Telecom Design – TD1208 Development Environment

Sylvain WORMSER – Business Development Manager – Avnet-Memec
Michel Stempin –Senior Engineer – Telecom Design

TELECOM DESIGN – TD1208
868MHZ SIGFOX™ CERTIFIED MODULE

One network A billion dreams

TELECOM DESIGN - TD1208 - CHARACTERISTICS

RF Features

- SIGFOX™ certified module
- ISM band 868 MHz
- Link budget 140 dBm
  - -126 dBm Rx
  - +14 dBm Tx
- Supported Modulation
  - (G)FSK, 4(G)FSK, GMSK
  - OOK

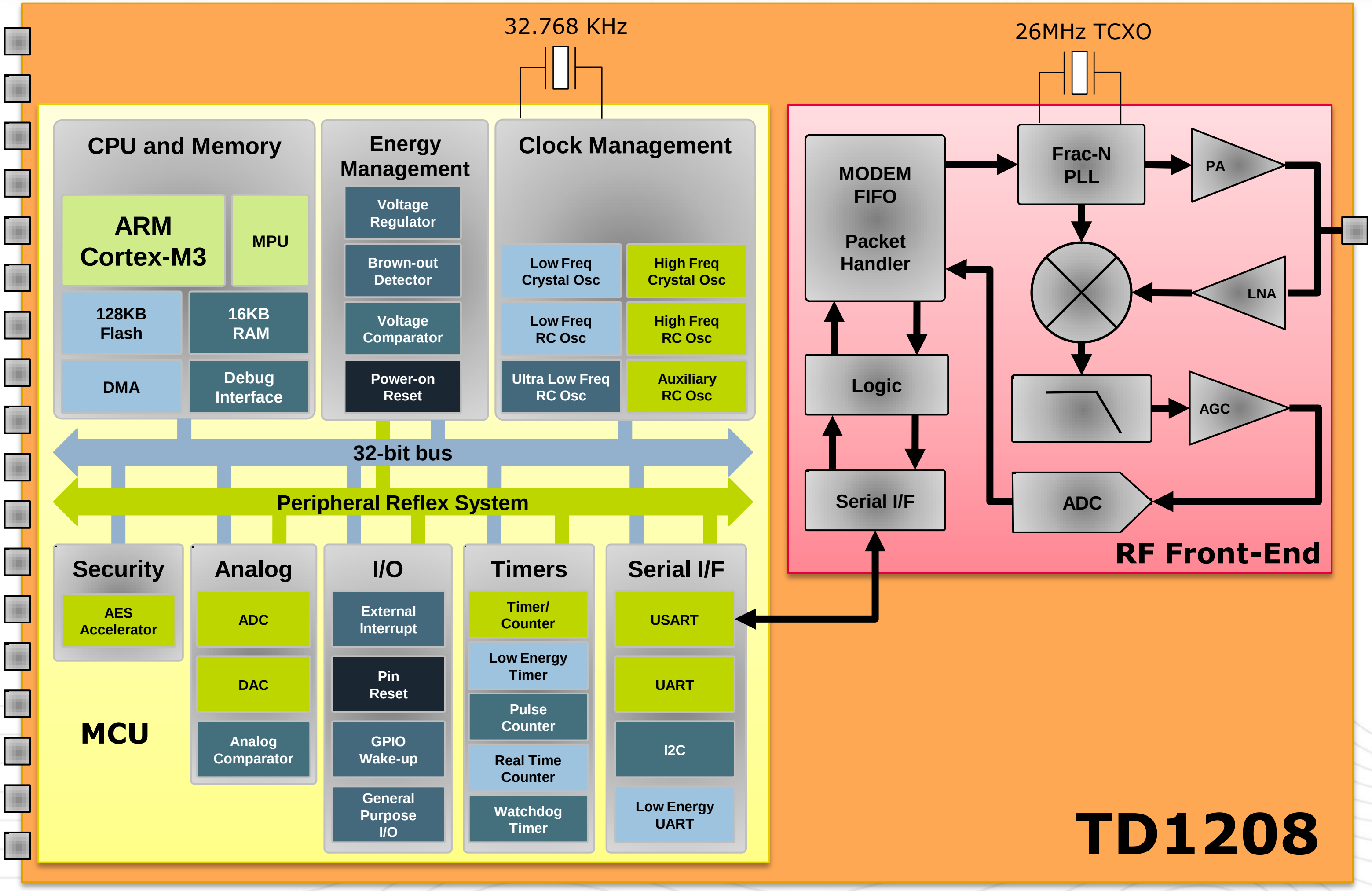
MCU Features

- ARM® Cortex-M3™
- 128KB Flash
- 16KB RAM
- AES-128/256
- 12-bit ADC, 12-bit DAC
- 16-bit Timers, RTC
- USART, SPI, I²C
- Application customizable

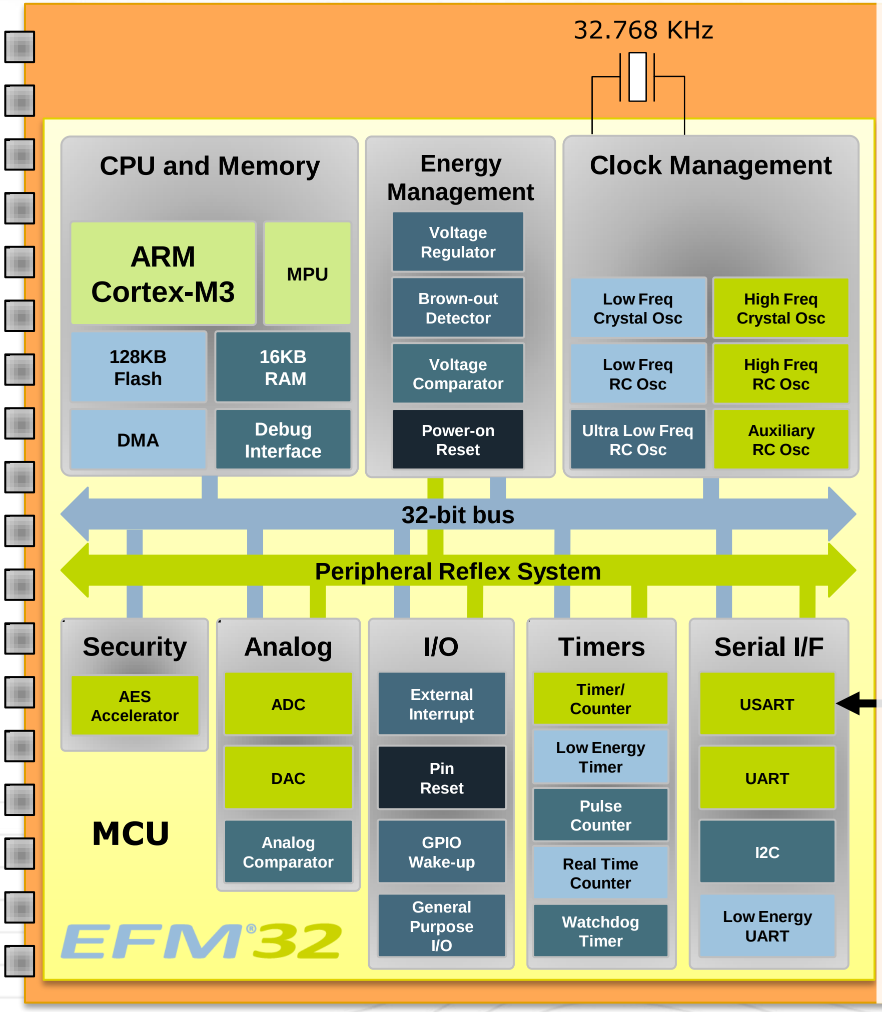
Module Features

- Low power consumption 13/16 mA Rx, 37 mA Tx @ +10 dBm
- 1.5µA Sleep mode with RTC, RAM retention and brown-out detector
- 2.3 to 3.3 V power supply
- 25.4×12.7×3.81mm

TELECOM DESIGN - TD1208



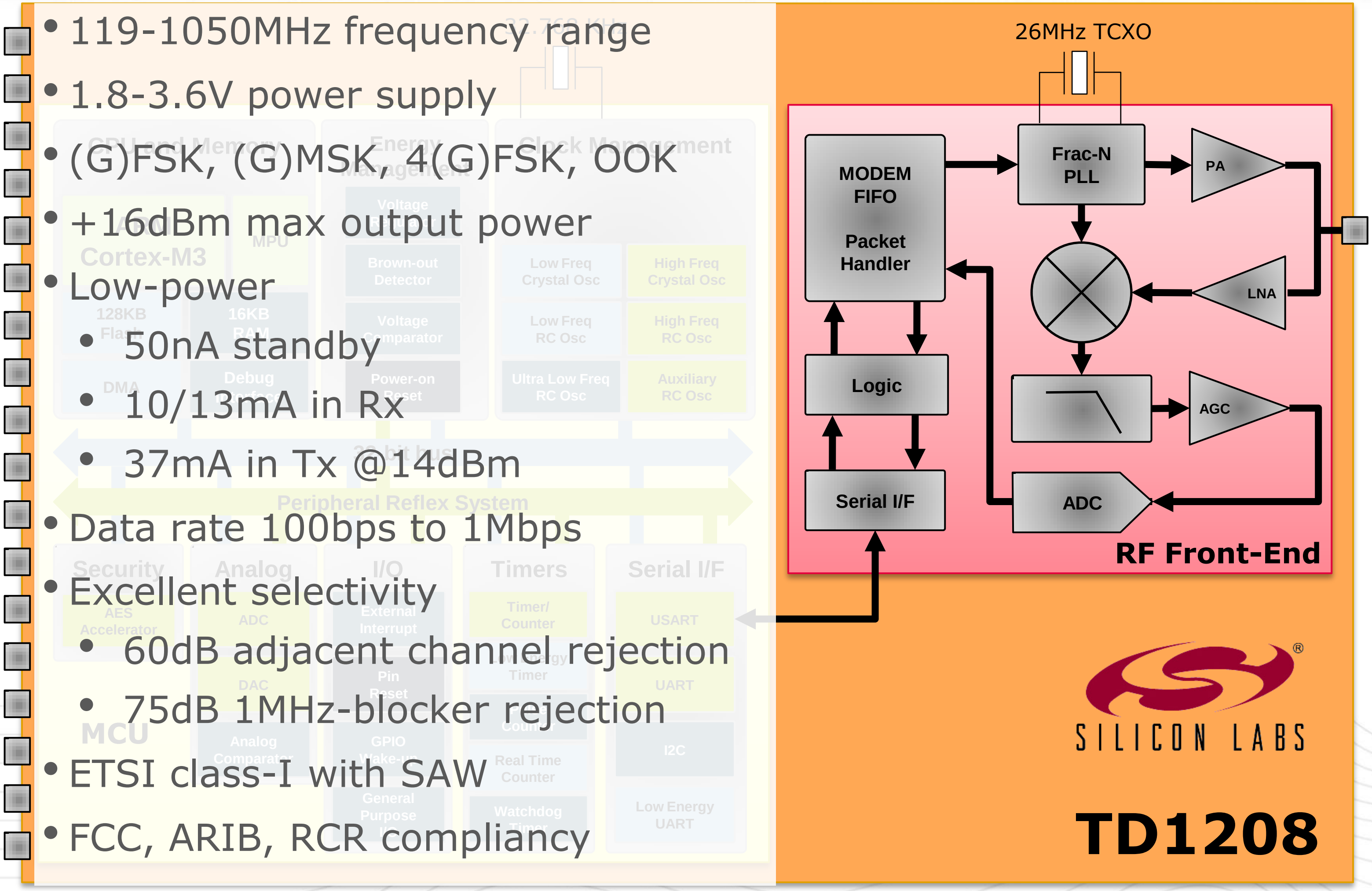
ENERGY MICRO EFM32 – ULTRA LOW POWER CORTEX-M3



- Ultra low power Energy modes
- Flexible clocks and oscillators
- 2-wire ARM SWD Debug I/F
- 2µs Fast-Wake-up time
- DMA and Peripheral Reflex System
- AES 128/256-bit Encryp./Decrypt
- Pulse Counter
- Low Energy UART
  - Functional up to EM3
- Low power 12-bit ADC 1Msps
- 12-bit DAC 500Ksps
- Firmware protection

