Tema 3:

El entorno de desarrollo Eclipse IDE for C/C++ Developers

Programación de sistemas y dispositivos

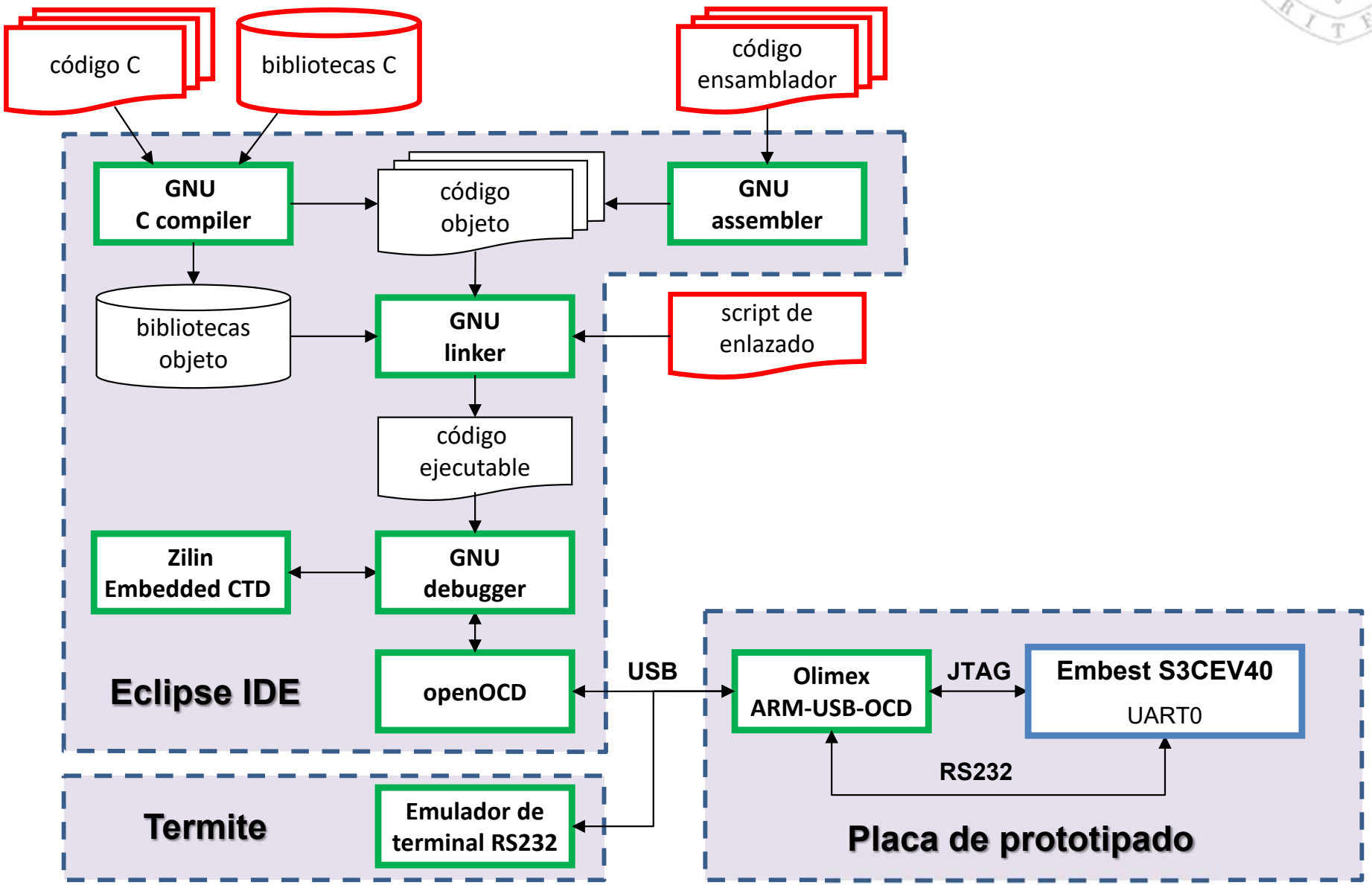
José Manuel Mendías Cuadros
Dpto. Arquitectura de Computadores y Automática
Universidad Complutense de Madrid

Contenidos

- ✓ Herramientas y flujo de desarrollo.
- ✓ Eclipse IDE.
- ✓ Toolchain.
- ✓ Aplicaciones volcadas en ROM.
- ✓ Aplicaciones volcadas en RAM.
- ✓ Organización del workspace.

