Microcontroladores ARM Cortex-M3

- Core ARM Cortex-M3
  - Soportado por las herramientas de desarrollo
- Periféricos de ARM Cortex-M3
  - Soportados por las herramientas de desarrollo
- Periféricos desarrollados por el fabricante del microcontrolador
  - Respetan ciertas pautas de operación fijadas por ARM
    - Poseen archivos de definiciones que permiten que sean soportados por las herramientas de desarrollo

Core ARM Cortex-M3

- Arquitectura ARMv7-M
- Harvard con espacios unificados en mapa de 4GB
- Load/Store
- Bit band

Periféricos de ARM Cortex-M3

- NVIC
  - Nested Vectored Interrupt Controller
- SysTick
  - Timer continuo

Herramientas de desarrollo

- IDE
- Programador – Debugger
- CMSIS

IDEs

- *El entorno de desarrollo en un Cortex-M3 no es una elección del fabricante, sino del desarrollador*

- (Si bien existen algunos casos particulares que son todo lo contrario, siempre existe la posibilidad de recurrir a un proveedor no atado con el fabricante o incluso gratuito o hasta Open Source)

Keil

- "una empresa del grupo ARM"
  - algo así como “el oficial”
  - uVision 4
  - ARM-MDK (Microcontroller Development Kit)
  - 32KB de código generado.
- CMSIS
  - por defecto
  - los include files son CMSIS.

IAR

- Embedded Workbench for ARM (EWARM)
  - 32KB de código generado.
- CMSIS
  - soporte a partir de la versión 6.2
    - debe activarse manualmente en las opciones del proyecto
  - los include files son para su propio formato, diferente de CMSIS.
  - paquetes provistos por fabricantes

CooCox

- basado en Eclipse
  - compilador GNU (gcc) provisto por Code Sourcery (hoy Mentor Graphics)
- No posee simulador, ni forma de ver los periféricos al momento.
- CMSIS
  - debe activarse manualmente en las opciones del proyecto.
  - los include files son CMSIS (sin el prefijo <fabcode>_)