EM0 (Run Mode)	EM1 (Sleep)	EM2 (Deep Sleep)	EM3 (Stop Mode)	EM4 (Shutoff Mode)

HIGH PERFORMANCE – LOW POWER RF



- 119-1050MHz frequency range
- 1.8-3.6V power supply
- (G)FSK, (G)MSK, 4(G)FSK, OOK
- +16dBm max output power
- Low-power
  - 50nA standby
  - 10/13mA in Rx
  - 37mA in Tx @14dBm
- Data rate 100bps to 1Mbps
- Excellent selectivity
  - 60dB adjacent channel rejection
  - 75dB 1MHz-blocker rejection
- ETSI class-I with SAW
- FCC, ARIB, RCR compliancy

EM0 (Run Mode)	EM1 (Sleep)	EM2 (Deep Sleep)	EM3 (Stop Mode)	EM4 (Shutoff Mode)

TD1208 DEVELOPMENT TOOLS

Hardware

OCD Probe

TD1208 Programming Kit

Compiler

IAR SYSTEMS

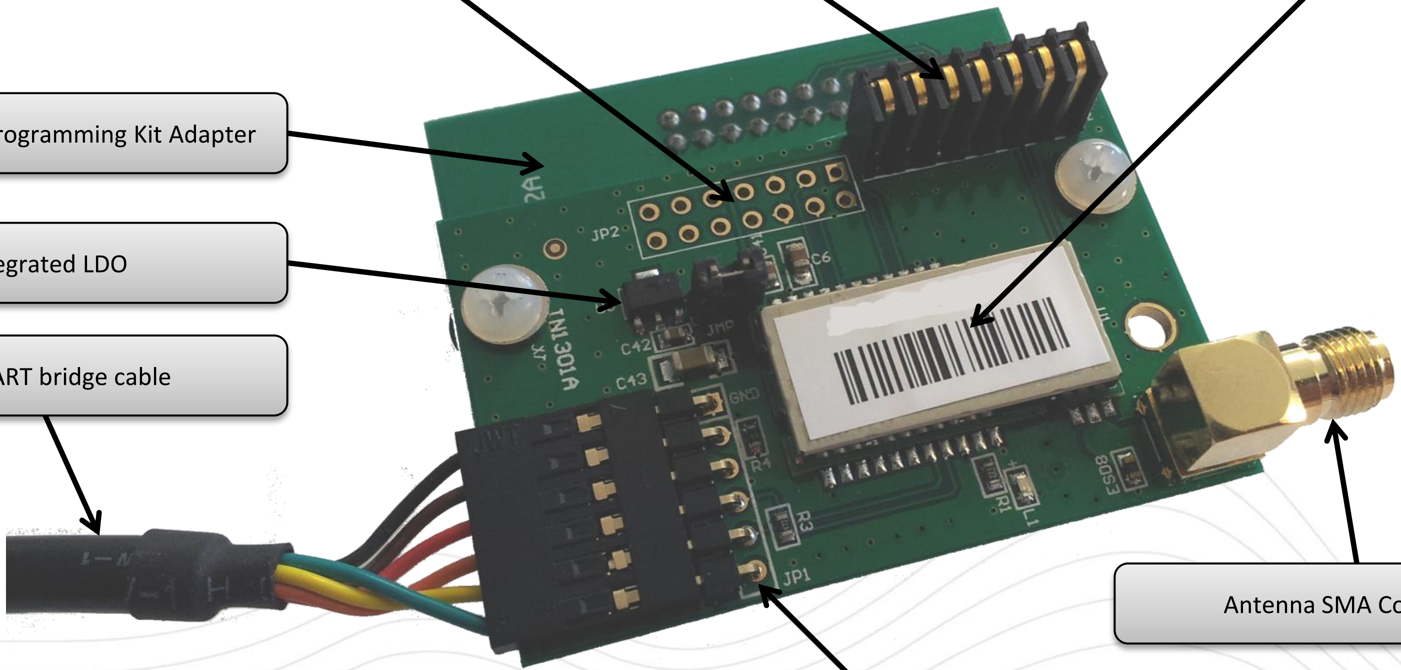
IDE

eclipse

Library

TD12xx RF Module SDK

PROGRAMMING KIT- HARDWARE



2-wire Debug – MCU Connector

Module I/Os Connector

TD1208 – Sigfox RF Module

ARM JTAG to Programming Kit Adapter

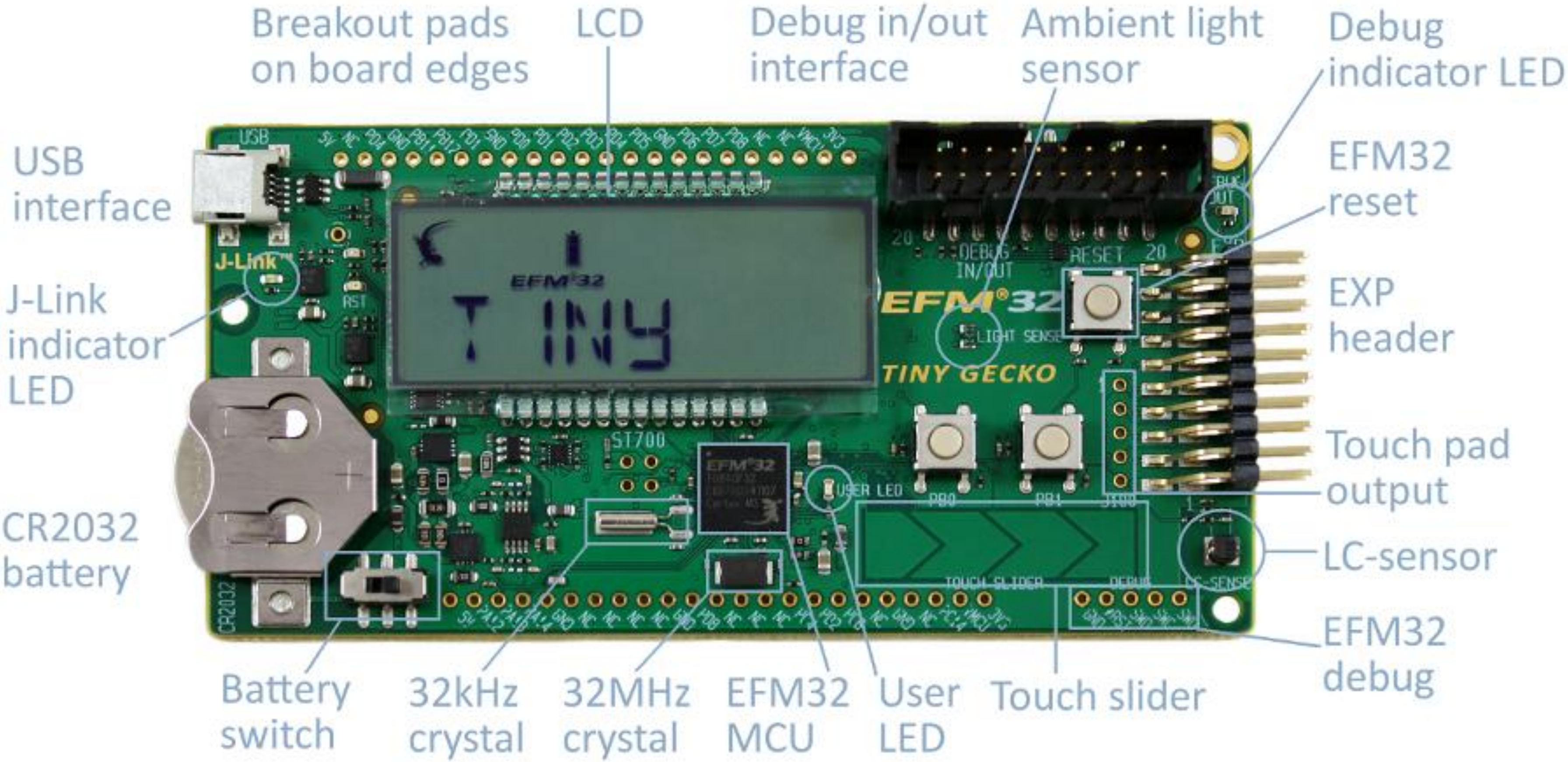
Integrated LDO

USB/UART bridge cable

Antenna SMA Connector

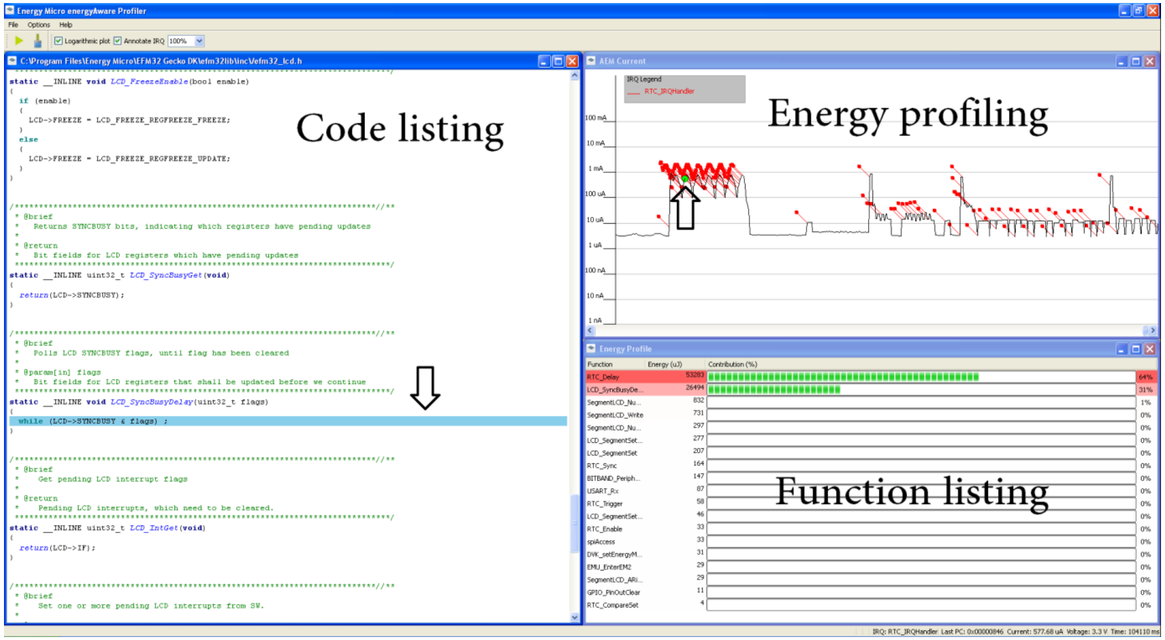
UART – Command Connector

ENERGY MICRO – EFM32TG-STK3300



ENERGY MICRO – ENERGY MEASUREMENT TOOLS

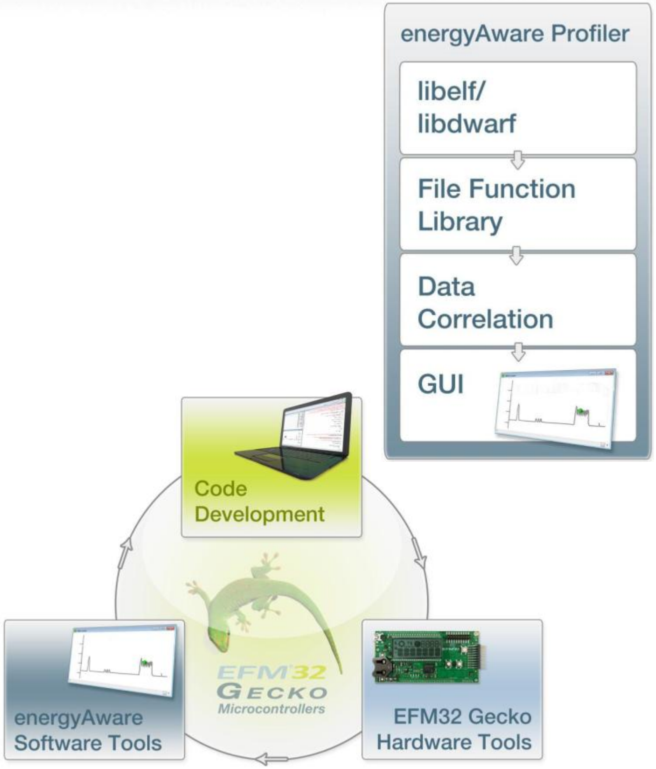
- EnergyAware Profiler



```
static __INLINE void LCD_FreezeEnable(bool enable)
{
  if (enable)
  {
    LCD->FREEZE = LCD_FREEZE_REGFREEZE_FREEZE;
  }
  else
  {
    LCD->FREEZE = LCD_FREEZE_REGFREEZE_UPDATE;
  }
}


/***************************************************************************//**
 * @brief
 *   Returns SYNCBUSY bits, indicating which registers have pending updates
 *
 * @return
 *   Bit fields for LCD registers which have pending updates
 ******************************************************************************/
static __INLINE uint32_t LCD_SyncBusyGet(void)
{
  return(LCD->SYNCBUSY);
}


/***************************************************************************//**
 * @brief
 *   Polls LCD SYNCBUSY flags, until flag has been cleared
 *
 * @param[in] flags
 *   Bit fields for LCD registers that shall be updated before we continue
 ******************************************************************************/
static __INLINE void LCD_SyncBusyDelay(uint32_t flags)
{
  while (LCD->SYNCBUSY & flags) ;
}


/***************************************************************************//**
 * @brief
 *    Get pending LCD interrupt flags
 *
 * @return
 *    Pending LCD interrupts, which need to be cleared.
 ******************************************************************************/
static __INLINE uint32_t LCD_IntGet(void)
{
  return(LCD->IF);
}


/***************************************************************************//**
 * @brief
 *    Set one or more pending LCD interrupts from SW.
 *
```