Entorno de trabajo

flujo de desarrollo y depuración

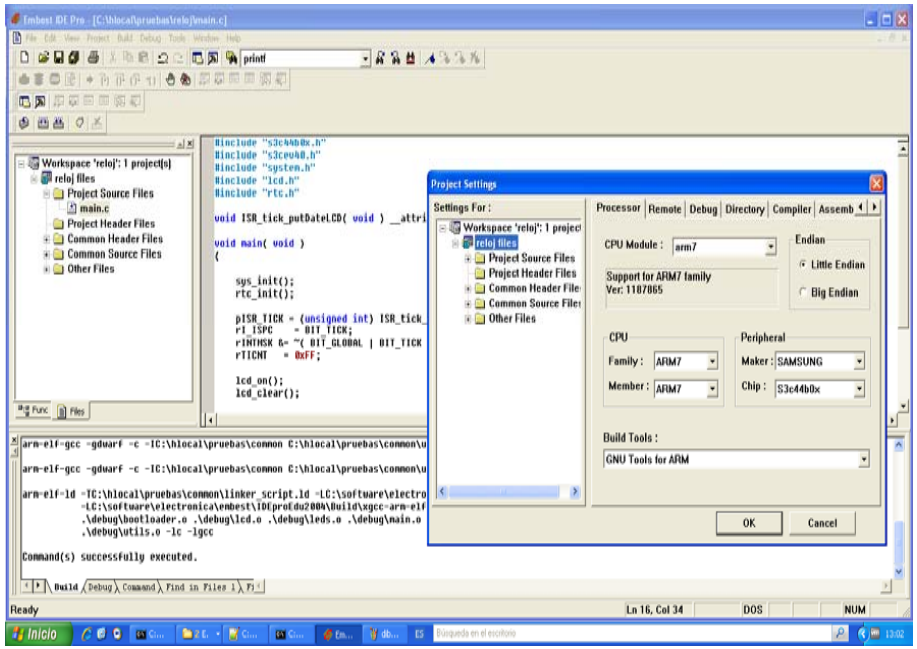


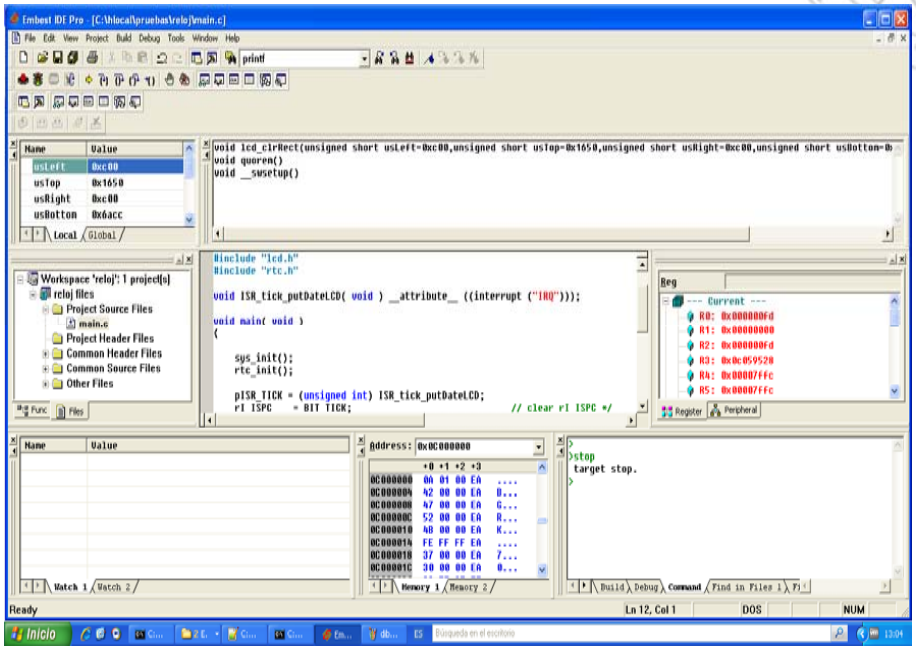
Entorno de trabajo

herramientas

- IDE: Eclipse Juno
  - Entorno gráfico para el desarrollo cruzado de aplicaciones
- Toolchain: Sourcery G++ Lite 2011.03-42
  - Compilador (GNU C Compiler 4.5.4): Compila código fuente C.
  - Ensamblador (GNU Assembler 2.20.51): Compila código fuente ensamblador.
  - Enlazador (GNU Linker 2.20.51): Genera código ejecutable.
  - Depurador (GNU Debugger 7.2.50) Permite volcar sobre la RAM del sistema una aplicación, controlar su ejecución y visualizar/modificar el contenido de registros/memoria.
  - Conversor de formatos (GNU BinUtils 2.20.51): Convierte un código ejecutable en formato ELF en otro en formato binario.
- Depurador on-chip: OpenOCD 0.7.0
  - Agente de depuración que efectúa, sobre el sistema físico, las acciones solicitadas por el depurador.
  - Además permite volcar sobre la ROM del sistema un código ejecutable en formato binario.
- Interfaz JTAG: Olimex ARM-USB-OCD-H
  - Adaptador hardware encargado de conectar físicamente el sistema hardware con el depurador.
- Emulador de plataforma: Zilyn Embedded CDT 4.18.1
  - Agente de depuración que efectúa, sobre un modelo de comportamiento del sistema, las acciones solicitadas por el depurador.
- Emulador de terminal serie: Termite 2.9

Eclipse IDE





- Entorno gráfico para el desarrollo cruzado de aplicaciones:
  - Gestión integral de opciones del proyecto.
  - Edición del código fuente.
  - Compilación y enlazado.
  - Depuración directa sobre el sistema.
  - Llamada a otras herramientas auxiliares

Toolchain

- ARM Sourcery Windows GCC C Compiler
  - Genera código objeto desde código fuente en lenguaje C.