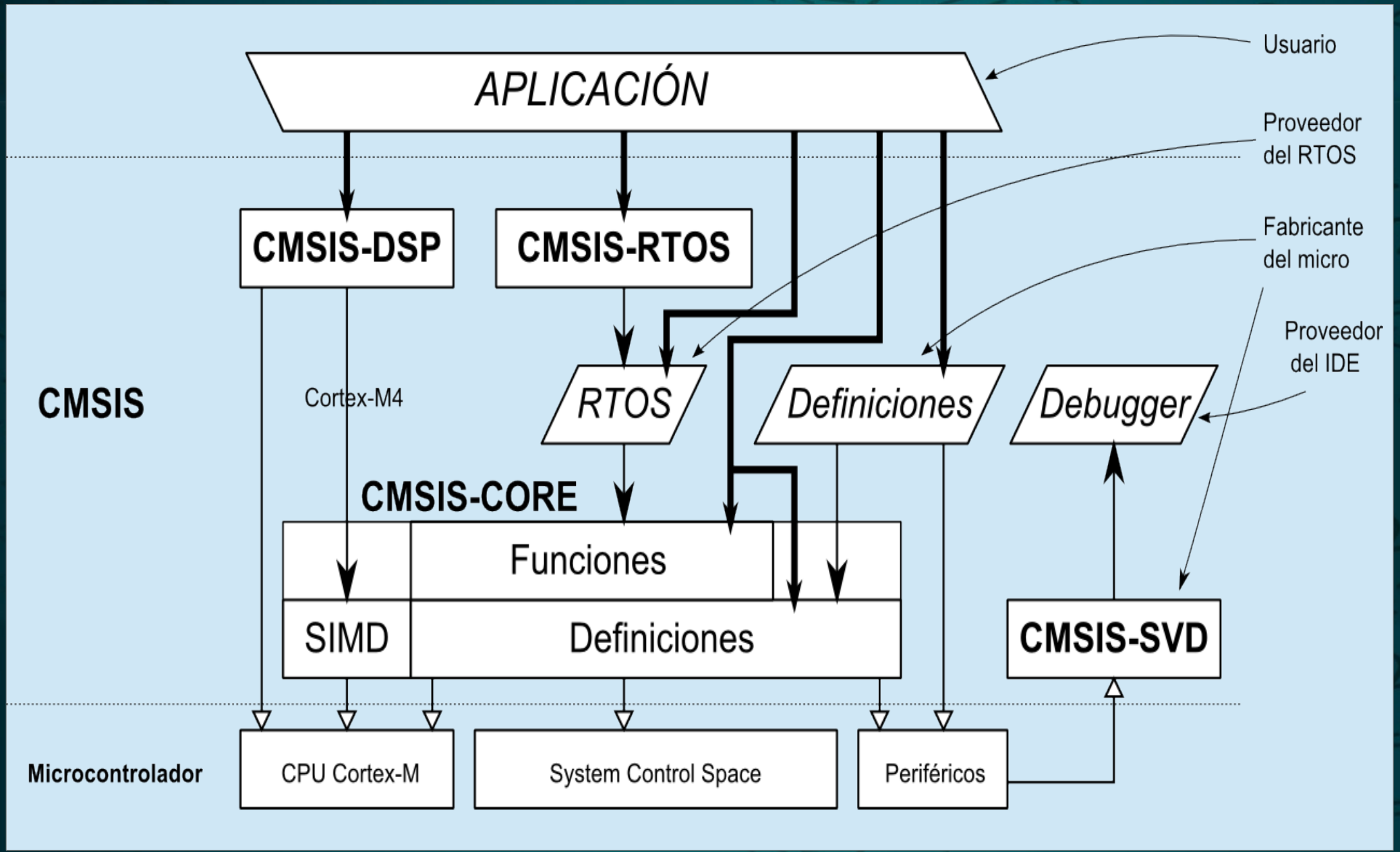
Programador-Debugger

- conector de .1” con 20 pines
  - compatible JTAG.
  - standard familia Cortex
  - 10 pines de paso .05”.
- Algunos micros poseen ambas interfaces: JTAG y SWD.

Programador-Debugger

- Segger J-Link
  - funciona con IAR, Keil, CooCox
- Keil uLink
  - funciona con IAR, Keil, CooCox
- otros fabricantes de JTAG
- embedded en los devkits *
- Alternativas Open Source
  - Colink-EX
    - funciona bajo CooCox
    - drivers para IAR y Keil

CMSIS


Usuario
APLICACIÓN
Proveedor del RTOS
Fabricante del micro
Proveedor del IDE
CMSIS-DSP
CMSIS-RTOS
CMSIS
Cortex-M4
RTOS
Definiciones
Debugger
CMSIS-CORE
Funciones
SIMD
Definiciones
CMSIS-SVD
Microcontrolador
CPU Cortex-M
System Control Space
Periféricos

Desarrollo de aplicaciones en C

- CMSIS
  - ARM
    - archivos de soporte para el core
  - fabricante del micro
    - headers de soporte para los periféricos
  - fabricante del compilador
    - inclusión de CMSIS (puede hacerse manual)
- Compilador
  - fabricante del compilador
    - soporte para ese micro (headers, prog de la flash)

