsem_wait(struct lwsem_struct *)	Thumb
9	931	0x00000c24	_sched_execute_scheduler_internal	Thumb
1	1003	0x00000a5c	<u>_lwsem_wait(struct lwsem_struct *) + 108</u>	Thumb
1	1025	0x00001f6e	<u>Task2_task(unsigned int) + 12</u>	Thumb

# Graphical call stack in Timeline window



# IAR Systems: Your strategic partner



- Different architecture, one solution
- Most widely used tool chain for ARM MCU
- Supporting almost all FSL cores / devices
- Efficient & High performance code
- Freescale MQX™ RTOS integration
- Freescale Processor Expert integration
- 3<sup>rd</sup>-party emulators and RTOS support
- Advanced trace debugging
- Power debugging
- Functional safety certificate
- Global professional technical support