Function	Energy (uJ)	Contribution (%)
RTC_Delay	53283	64%
LCD_SyncBusyDe...	26494	31%
SegmentLCD_Nu...	832	1%
SegmentLCD_Write	731	0%
SegmentLCD_Nu...	297	0%
LCD_SegmentSet...	277	0%
LCD_SegmentSet	207	0%
RTC_Sync	164	0%
BITBAND_Periph...	147	0%
USART_Rx	87	0%
RTC_Trigger	58	0%
LCD_SegmentSet...	46	0%
RTC_Enable	33	0%
spiAccess	33	0%
DVK_setEnergyM...	31	0%
EMU_EnterEM2	29	0%
SegmentLCD_ARi...	29	0%
GPIO_PinOutClear	11	0%
RTC_CompareSet	4	0%

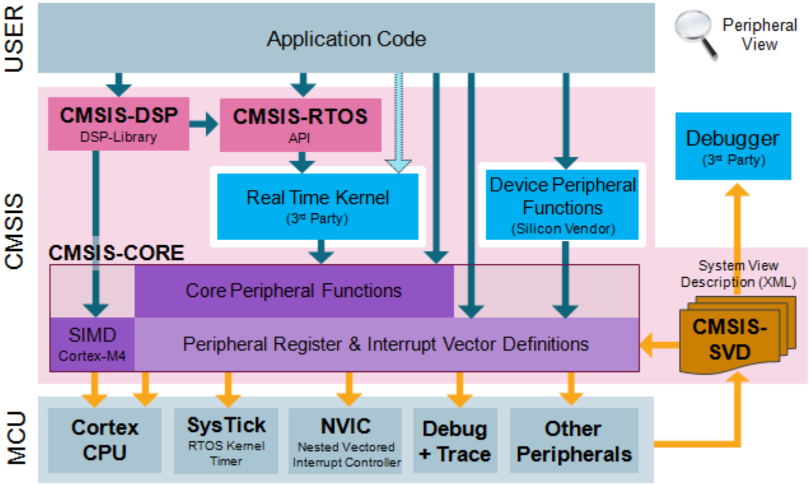
IRQ: RTC_IRQHandler Last PC: 0x00000846 Current: 577.68 uA Voltage: 3.3 V Time: 104110 ms



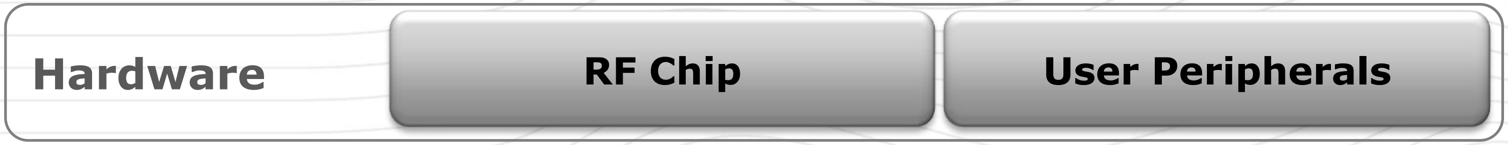
- AEM - Advanced Energy Monitoring
  - Hardware for Instantaneous current measurement
  - Available on all Energy Micro kits
- See WP0002 – EFM32 Energy Debugging

RF MODULE SDK SOFTWARE STACK

- ARM CMSIS (Cortex Microcontroller Software Interface Standard)
  - Core API
  - DSP library
  - RTOS API
  - SVD (System View Description)
  - Device Peripheral Functions (EM)
- Energy Micro Emlib
  - Low Level Drivers for MCU Peripherals
- TD Core library
  - Resource Sharing for TD Application Layer
- TD RF library
  - Sigfox Stack
  - RF Stack and RF Configuration
  - Low Energy LAN Stack
- TD Sensor library
  - Sensor UDM (Unified Device Model) Object Stack