CMSIS

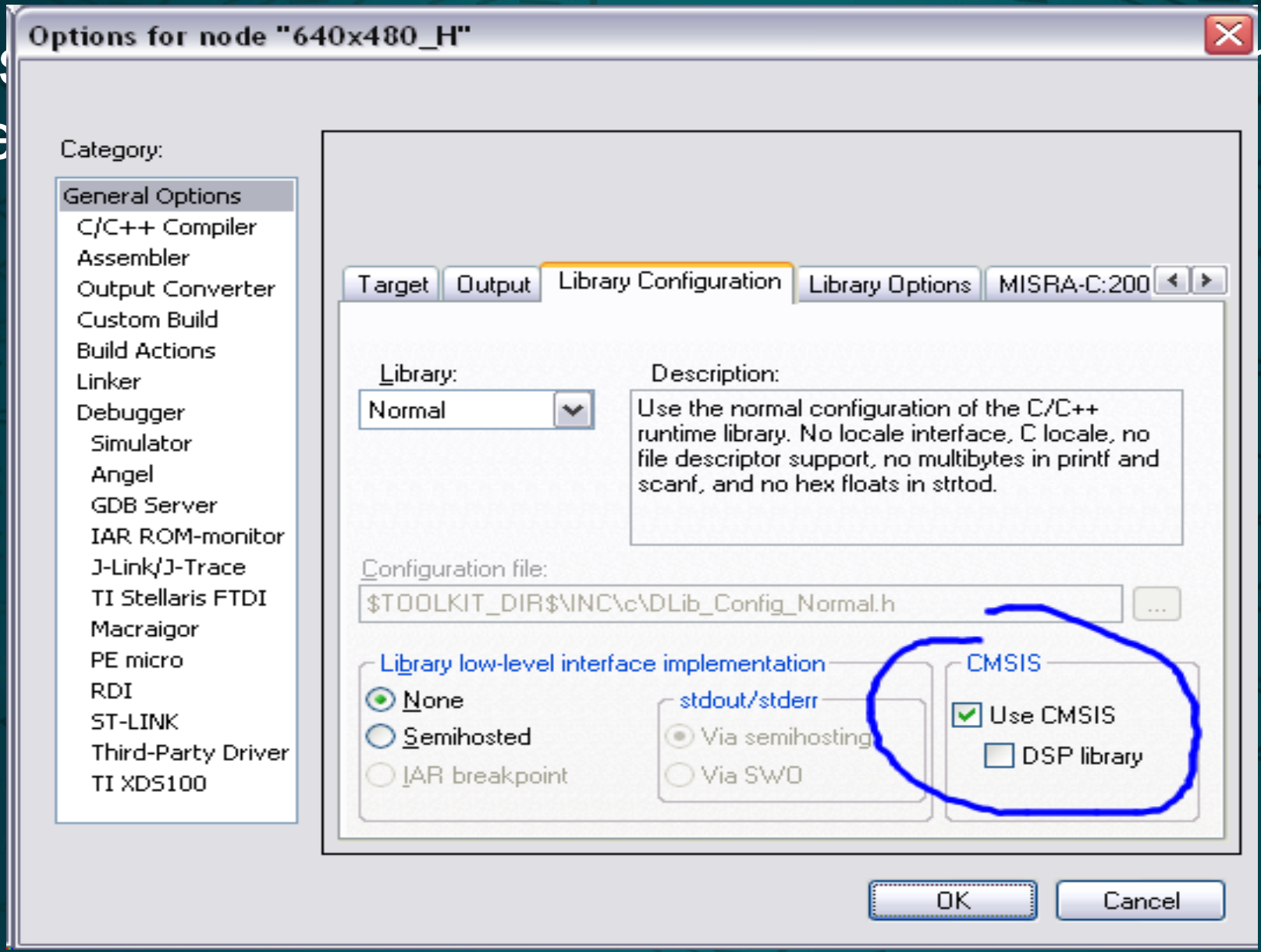
- Archivos generales para soportar core y compilador
  - *core_cm3.h*
- Archivos particulares para soportar el micro
  - *system_<micro>.c*
    - contiene *SystemInit()*, rutina que se encarga de inicializar el clock
  - *startup_<micro>.s* o *startup_<micro>.c*
    - contiene vectores de interrupción y reset. Este último llama a *SystemInit()* y luego salta a ejecutar el código del usuario en *main()*.