  - Comando: **`arm-none-eabi-gcc`** <opciones> **`-o`** ***`file.o file.c`***
  - Opciones:
    - **`-I`** indica la ruta de los ficheros referenciados por la directiva #include
    - **`-O0`** no optimiza (reduce tiempo de compilación y facilita la depuración)
    - **`-c`** solo compila, no enlaza
    - **`-Wall`** habilita todos los warnings opcionales
    - **`-g3`** incluye máxima información para depuración
    - **`-gdwarf-2`** la info de depuración se genera en formato DWARF versión 2
    - **`-mapcs-frame`** genera marcos de activación de funciones según el estándar ARM
    - **`-fmessage-length=0`** no pone restricciones a la longitud de los mensajes de error
    - **`-mcpu=arm7tdmi`** indica la arquitectura objetivo
    - **`-MF`** ***`file.d`*** crea un fichero con información de dependencias

Toolchain

- ARM Sourcery Windows GCC C Assembler
  - Genera código objeto desde código fuente en lenguaje ensamblador
  - No se invoca directamente si no a través de GCC
  - Comando: **`arm-none-eabi-gcc`** <opciones> ***`-o file.o file.asm`***
    - **`-x assembler-with-cpp`** indica que el lenguaje del código fuente es ensamblador
    - **`-I`** indica la ruta de los ficheros referenciados por la directiva .include
    - **`-c`** solo compila, no enlaza
    - **`-Wall`** habilita todos los warnings opcionales
    - **`-g3`** incluye máxima información para depuración
    - **`-gdwarf-2`** la info de depuración se genera en formato DWARF versión 2
    - **`-fmessage-length=0`** no pone restricciones a la longitud de los mensajes de error
    - **`-mcpu=arm7tdmi`** indica la arquitectura objetivo
    - **`-Wa,-adhlns=`*`file.lst`*** crea un listado con información del fichero objeto creado
    - **`-MF`** ***`file.d`*** crea un fichero con información de dependencias