RF MODULE HW/SW ARCHITECTURE

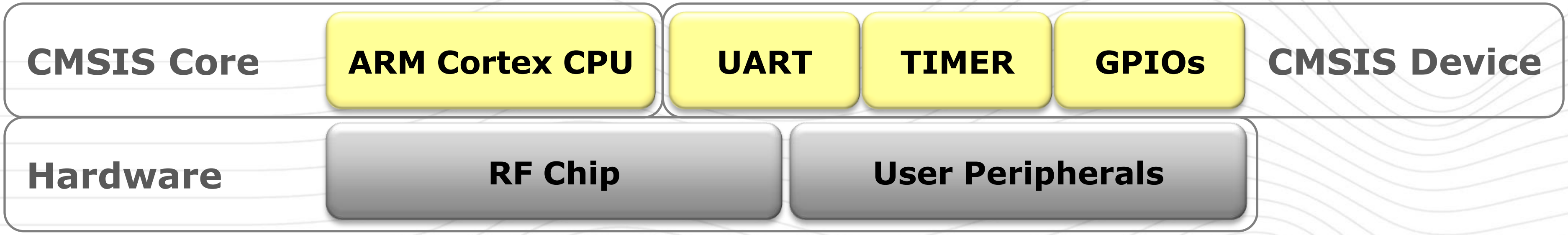


Hardware

RF Chip

User Peripherals

RF MODULE HW/SW ARCHITECTURE

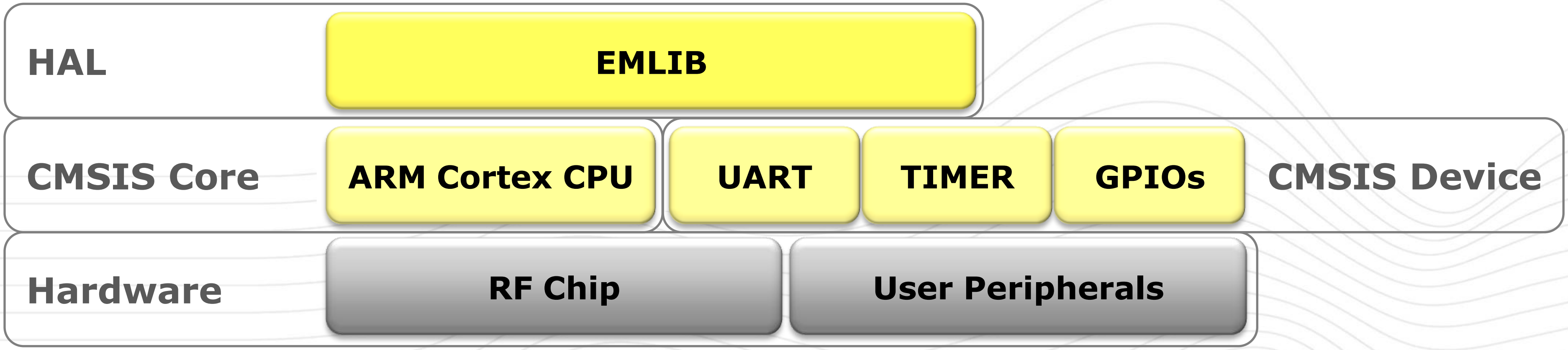


CMSIS Core: ARM Cortex CPU

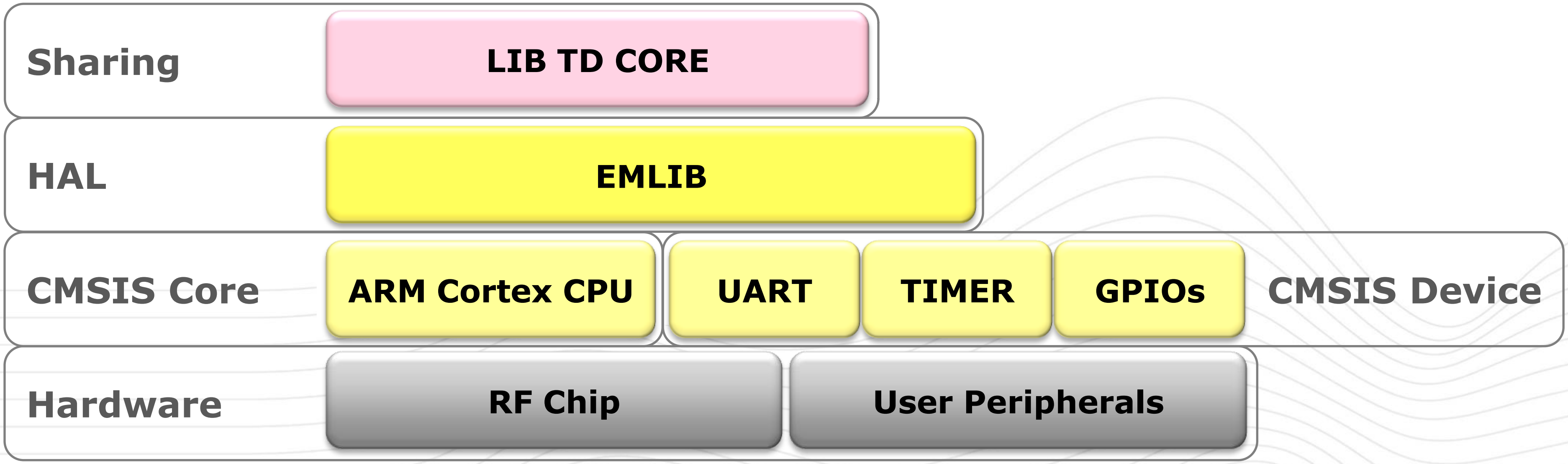
CMSIS Device: UART, TIMER, GPIOs

Hardware: RF Chip, User Peripherals

RF MODULE HW/SW ARCHITECTURE



RF MODULE HW/SW ARCHITECTURE



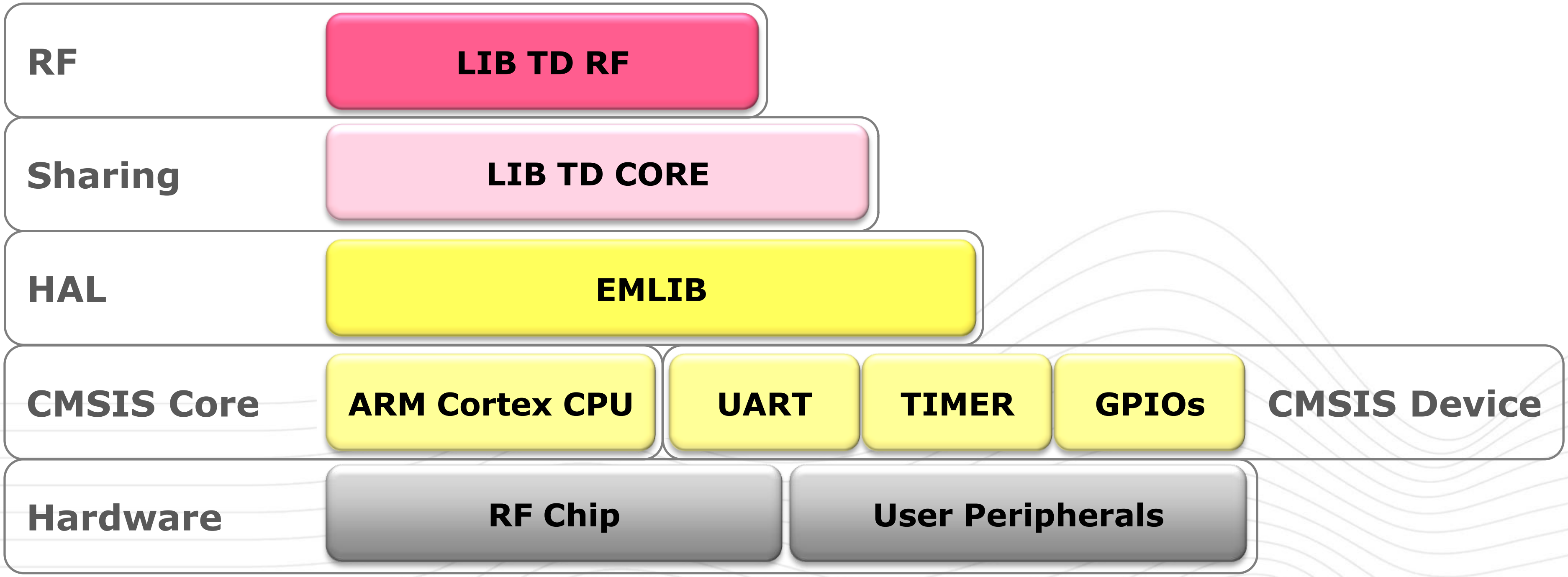
Sharing
LIB TD CORE

HAL
EMLIB

CMSIS Core
ARM Cortex CPU
UART
TIMER
GPIOs
CMSIS Device

Hardware
RF Chip
User Peripherals

RF MODULE HW/SW ARCHITECTURE



RF: LIB TD RF

Sharing: LIB TD CORE

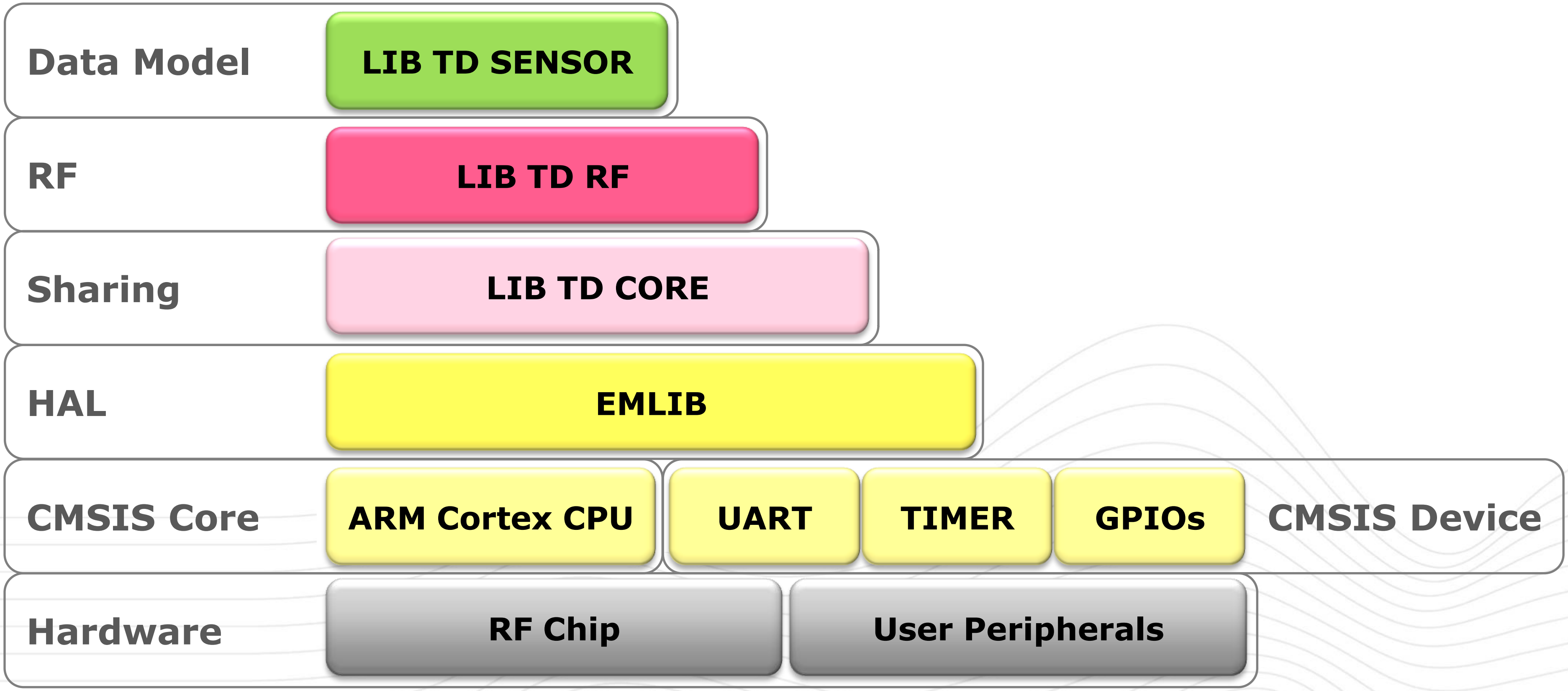
HAL: EMLIB

CMSIS Core: ARM Cortex CPU

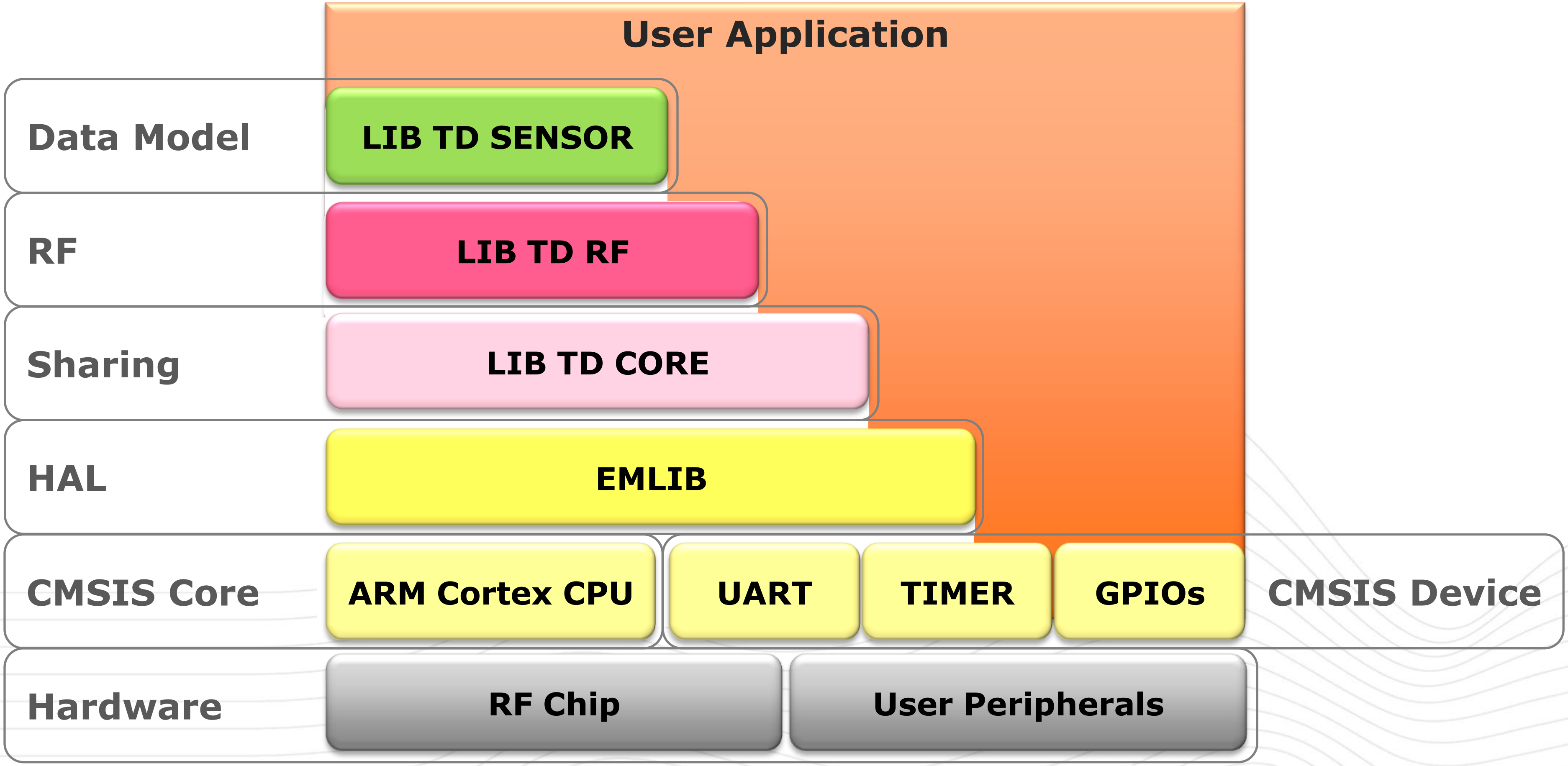
UART | TIMER | GPIOs: CMSIS Device

Hardware: RF Chip | User Peripherals

RF MODULE HW/SW ARCHITECTURE



RF MODULE HW/SW ARCHITECTURE



User Application

Data Model — LIB TD SENSOR

RF — LIB TD RF

Sharing — LIB TD CORE

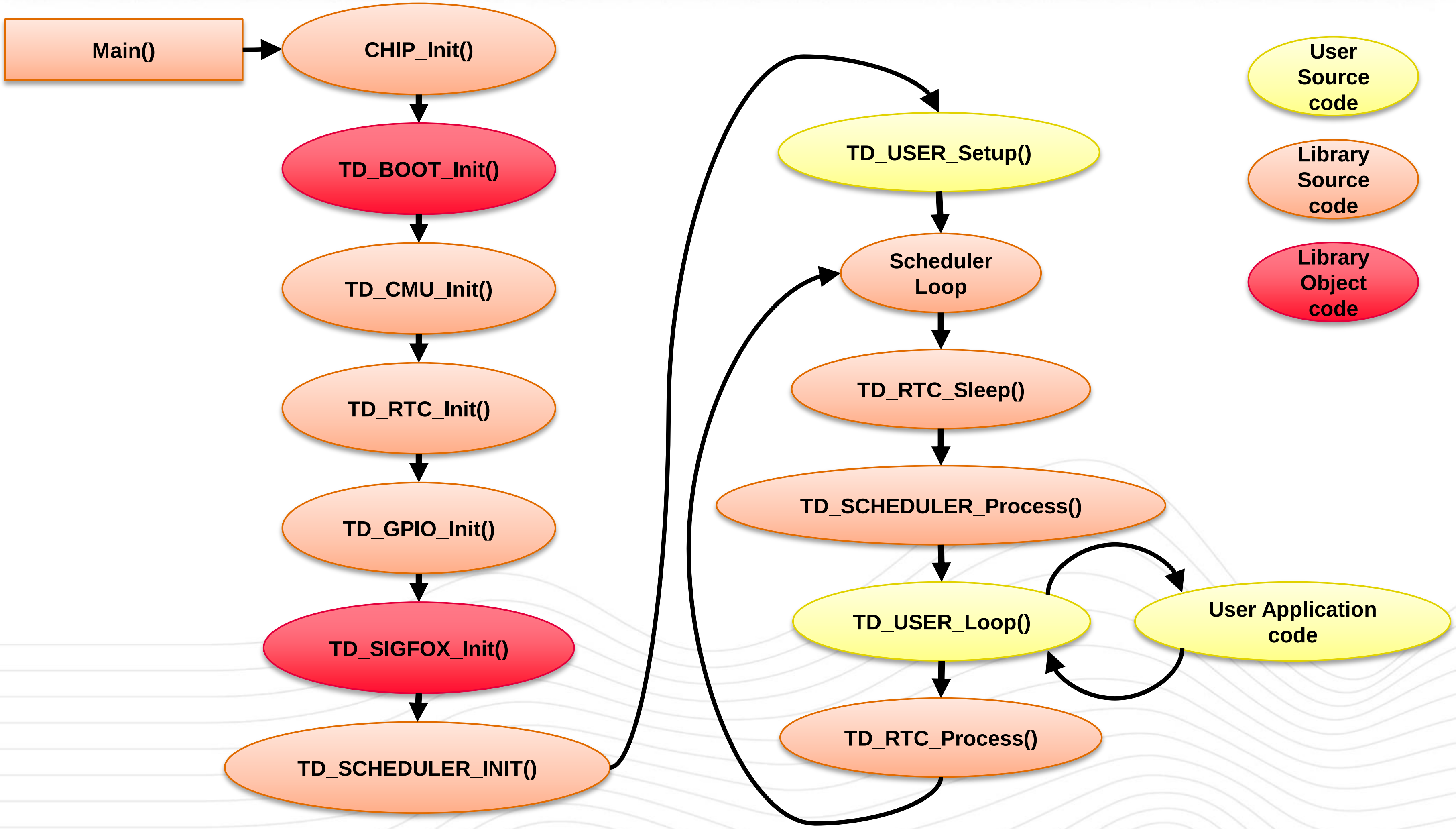
HAL — EMLIB

CMSIS Core — ARM Cortex CPU

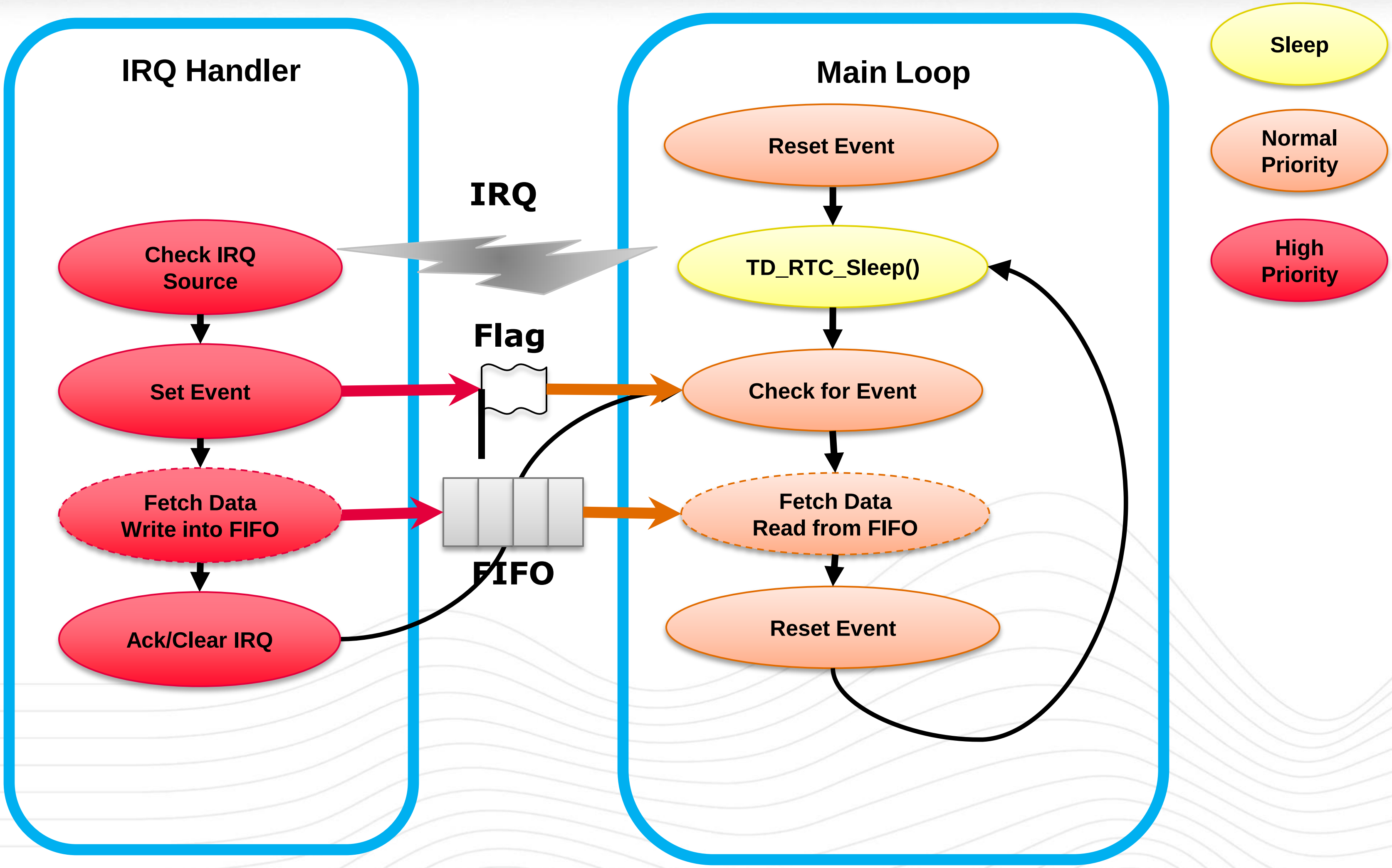
UART — TIMER — GPIOs — CMSIS Device

Hardware — RF Chip — User Peripherals

SOFTWARE ARCHITECTURE

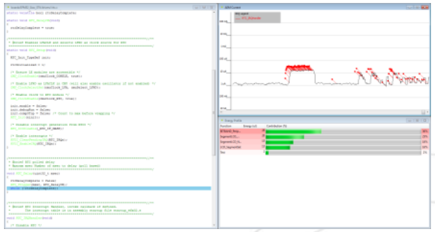


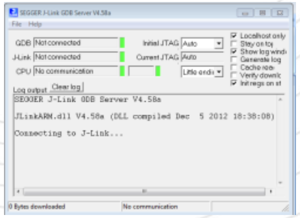
INTERRUPTS/EVENTS



TELECOM DESIGN
DEVELOPMENT TOOLS

- Pre-configured package in the TD RF Module SDK 3.0.0
  - Includes :
    - Eclipse IDE (version Juno SR2)
    - ARM GNU GCC Compiler (V4.7.2)
    - Git plugin for Eclipse
    - Energy Micro libraries emlib and CMSIS
    - Energy Micro software tools
    - J-Link from Segger drivers
    - Putty Terminal
    - FTDI VCP drivers
    - TD120x documentation



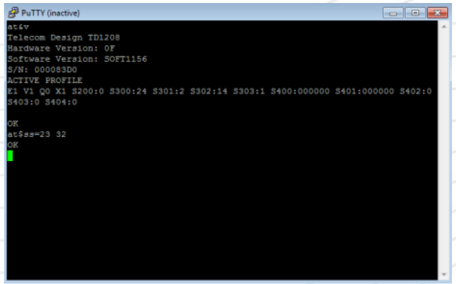


GDB: Not connected
J-Link: Not connected
CPU: No communication
Initial JTAG: Auto
Current JTAG: Auto
Little endi
☑ Localhost only
☐ Stay on to
☑ Show log wind
☐ Generate log
☐ Cache rea
☐ Verify downl
☑ Init regs on st

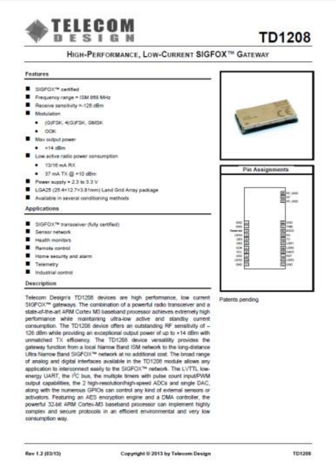
Log output

```
SEGGER J-Link GDB Server V4.58a

JLinkARM.dll V4.58a (DLL compiled Dec  5 2012 18:38:08)

Connecting to J-Link...
```