Keil

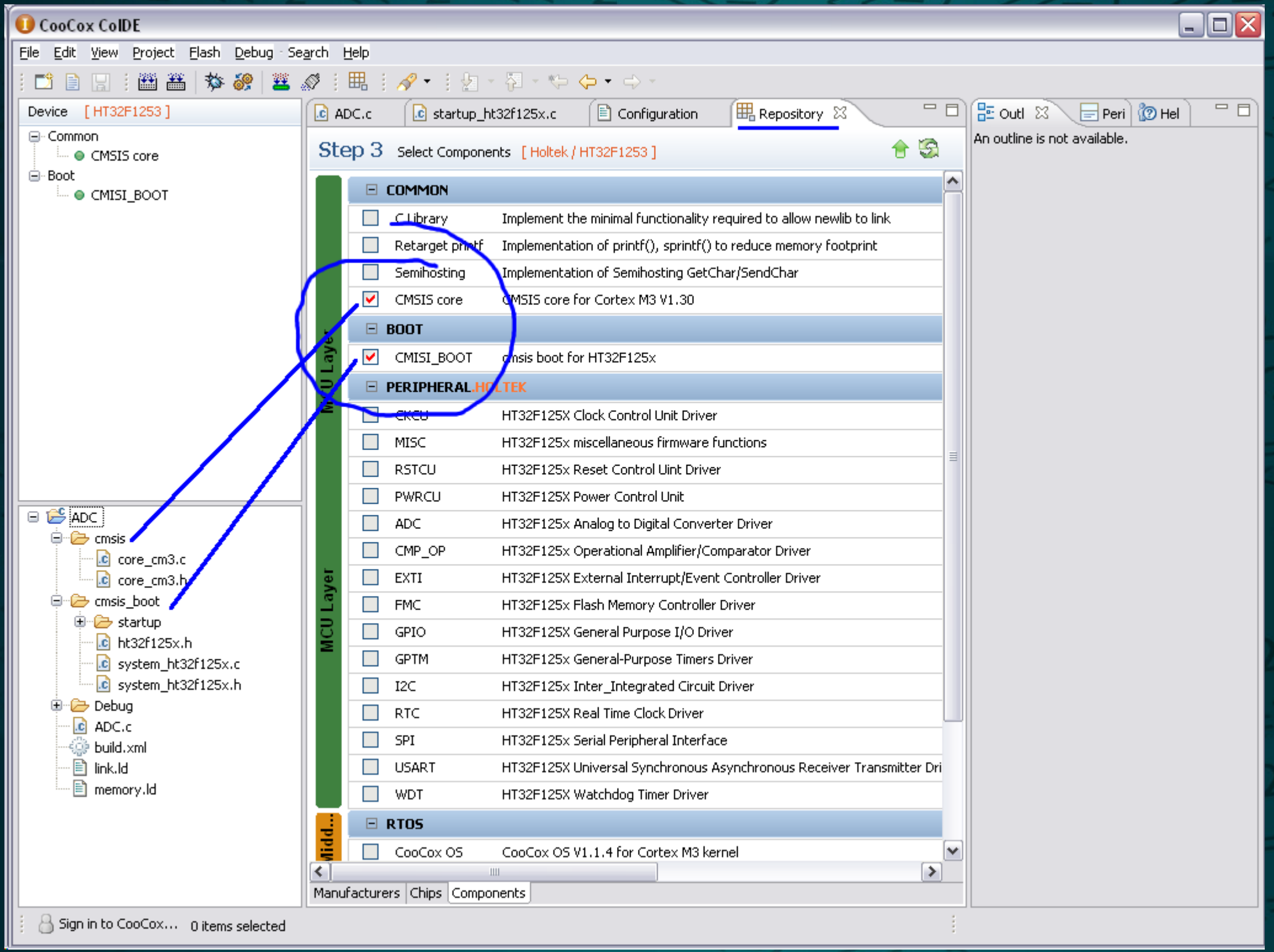
- Agregamos los archivos desde donde residen.
  - Podemos simplemente incluirlos o copiarlos a nuestro directorio de trabajo.
  - Los archivos requeridos se encuentran en la siguiente ubicación por defecto:
    - Generales
      - incluídos por el header del micro en particular: mimicro.h
    - Particulares
      - C:\Keil\ARM\Startup\<fabricante>\<micro>\

IAR

- Los
me


Options for node "640x480_H"
Category:
General Options
C/C++ Compiler
Assembler
Output Converter
Custom Build
Build Actions
Linker
Debugger
Simulator
Angel
GDB Server
IAR ROM-monitor
J-Link/J-Trace
TI Stellaris FTDI
Macraigor
PE micro
RDI
ST-LINK
Third-Party Driver
TI XDS100
Target
Output
Library Configuration
Library Options
MISRA-C:200
Library:
Normal
Description:
Use the normal configuration of the C/C++ runtime library. No locale interface, C locale, no file descriptor support, no multibytes in printf and scanf, and no hex floats in strtod.
Configuration file:
$TOOLKIT_DIR$\INC\c\DLib_Config_Normal.h
Library low-level interface implementation
None
Semihosted
IAR breakpoint
stdout/stderr
Via semihosting
Via SWO
CMSIS
Use CMSIS
DSP library
OK
Cancel

Los archivos particulares los debemos copiar a nuestro directorio de trabajo e incluirlos manualmente, por nuestra cuenta.