Toolchain

- ARM Sourcery Windows GCC C Linker
  - Genera un programa ejecutable combinando uno o más códigos objeto.
  - No se invoca directamente si no a través de GCC.
  - Comando: **`arm-none-eabi-gcc`** <opciones> **`-o file.elf file.o`**
    - **`-T file.ld`** indica el script de enlazado
    - **`-nostartfiles`** no usar ficheros estándar de startup del sistema
    - **`-Wl-Map file.map`** indica que cree un mapa de la aplicación
    - **`-g3`** incluye máxima información para depuración
    - **`-gdwarf-2`** la info de depuración se genera en formato DWARF versión 2
    - **`-mcpu=arm7tdmi`** indica la arquitectura objetivo
    - **`-L directory`** indica directorio donde encontrar bibliotecas
    - **`-l library`** indica nombre de biblioteca

Toolchain

- ARM Sourcery Windows GNU Create Flash Image
  - Copia el contenido de un fichero objeto a otro realizando algún tipo de transformación
  - Comando: **`arm-none-eabi-objcopy`** <opciones> ***`file.elf file.hex`***
  - Opciones:
    - **`-O binary`** el fichero destino será una imagen binaria de memoria

- ARM Sourcery Windows GNU Create Listing
  - Muestra información de un fichero objeto
  - Comando: **`arm-none-eabi-objdump`** <opciones> ***`file.elf > file.lst`***
  - Opciones:
    - **`-h`** muestra información extraída de las cabeceras de secciones
    - **`-S`** muestra el código fuente mezclado con el código ensamblador

- ARM Sourcery Windows GNU Print Size
  - Lista los tamaños de una de las secciones del fichero objeto
  - Comando: **`arm-none-eabi-size`** <opciones> ***`file.elf`***
    - **`--format=berkeley`** indica el formato del volcado