0 Bytes downloaded | No communication



```
at&v
Telecom Design TD1208
Hardware Version: 0F
Software Version: SOFT1156
S/N: 000083D0
ACTIVE PROFILE
E1 V1 Q0 X1 S200:0 S300:24 S301:2 S302:14 S303:1 S400:000000 S401:000000 S402:0
S403:0 S404:0

OK
at$ss=23 32
OK
```



TELECOM DESIGN

TD1208

HIGH-PERFORMANCE, LOW-CURRENT SIGFOX™ GATEWAY

Features

- SIGFOX™ certified
- Frequency range = ISM 868 MHz
- Receive sensitivity =-126 dBm
- Modulation
  - (G)FSK, 4(G)FSK, GMSK
  - OOK
- Max output power
  - +14 dBm
- Low active radio power consumption
  - 13/16 mA RX
  - 37 mA TX @ +10 dBm
- Power supply = 2.3 to 3.3 V
- LGA25 (25.4×12.7×3.81mm) Land Grid Array package
- Available in several conditioning methods

Applications

- SIGFOX™ transceiver (fully certified)
- Sensor network
- Health monitors
- Remote control
- Home security and alarm
- Telemetry
- Industrial control

Description

Telecom Design's TD1208 devices are high performance, low current SIGFOX™ gateways. The combination of a powerful radio transceiver and a state-of-the-art ARM Cortex M3 baseband processor achieves extremely high performance while maintaining ultra-low active and standby current consumption. The TD1208 device offers an outstanding RF sensitivity of –126 dBm while providing an exceptional output power of up to +14 dBm with unmatched TX efficiency. The TD1208 device versatility provides the gateway function from a local Narrow Band ISM network to the long-distance Ultra Narrow Band SIGFOX™ network at no additional cost. The broad range of analog and digital interfaces available in the TD1208 module allows any application to interconnect easily to the SIGFOX™ network. The LVTTL low-energy UART, the I²C bus, the multiple timers with pulse count input/PWM output capabilities, the 2 high-resolution/high-speed ADCs and single DAC, along with the numerous GPIOs can control any kind of external sensors or activators. Featuring an AES encryption engine and a DMA controller, the powerful 32-bit ARM Cortex-M3 baseband processor can implement highly complex and secure protocols in an efficient environmental and very low consumption way.