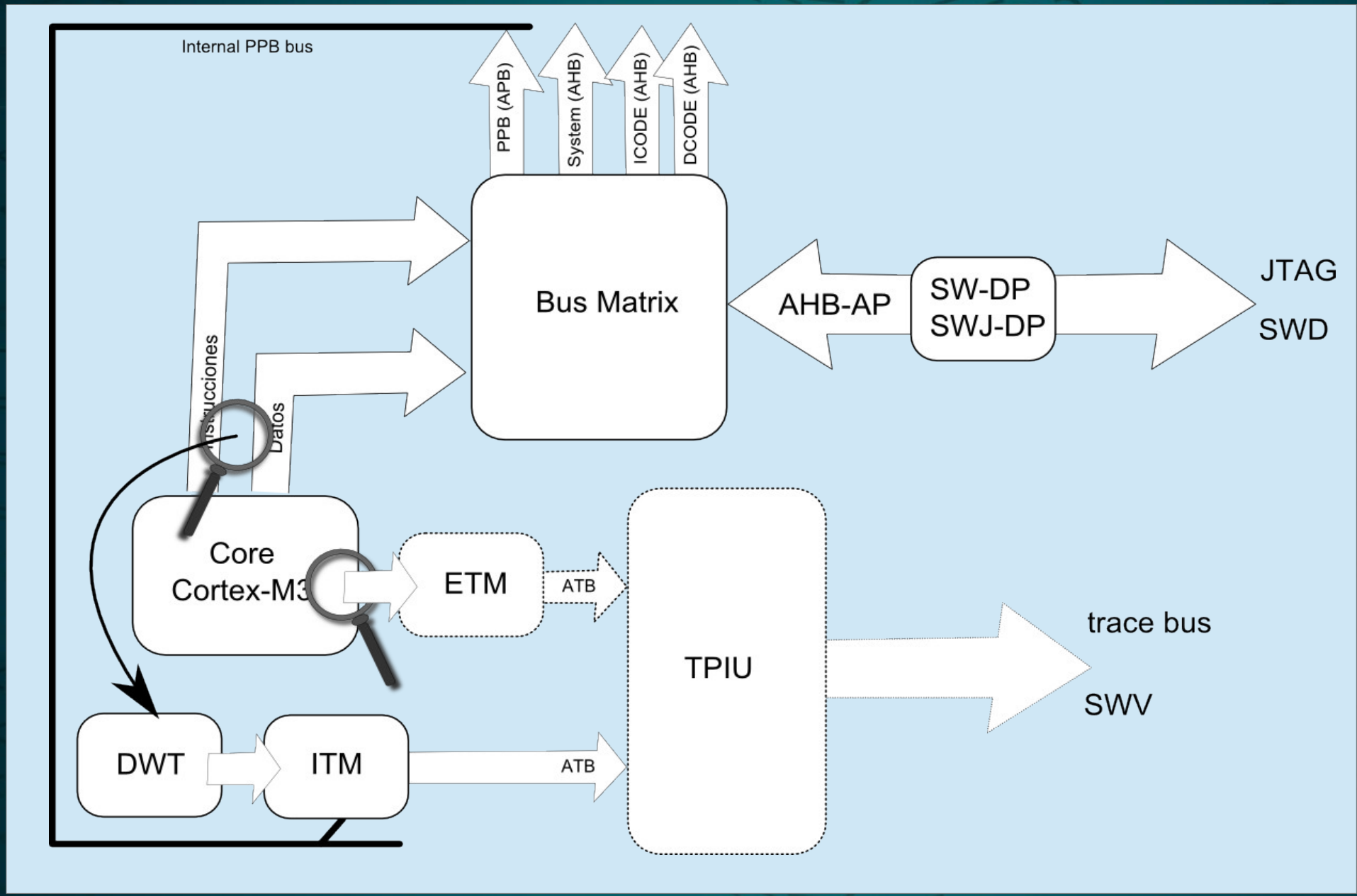

CooCox CoIDE
File Edit View Project Flash Debug Search Help
Device [HT32F1253]
Common
CMSIS core
Boot
CMISI_BOOT
ADC
cmsis
core_cm3.c
core_cm3.h
cmsis_boot
startup
ht32f125x.h
system_ht32f125x.c
system_ht32f125x.h
Debug
ADC.c
build.xml
link.ld
memory.ld
ADC.c
startup_ht32f125x.c
Configuration
Repository
Step 3 Select Components [Holtek / HT32F1253]
MCU Layer
COMMON
C Library Implement the minimal functionality required to allow newlib to link
Retarget printf Implementation of printf(), sprintf() to reduce memory footprint
Semihosting Implementation of Semihosting GetChar/SendChar
CMSIS core CMSIS core for Cortex M3 V1.30
BOOT
CMISI_BOOT cmsis boot for HT32F125x
PERIPHERAL.HOLTEK
CKCU HT32F125X Clock Control Unit Driver
MISC HT32F125x miscellaneous firmware functions
RSTCU HT32F125x Reset Control Uint Driver
PWRCU HT32F125X Power Control Unit
ADC HT32F125x Analog to Digital Converter Driver
CMP_OP HT32F125x Operational Amplifier/Comparator Driver
EXTI HT32F125X External Interrupt/Event Controller Driver
FMC HT32F125x Flash Memory Controller Driver
GPIO HT32F125X General Purpose I/O Driver
GPTM HT32F125x General-Purpose Timers Driver
I2C HT32F125x Inter_Integrated Circuit Driver
RTC HT32F125X Real Time Clock Driver
SPI HT32F125x Serial Peripheral Interface
USART HT32F125X Universal Synchronous Asynchronous Receiver Transmitter Dri
WDT HT32F125X Watchdog Timer Driver
RTOS
CooCox OS CooCox OS V1.1.4 for Cortex M3 kernel
Manufacturers Chips Components
Outl Peri Hel