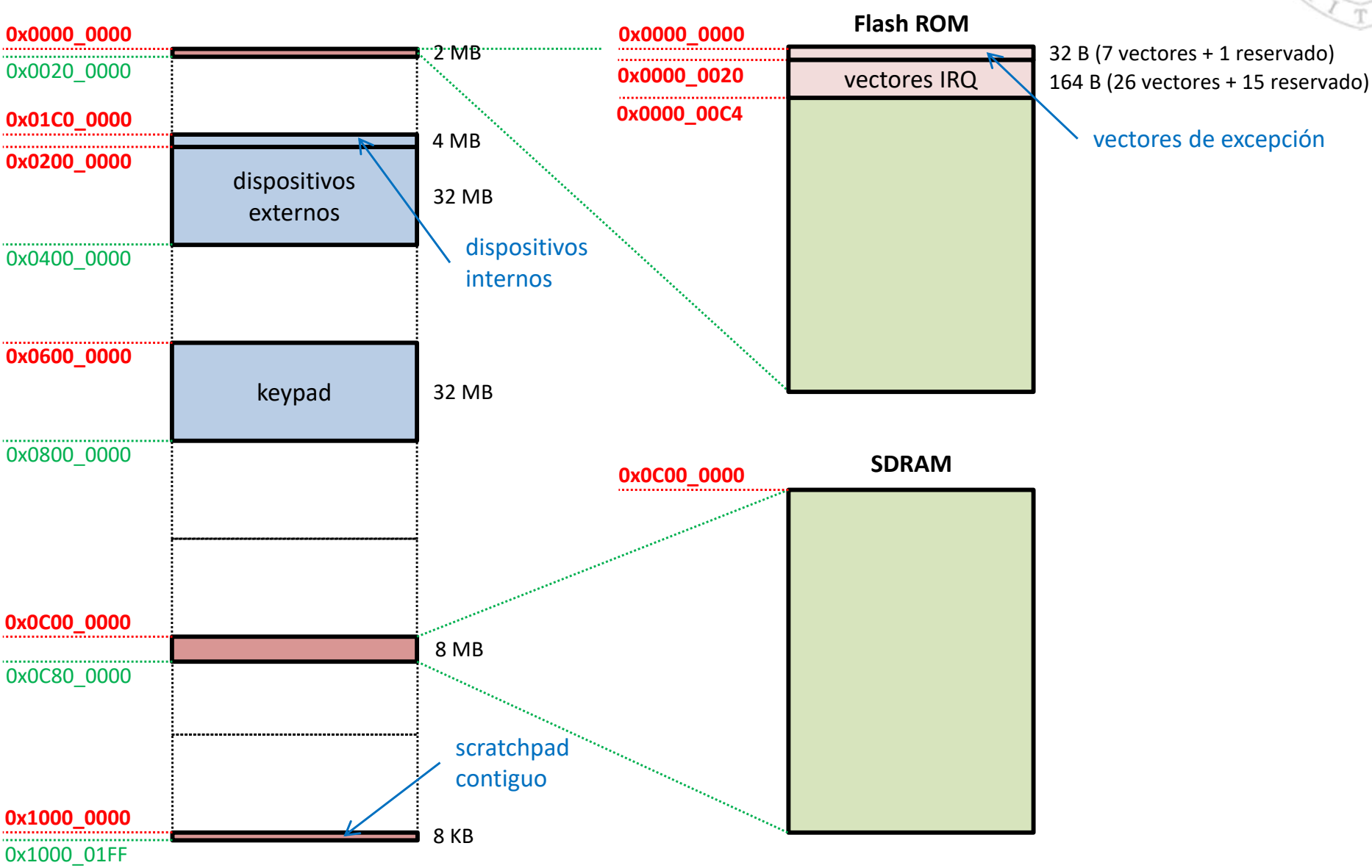
Aplicaciones volcadas en ROM

- Proyecto para volcado en ROM: es el tipo usado para la puesta en producción de una aplicación empotrada.
  - En la Flash ROM reside la aplicación.
    - La aplicación debe volcarse una única vez usando el OpenOCD y se arranca automáticamente tras cada reset o encendido (la dirección 0 es el punto de entrada).
  - La memoria SDRAM está vacía
- Un programa volcado sobre ROM:
  - Arranca en estado ARM, modo de operación SVC con IRQ y FIQ deshabilitadas.
  - Debe inicializar al completo del sistema (pilas, controladores, etc.): si la inicialización fuera errónea el sistema caería sin posibilidad de recuperación.
  - Al menos, debe capturar la interrupción de RESET
    - su vector debe apuntar a la dirección de inicio de la aplicación.
  - La tabla de vectores de excepción/interrupción se haya en su ubicación normal.
  - Su mapa lógico de memoria depende solamente de la arquitectura del microcontrolador y de la placa.

Tras finalizar la sesión de laboratorio **hay que restaurar la imagen original de la Flash** (para que los alumnos de 1º y 2º puedan usar la placa).