Pin Assignments

Patents pending

Rev 1.2 (03/13) Copyright © 2013 by Telecom Design TD1208

ECLIPSE IDE – GETTING STARTED

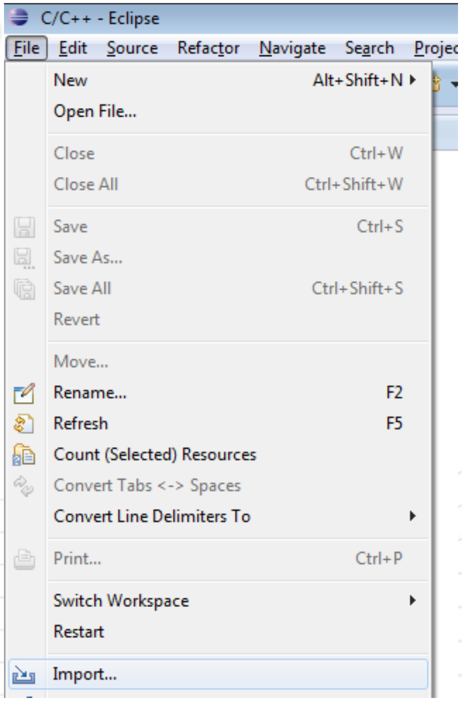
- Open Eclipse : C:\TD\TD_RF_Module_SDK-v3.0.0\eclipse\ eclipse.exe
  - Shortcut available on your desktop

- Projects location: C:\TD\TD_RF_Module_SDK-v3.0.0\Github\TD_RF_Module_SDK

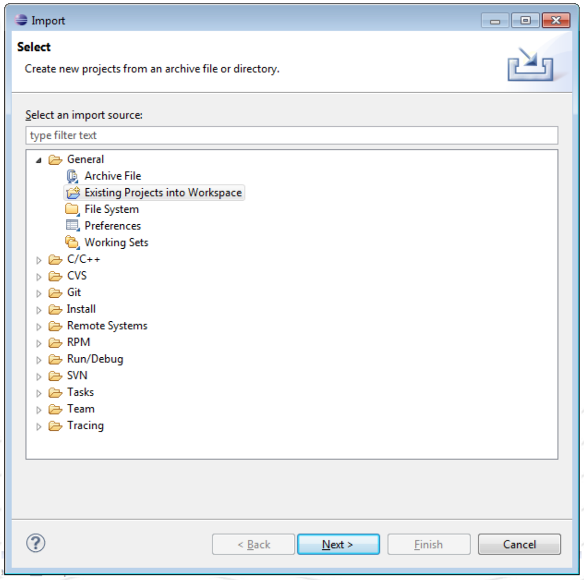
ECLIPSE – PROJECT IMPORT FROM FILE SYSTEM

- Step 1: Imports projects From File System (directory)
  - File ➔ Import… ➔ Existing Projects into Workspace ➔
    - Select root directory = C:\TD\TD_RF_Module_SDK-v3.0.0\Github\TD_RF_Module_SDK
    - **Uncheck** Copy projects into workspace,click Select All

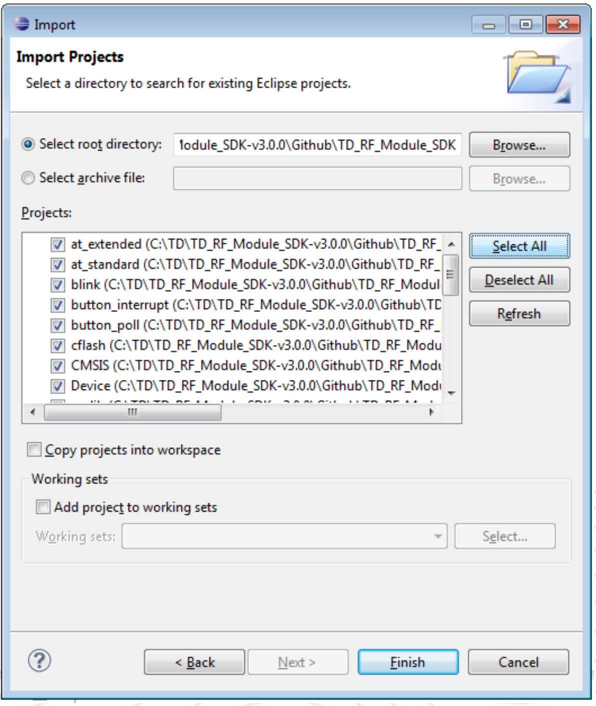
1



2



3



● Select root directory: 1odule_SDK-v3.0.0\Github\TD_RF_Module_SDK

○ Select archive file:

Projects:

☑ at_extended (C:\TD\TD_RF_Module_SDK-v3.0.0\Github\TD_RF_
☑ at_standard (C:\TD\TD_RF_Module_SDK-v3.0.0\Github\TD_RF_
☑ blink (C:\TD\TD_RF_Module_SDK-v3.0.0\Github\TD_RF_Modul
☑ button_interrupt (C:\TD\TD_RF_Module_SDK-v3.0.0\Github\TD
☑ button_poll (C:\TD\TD_RF_Module_SDK-v3.0.0\Github\TD_RF_
☑ cflash (C:\TD\TD_RF_Module_SDK-v3.0.0\Github\TD_RF_Modu
☑ CMSIS (C:\TD\TD_RF_Module_SDK-v3.0.0\Github\TD_RF_Modu
☑ Device (C:\TD\TD_RF_Module_SDK-v3.0.0\Github\TD_RF_Mod

☐ Copy projects into workspace

☐ Add project to working sets

Working sets:

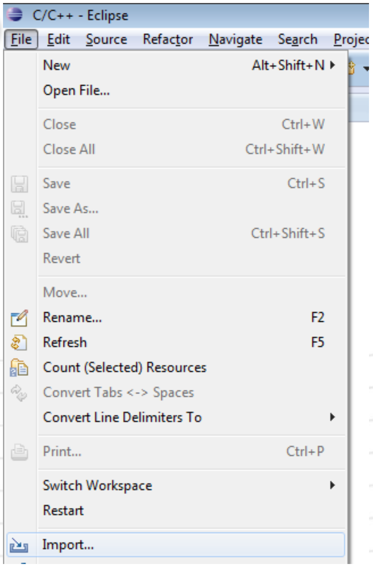
ECLIPSE – PROJECT IMPORT FROM GITHUB

- Step 1: Imports projects from Github Telecom Design

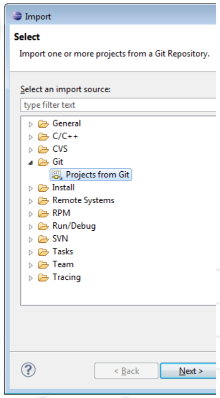
Note: private repository (need a Github account validation from TD)

- File ➔ Import… ➔ Git/Projects from Git ➔ URI
    - URI = https://github.com/Telecom-Design/TD_RF_Module_SDK.git
    - Fill your GitHub User and password
- Select Master ➔ Local destination
    - Directory = C:\TD\TD_RF_Module_SDK-v3.0.0\Github\TD_RF_Module_SDK

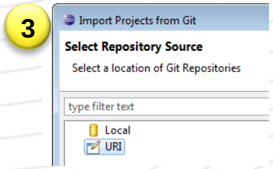
1



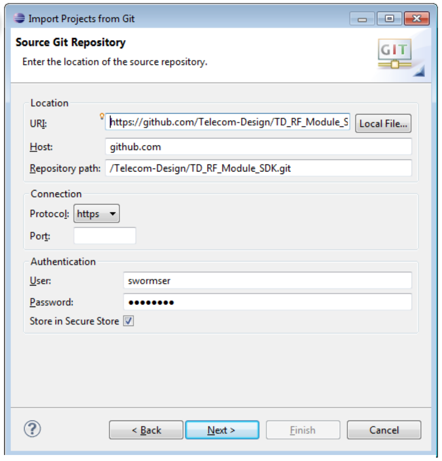
2



3



4



URI: https://github.com/Telecom-Design/TD_RF_Module_S

Host: github.com

Repository path: /Telecom-Design/TD_RF_Module_SDK.git

Protocol: https

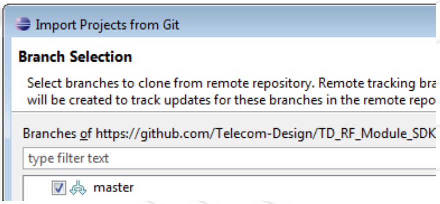
Port:

User: swormser

Password: ••••••••

Store in Secure Store ☑

5

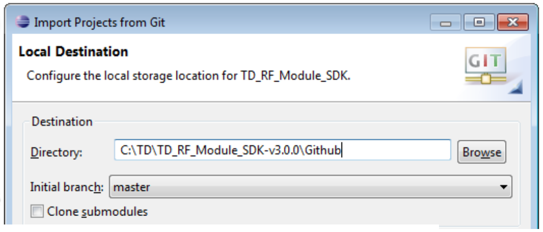


Select branches to clone from remote repository. Remote tracking bra
will be created to track updates for these branches in the remote repo

Branches of https://github.com/Telecom-Design/TD_RF_Module_SDK

☑ master

6

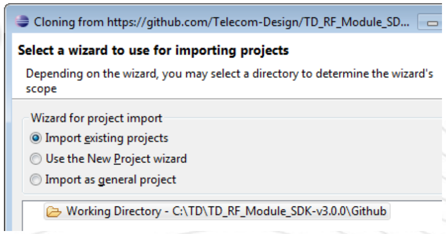


Directory: C:\TD\TD_RF_Module_SDK-v3.0.0\Github

Initial branch: master

□ Clone submodules

7


Cloning from https://github.com/Telecom-Design/TD_RF_Module_SD...
Select a wizard to use for importing projects
Depending on the wizard, you may select a directory to determine the wizard's scope
Wizard for project import

● Import existing projects

○ Use the New Project wizard

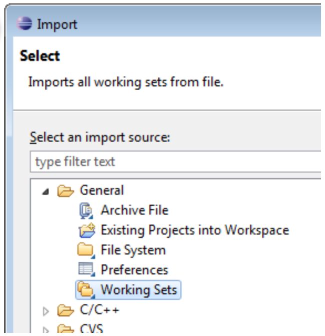
○ Import as general project

Working Directory - C:\TD\TD_RF_Module_SDK-v3.0.0\Github

ECLIPSE - ADDING A WORKING SET TO WORKSPACE

- Step 2: imports a working set to workspace
  - File ➔ Imports … ➔ General/Working Sets
    - Uncheck TD1202 (we are using TD1208 only for this lab)

1


Import

Select

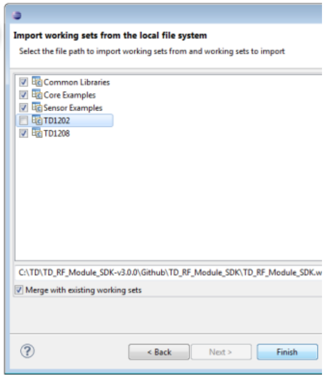
Imports all working sets from file.

Select an import source:

type filter text

- General
  - Archive File
  - Existing Projects into Workspace
  - File System
  - Preferences
  - Working Sets
- C/C++
- CVS

2


Import working sets from the local file system

Select the file path to import working sets from and working sets to import

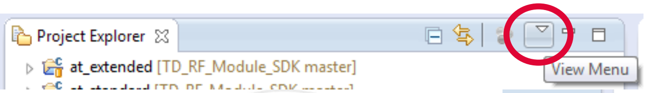
☑ Common Libraries
☑ Core Examples
☑ Sensor Examples
☐ TD1202
☑ TD1208

C:\TD\TD_RF_Module_SDK-v3.0.0\Github\TD_RF_Module_SDK\TD_RF_Module_SDK.w

☑ Merge with existing working sets

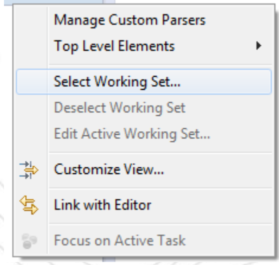
- Step 3: Set the Working Set to the current Workspace
  - In Project Explorer ➔ view menu icon ➔ Select Working Set… ➔ Select All but TD1202

1

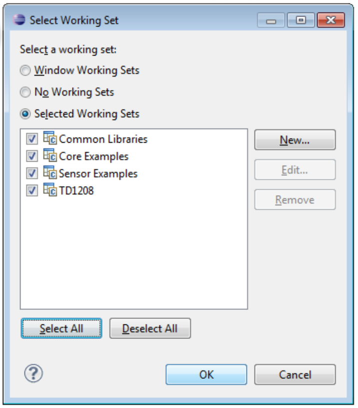

Project Explorer

at_extended [TD_RF_Module_SDK master]