An outline is not available.
Sign in to CooCox... 0 items selected

Tipos y registros internos

- Los tipos básicos C99 se encuentran en *stdint.h*
  - este archivo puede ser incluido por el archivo <*micro*>.*h*, que incluye el soporte para todos los registros y periféricos particulares del micro

- #include "HT32F125x.h"

Debugging


Internal PPB bus
PPB (APB)
System (AHB)
ICODE (AHB)
DCODE (AHB)
Bus Matrix
AHB-AP
SW-DP
SWJ-DP
JTAG
SWD
Instrucciones
Datos
Core
Cortex-M3
ETM
ATB
TPIU
trace bus
SWV
DWT
ITM
ATB

Debugging

- JTAG
- SWD
- Trace
  - Requiere ETM en micro y JTAG c/trace, CARO
- SWV
  - printf(), pseudo-quasi trace económico

I/O: LED + switch

- Holtek HT32F125x
  - Keil
  - IAR
  - CooCox
- Fujitsu MB9BF506
  - Keil
  - IAR
- Toshiba TMPM330
  - Keil
  - IAR
- Fujitsu MB9BF618T
  - Keil

Systick: Timer de ARM Cortex-M

- Holtek HT32F125x
  - Keil
  - IAR
  - CooCox
- Fujitsu MB9BF506
  - Keil
  - IAR
- Toshiba TMPM330
  - Keil
  - IAR
- Fujitsu MB9BF618T
  - Keil

Estructuras en memoria: directivas de los compiladores

- Holtek HT32F125x
  - Keil
  - IAR
  - CooCox

NVIC: interrupciones

- Configurar interrupción en el periférico
- Habilitar en el NVIC
  - `NVIC_EnableIRQ(IRQn);`
- En el interrupt handler:
  - Bajar el flag de “pendiente”
  - El handler es una función C como cualquier otra, se la identifica por su nombre
    - *startup_<micro>.s* o *startup_<micro>.c*
      - contiene vectores de interrupción y reset; definidos como “weak”, de modo que una re-definición toma precedencia.