Aplicaciones volcadas en ROM

mapa lógico de memoria (i)



J.M. Mendías
2016

Aplicaciones volcadas en ROM

mapa lógico de memoria (ii)

tabla vectores excepción

fuente	dir. vector
Reset	0x0000_0000
Undefined instruction	0x0000_0004
Software interrupt	0x0000_0008
Abort (prefetch)	0x0000_000C
Abort (data)	0x0000_0010
Reservado	0x0000_0014
IRQ	0x0000_0018
FIQ	0x0000_001C

Cada entrada de la tablas debe contener una **instrucción de salto** a la ISR que corresponda.

tabla vectores IRQ

fuente	dir. vector
EINT0	0x0000_0020
EINT1	0x0000_0024
EINT2	0x0000_0028
EINT3	0x0000_002C
EINT4/5/6/7	0x0000_0030
TICK	0x0000_0034
Reservado	0x0000_0038
Reservado	0x0000_003C
ZDMA0	0x0000_0040
ZDMA1	0x0000_0044
BDMA0	0x0000_0048
BDMA1	0x0000_004C
WDT	0x0000_0050
UERR0/1	0x0000_0054
Reservado	0x0000_0058
Reservado	0x0000_005C
TIMER0	0x0000_0060
TIMER1	0x0000_0064
TIMER2	0x0000_0068
TIMER3	0x0000_006C
TIMER4	0x0000_0070

fuente	dir. vector
TIMER5	0x0000_0074
Reservado	0x0000_0078
Reservado	0x0000_007C
URxD0	0x0000_0080
URxD1	0x0000_0084
IIC	0x0000_0088
SIO	0x0000_008C
UTxD0	0x0000_0090
UTxD1	0x0000_0094
Reservado	0x0000_0098
Reservado	0x0000_009C
RTC	0x0000_00A0
Reservado	0x0000_00A4
Reservado	0x0000_00A8
Reservado	0x0000_00AC
Reservado	0x0000_00B0
Reservado	0x0000_00B4
Reservado	0x0000_00B8
Reservado	0x0000_00BC
ADC	0x0000_00C0

Aplicaciones volcadas en RAM

- Proyecto para volcado en RAM: es el tipo usado para el desarrollo y depuración de módulos de la aplicación empotrada.
  - En la Flash ROM reside un programa facilitado por el fabricante.
  - En la memoria SDRAM reside la aplicación que, tras cada apagado o reset (según la dirección de inicio), debe volcarse a través del IDE.
    - Porque es sobreescrita por el programa residente.

- Un programa volcado sobre RAM:
  - Arranca en estado ARM, modo de operación SVC y con IRQ y FIQ habilitadas.
  - Estrictamente, no requiere la inicialización del sistema (pilas, controladores, etc.): tras cada reset lo hace (según el uso) el programa residente en ROM.
  - No puede utilizar la excepción de RESET: la captura el programa residente en ROM.
  - La tabla de vectores de excepción/interrupción restantes está reubicada en RAM.
  - Su mapa lógico de memoria depende no solo de la arquitectura del microcontrolador y de la placa, sino también del programa residente.
  - Su punto de entrada puede estar en cualquier dirección.

J.M. Mendías
2016

Aplicaciones volcadas en RAM

programa residente en ROM

- El programa desarrollado por Embest y ubicado en la Flash ROM:
  - Captura la interrupción de RESET y pasa a ser ejecutada desde la ROM.
  - Inicializa la tabla de excepciones/interrupciones asociando a cada vector una ISR (*interrupt handler*) cuya única función es saltar a una dirección indicada en una tabla de excepciones/interrupciones virtual ubicada en SDRAM.
    - Permitirá que las aplicaciones volcadas en RAM puedan ubicar libremente sus ISR sin necesidad de reprogramar la Flash.
  - Configura el controlador de reloj y el controlador de memoria.
  - Inicializa las pilas de cada uno de los modos de ejecución protegidos.
  - Se autocopia a partir de la dirección 0x0C00_0000 de la SDRAM y pasa a ser ejecutado desde la RAM.
    - Sobrescribiendo lo que haya y pasándose a ejecutarse sobre RAM
  - Salta a un programa que indefinidamente hace una demo de la placa.
    - Inicializando los restantes controladores para un uso particular