2



3


Select Working Set

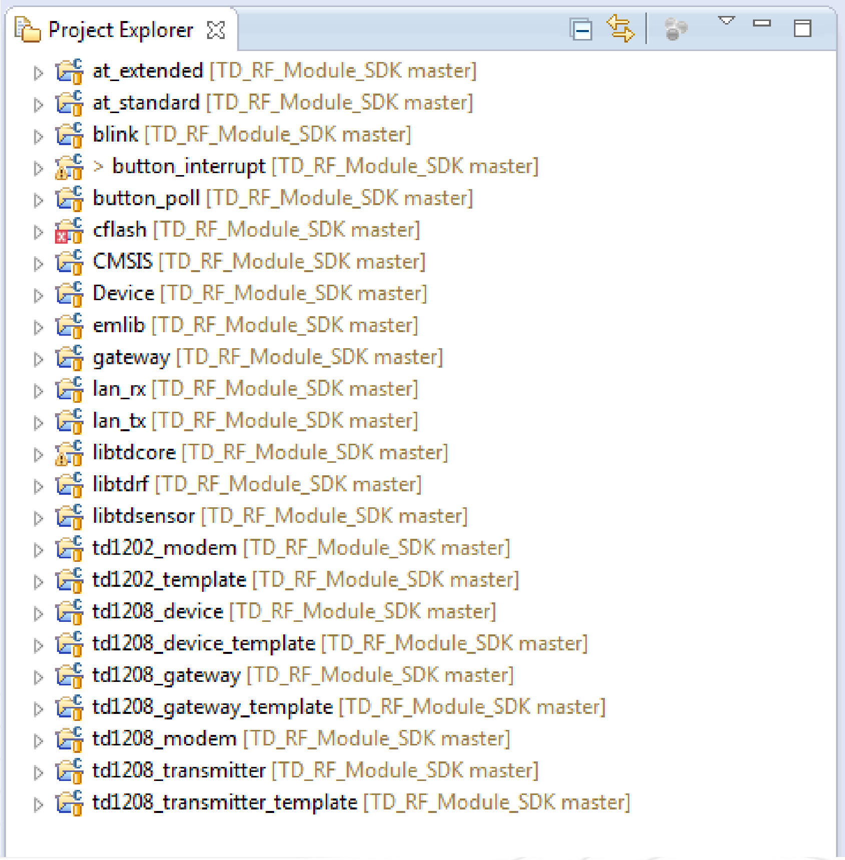
Select a working set:

○ Window Working Sets
○ No Working Sets
● Selected Working Sets

☑ Common Libraries
☑ Core Examples
☑ Sensor Examples
☑ TD1208

ECLIPSE – VIEW BEFORE/AFTER WORKING SET

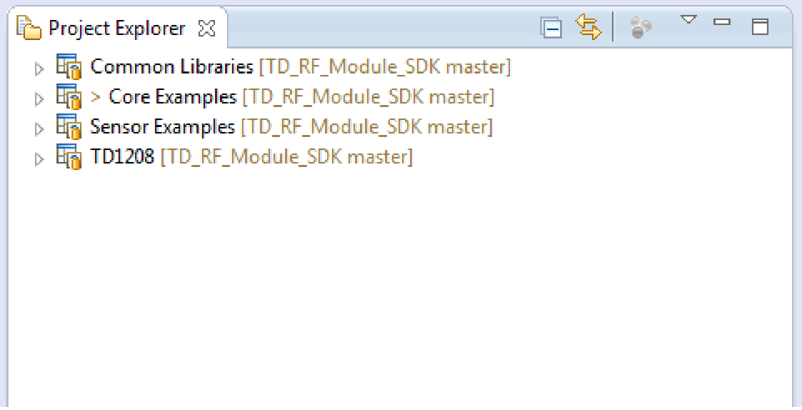
Before



Project Explorer

- at_extended [TD_RF_Module_SDK master]
- at_standard [TD_RF_Module_SDK master]
- blink [TD_RF_Module_SDK master]
- > button_interrupt [TD_RF_Module_SDK master]
- button_poll [TD_RF_Module_SDK master]
- cflash [TD_RF_Module_SDK master]
- CMSIS [TD_RF_Module_SDK master]
- Device [TD_RF_Module_SDK master]
- emlib [TD_RF_Module_SDK master]
- gateway [TD_RF_Module_SDK master]
- lan_rx [TD_RF_Module_SDK master]
- lan_tx [TD_RF_Module_SDK master]
- libtdcore [TD_RF_Module_SDK master]
- libtdrf [TD_RF_Module_SDK master]
- libtdsensor [TD_RF_Module_SDK master]
- td1202_modem [TD_RF_Module_SDK master]
- td1202_template [TD_RF_Module_SDK master]
- td1208_device [TD_RF_Module_SDK master]
- td1208_device_template [TD_RF_Module_SDK master]
- td1208_gateway [TD_RF_Module_SDK master]
- td1208_gateway_template [TD_RF_Module_SDK master]
- td1208_modem [TD_RF_Module_SDK master]
- td1208_transmitter [TD_RF_Module_SDK master]
- td1208_transmitter_template [TD_RF_Module_SDK master]

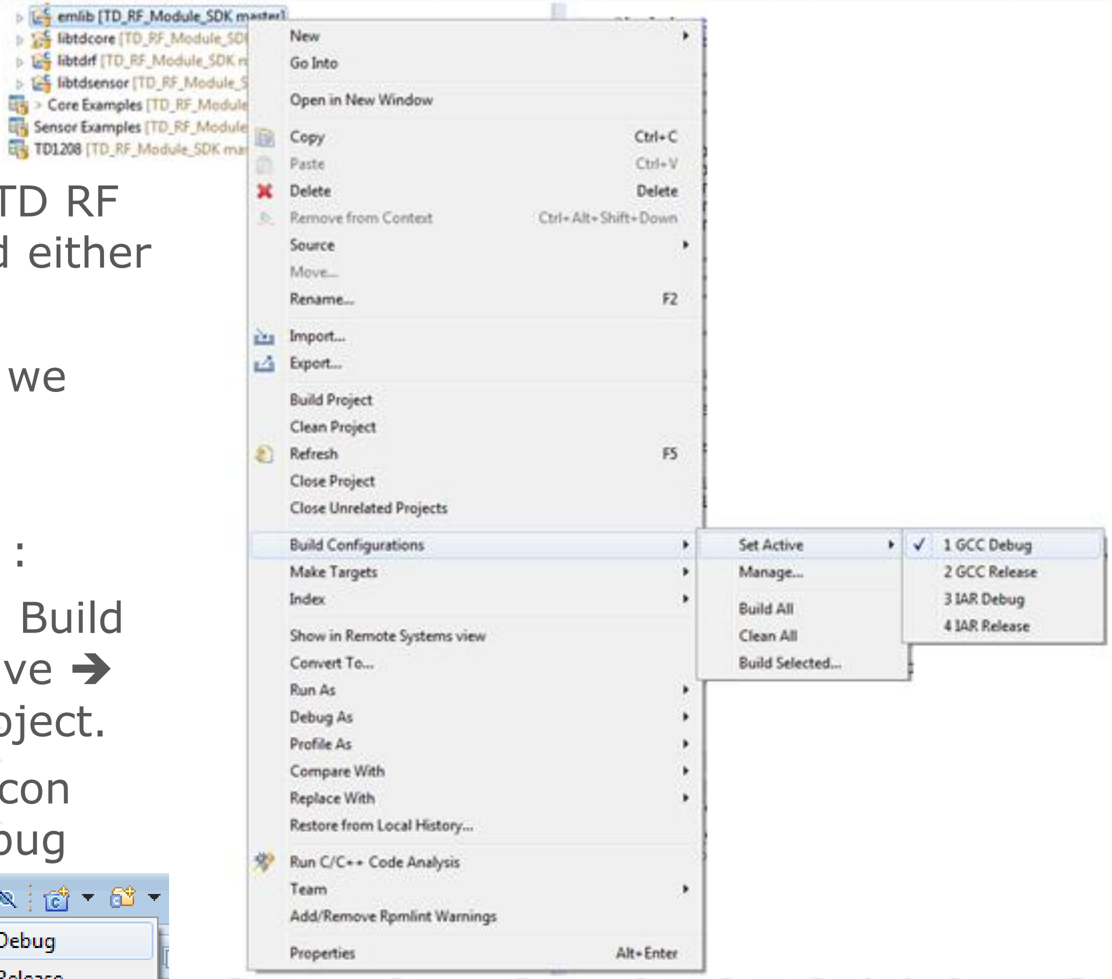
After



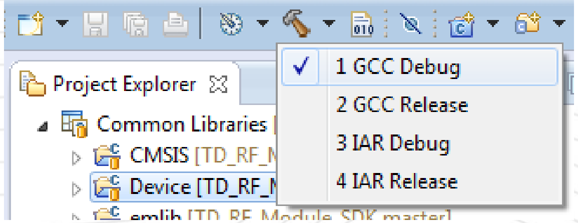
Project Explorer

- Common Libraries [TD_RF_Module_SDK master]
- > Core Examples [TD_RF_Module_SDK master]
- Sensor Examples [TD_RF_Module_SDK master]
- TD1208 [TD_RF_Module_SDK master]

ECLIPSE – BUILD PROJECT



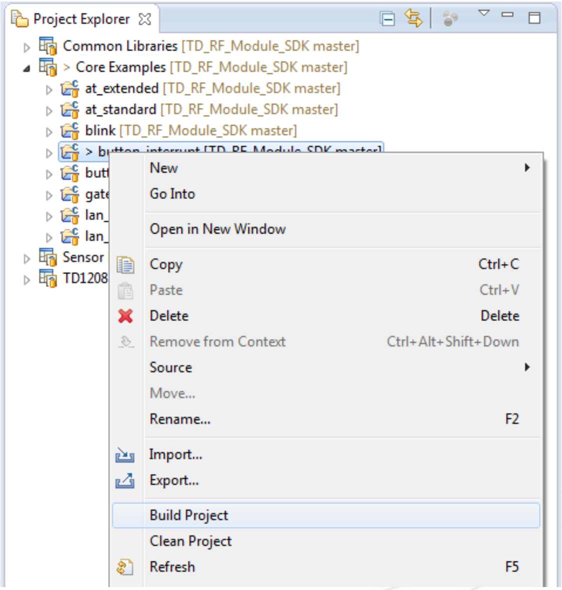
- Imported projects from the TD RF Module SDK can be compiled either for GCC or IAR compilers.
- We use GCC for the labs, so we configure all projects for it.
- Step 4 : Set or verify build configuration for all projects :
  - Right click on a Project ➔ Build Configurations ➔ Set Active ➔ GCC Debug then Build Project.
  - Or Click on the Hammer icon arrow and select GCC Debug



ECLIPSE – BUILD, FLASH AND DEBUG

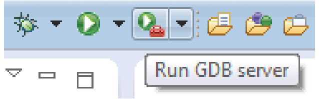
- Step 5: Build the project
  - Select the project to build in the Project manager ➔ Right click ➔ Build Project

1



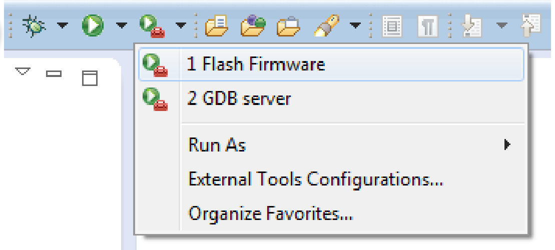
Project Explorer
Common Libraries [TD_RF_Module_SDK master]
> Core Examples [TD_RF_Module_SDK master]
at_extended [TD_RF_Module_SDK master]
at_standard [TD_RF_Module_SDK master]
blink [TD_RF_Module_SDK master]
> button_interrupt [TD_RF_Module_SDK master]
butt
gate
lan_
lan_
Sensor
TD1208
New
Go Into
Open in New Window
Copy Ctrl+C
Paste Ctrl+V
Delete Delete
Remove from Context Ctrl+Alt+Shift+Down
Source
Move...
Rename... F2
Import...
Export...
Build Project
Clean Project
Refresh F5

- Step 6: Flash the firmware or Debug



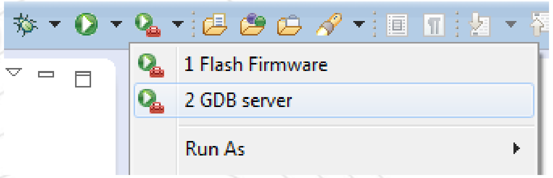
- To Flash :
  - Flash Firmware

2a

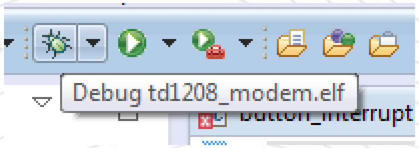


- To Debug:
  - GDB server
  - Debug

2b



3b



HANDS-ON : BUTTON INTERRUPT

- Used functions:
  - CMSIS
    - **NVIC_ClearPendingIRQ():** Clear IRQ in ARM Core
    - **NVIC_EnableIRQ():** Enable IRQ in ARM Core
  - Emlib
    - **GPIO_PinModeSet():** Configure GPIO Pin
    - **GPIO_IntConfig():** Configure Peripheral IRQ
    - **GPIO_PinOutSet():** Set GPIO Pin Level to 1
    - **GPIO_PinOutClear():** Set GPIO Pin Level to 0
    - **GPIO_PinInGet():** Get GPIO Pin Level
  - Libtdcore
    - **TD_USER_Setup() :** User hardware module configuration. This function is called once at initialization
    - **TD_USER_Loop():** User loop function
    - **TD_GPIO_SetCallback() :** Register a callback function for processing IRQ matching a given bit mask
  - Libtdrf
    - **TD_SIGFOX_Send() :** Send a SIGFOX frame

NVIC_CLEARPENDINGIRQ()

- Example:

```
#include <efm32.h>
/* Clear ADC0 interrupts. */
NVIC_ClearPendingIRQ(ADC0_IRQn);
```

- Synopsis:
  - The function clears the pending bit of an external interrupt.
- Input parameters:
  - **IRQn_Type IRQn**: External interrupt number. Value cannot be negative.

IRQn_Type	
DMA_IRQn	EFM32 DMA Interrupt
GPIO_EVEN_IRQn	EFM32 GPIO_EVEN Interrupt
TIMER0_IRQn	EFM32 TIMER0 Interrupt
USART0_RX_IRQn	EFM32 USART0_RX Interrupt
USART0_TX_IRQn	EFM32 USART0_TX Interrupt
ACMP0_IRQn	EFM32 ACMP0 Interrupt
ADC0_IRQn	EFM32 ADC0 Interrupt
DAC0_IRQn	EFM32 DAC0 Interrupt
I2C0_IRQn	EFM32 I2C0 Interrupt
GPIO_ODD_IRQn	EFM32 GPIO_ODD Interrupt
TIMER1_IRQn	EFM32 TIMER1 Interrupt
USART1_RX_IRQn	EFM32 USART1_RX Interrupt
USART1_TX_IRQn	EFM32 USART1_TX Interrupt
LEUART0_IRQn	EFM32 LEUART0 Interrupt
LETIMER0_IRQn	EFM32 LETIMER0 Interrupt
PCNT0_IRQn	EFM32 PCNT0 Interrupt
RTC_IRQn	EFM32 RTC Interrupt
CMU_IRQn	EFM32 CMU Interrupt
VCPM_IRQn	EFM32 VCMP Interrupt
MSC_IRQn	EFM32 MSC Interrupt
AES_IRQn	EFM32 AES Interrupt

NVIC_ENABLEIRQ()

- Example:

```
#include <efm32.h>
/* Enable ADC0 interrupts. */
NVIC_EnableIRQ(ADC0_IRQn);
```

- Synopsis:
  - The function enables a device-specific interrupt in the NVIC interrupt controller.
- Input parameters:
  - **IRQn_Type IRQn**: External interrupt number. Value cannot be negative.

IRQn_Type	
DMA_IRQn	EFM32 DMA Interrupt
GPIO_EVEN_IRQn	EFM32 GPIO_EVEN Interrupt
TIMER0_IRQn	EFM32 TIMER0 Interrupt
USART0_RX_IRQn	EFM32 USART0_RX Interrupt
USART0_TX_IRQn	EFM32 USART0_TX Interrupt
ACMP0_IRQn	EFM32 ACMP0 Interrupt
ADC0_IRQn	EFM32 ADC0 Interrupt
DAC0_IRQn	EFM32 DAC0 Interrupt
I2C0_IRQn	EFM32 I2C0 Interrupt
GPIO_ODD_IRQn	EFM32 GPIO_ODD Interrupt
TIMER1_IRQn	EFM32 TIMER1 Interrupt
USART1_RX_IRQn	EFM32 USART1_RX Interrupt
USART1_TX_IRQn	EFM32 USART1_TX Interrupt
LEUART0_IRQn	EFM32 LEUART0 Interrupt
LETIMER0_IRQn	EFM32 LETIMER0 Interrupt
PCNT0_IRQn	EFM32 PCNT0 Interrupt
RTC_IRQn	EFM32 RTC Interrupt
CMU_IRQn	EFM32 CMU Interrupt
VCPM_IRQn	EFM32 VCMP Interrupt
MSC_IRQn	EFM32 MSC Interrupt
AES_IRQn	EFM32 AES Interrupt

GPIO_PINMODESET()

- Example:

```
#include <efm32.h>
#include <em_gpio.h>
/* PortE1 set in push-pull
mode. */
GPIO_PinModeSet(gpioPortE, 1,
  gpioModePushPull, 0);
```

GPIO_Port_TypeDef	
gpioPortA	Port A
gpioPortB	Port B
gpioPortC	Port C
gpioPortD	Port D
gpioPortE	Port E
gpioPortF	Port F

GPIO_Mode_TypeDef	
gpioModeDisabled	Input disabled. Pullup if DOUT is set.
gpioModeInput	Input enabled. Filter if DOUT is set
gpioModeInputPull	Input enabled. DOUT determines pull direction
gpioModeInputPullFilter	Input enabled with filter. DOUT determines pull direction
gpioModePushPull	Push-pull output
gpioModePushPullDrive	Push-pull output with drive-strength set by DRIVEMODE
gpioModeWiredOr	Wired-or output
gpioModeWiredOrPullDown	Wired-or output with pull-down
gpioModeWiredAnd	Open-drain output
gpioModeWiredAndFilter	Open-drain output with filter
gpioModeWiredAndPullUp	Open-drain output with pullup
gpioModeWiredAndPullUpFilter	Open-drain output with filter and pullup
gpioModeWiredAndDrive	Open-drain output with drive-strength set by DRIVEMODE
gpioModeWiredAndDriveFilter	Open-drain output with filter and drive-strength set by DRIVEMODE
gpioModeWiredAndDrivePullUp	Open-drain output with pullup and drive-strength set by DRIVEMODE
gpioModeWiredAndDrivePullUpFilter	Open-drain output with filter, pullup and drive-strength set by DRIVEMODE

- Synopsis:
  - Set the mode for a GPIO pin.
- Input parameters :
  - **GPIO_Port_TypeDef port**: The GPIO port to access.
  - **unsigned int pin** : The pin number in the port.
  - **GPIO_Mode_TypeDef mode**: The desired pin mode.
  - **unsigned int out**: Value to set for pin in DOUT register. The DOUT setting is important for even some input mode configurations, determining pull-up/down direction.

GPIO_INTCONFIG()

- Example:

```
#include <efm32.h>

#include <em_gpio.h>

/* PortE1 IRQ enabled on rising edge. */

GPIO_IntConfig(gpioPortE, 1, true, false, true);
```

GPIO_Port_TypeDef	
gpioPortA	Port A
gpioPortB	Port B
gpioPortC	Port C
gpioPortD	Port D
gpioPortE	Port E
gpioPortF	Port F

- Synopsis:
  - Configure GPIO interrupt.
- Input parameters:
  - **GPIO_Port_TypeDef port**: The port to associate with pin.
  - **unsigned int pin** : The GPIO interrupt number.
  - **bool risingEdge**: Set to true if interrupts shall be enabled on rising edge, otherwise false.
  - **bool fallingEdge**: Set to true if interrupts shall be enabled on falling edge , otherwise false.
  - **bool enable**: Set to true if interrupts shall be enabled after configuration completed, false to leave disabled.

GPIO_PINOUTSET()

- Example:

```
#include <efm32.h>

#include <em_gpio.h>

/* Set PortE1 output level to 1. */

GPIO_PinOutSet(gpioPortE, 1);
```

- Synopsis:
  - Set a single pin in GPIO data out register to 1.
    In order for the setting to take effect on the output pad, the pin must have been configured properly. If not, it will take effect whenever the pin has been properly configured.
- Input parameters:
  - **GPIO_Port_TypeDef port**: The GPIO port to access.
  - **unsigned int pin** : The pin to set.

GPIO_Port_TypeDef	
gpioPortA	Port A
gpioPortB	Port B
gpioPortC	Port C
gpioPortD	Port D
gpioPortE	Port E
gpioPortF	Port F

GPIO_PINOUTCLEAR()

- Example:

```
#include <efm32.h>
#include <em_gpio.h>
/* Set PortE1 output level to 0. */
GPIO_PinOutClear(gpioPortE, 1);
```

GPIO_Port_TypeDef	
gpioPortA	Port A
gpioPortB	Port B
gpioPortC	Port C
gpioPortD	Port D
gpioPortE	Port E
gpioPortF	Port F

- Synopsis:
  - Set a single pin in GPIO data out register to 0.
    In order for the setting to take effect on the output pad, the pin must have been configured properly. If not, it will take effect whenever the pin has been properly configured.
- Input parameters:
  - **GPIO_Port_TypeDef port**: The GPIO port to access.
  - **unsigned int pin** : The pin to set.

GPIO_PININGET()

- Example:

```
#include <efm32.h>
#include <em_gpio.h>
/* Read PortE1 input level. */
unsigned int value = GPIO_PinInGet(gpioPortE, 1);
```

GPIO_Port_TypeDef	
gpioPortA	Port A
gpioPortB	Port B
gpioPortC	Port C
gpioPortD	Port D
gpioPortE	Port E
gpioPortF	Port F

- Synopsis:
  - Read the pad value for a single pin in GPIO port.
- Input parameters:
  - **GPIO_Port_TypeDef port**: The GPIO port to access.
  - **unsigned int pin** : The pin to read.
- Result value:
  - The read pad value.

TD_USER_SETUP()

- Example:

```
#include <td_core.h>
/* Perform initial user setup. */
TD_USER_Setup();
```

- Synopsis:
  - User hardware module configuration. This function is called once at initialization.

TD_USER_LOOP()

- Example:

```
#include <td_core.h>
/* Perform user actions when awaken. */
TD_USER_Loop();
```

- Synopsis:
  - User function called when the module is awaken to perform actions.

TD_GPIO_SETCALLBACK()

- Example:

```
#include <td_gpio.h>

/* Setup a callback function upon odd user-level interrupt
  on pin 3. */

static void my_handler(void)

{

}

TD_GPIO_SetCallback(TD_GPIO_USER_ODD, my_handler, (1 <<
  3));
```

- Synopsis:
  - Register a callback function for processing IRQ matching a given bit mask .
- Input parameters:
  - **int type**: The type of callback to set up.
  - **TD_GPIO_callback_t callback**: Pointer to the callback function called when an interrupt matching the mask is received.
  - **uint32_t mask**: Mask for testing a received interrupt

type	
TD_GPIO_SYSTEM_ODD	Odd system IRQ
TD_GPIO_SYSTEM_EVEN	Even system IRQ
TD_GPIO_USER_ODD	Odd user IRQ
TD_GPIO_USER_EVEN	Even User IRQ

TD_SIGFOX_SEND()

- Example:

```
#include <td_sigfox.h>

/* Send a 12-byte character buffer to the SIGFOX network, using 2 repeats. */

bool result = TD_SIGFOX_Send(buffer, 12, 2);
```

- Synopsis:
  - Send a SIGFOX frame.
- Input parameters:
  - **uint8_t mess**: Pointer to the input frame buffer.
  - **uint8_t size:** The size of the input frame buffer from 1 to 12 bytes.
  - **uint8_t retry**: The number of retries that must be performed when sending the frame. This value is kept for compatibility but is ignored, the number of retries is fixed to 2.
- Result value:
  - Returns true if the operation was successful, false otherwise.

EXERCISE : BUTTON INTERRUPT

- **Goal: Send a button-press counter to the SIGFOX network every time a button is pressed**
  - The module must be sleeping when not active to save power
  - As there is no physical button on the board, the button press is simulated by sending a « BREAK » signal on the RX pin using a terminal emulator software
- **You must:**
  - Initialize the LED GPIO on the TIM2 pin to « push-pull » output mode, with a default 0 (off) value
  - Initialize the pseudo-button GPIO on RX pin to input with « pull-up » mode, with a default 1 value
  - Setup a user-supplied function to handle interrupts generated on the pseudo-button pin falling edge transitions
  - When a corresponding interrupt is received:
    - Turn on the LED
    - Send a SIGFOX frame containing 2 fixed header bytes « 0x00 » and » 0x04 », followed by the single-byte unsigned counter, starting with a 1 value
    - Wait actively until the pseudo-button is released
    - Turn off the LED