Aplicaciones volcadas en RAM

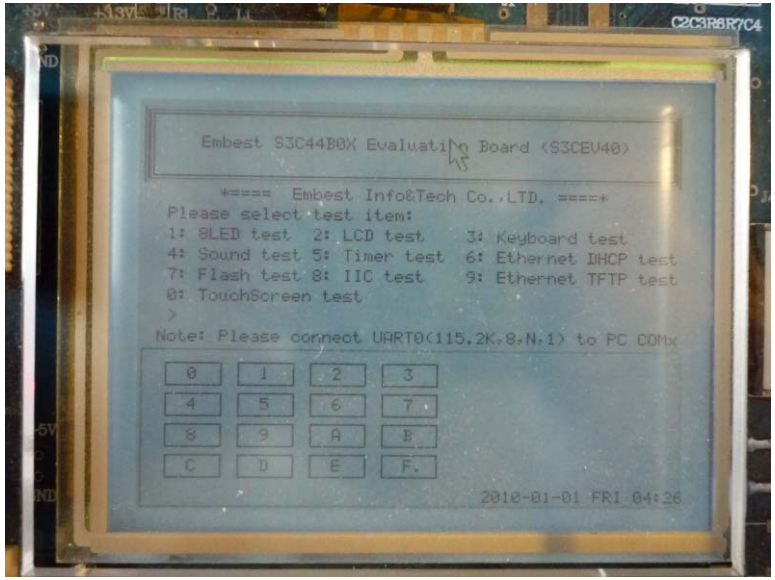
interrupt handlers

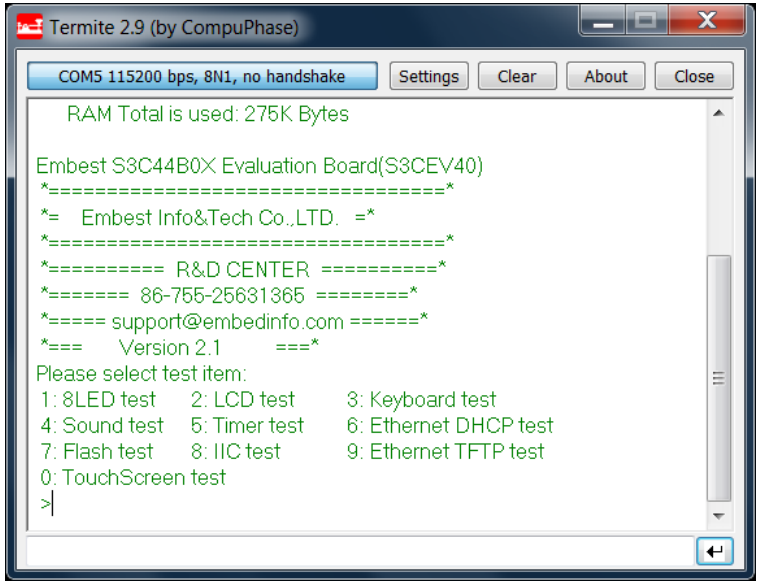
- La tabla de vectores de excepción/interrupción está en ROM
  - No es modificable por una aplicación volcada en RAM.
- Los *interrupt handlers* ofrecen una solución para que las aplicaciones volcadas en RAM puedan establecer sus propias ISR.
  - Crean en RAM una tabla virtual de vectores de excepción/interrupción.
  - El programador ubica en dicha tabla las direcciones de cada una de las ISR.
  - El handler es el encargado de capturar la excepción/interrupción y saltar a la ISR.
- Cuando se produce una excepción/interrupción
  - Por hardware se salta a la dirección del vector correspondiente (ubicado en ROM)
  - Por software se salta a la dirección de comienzo del handler (ubicado en ROM)
    - Hay un handler distinto para cada tipo de excepción/interrupción que ha sido enlazado en la tabla de excepciones/interrupciones por el programa residente.
  - El handler lee en la tabla virtual de excepciones/interrupciones (ubicada en RAM) la dirección de comienzo de la RTI correspondiente
  - El handler salta a la dirección leída, es decir, a la RTI apropiada (ubicada en RAM).
  - Al retorno de la RTI, retorna al flujo de programa.

Aplicaciones volcadas en RAM

programa demo

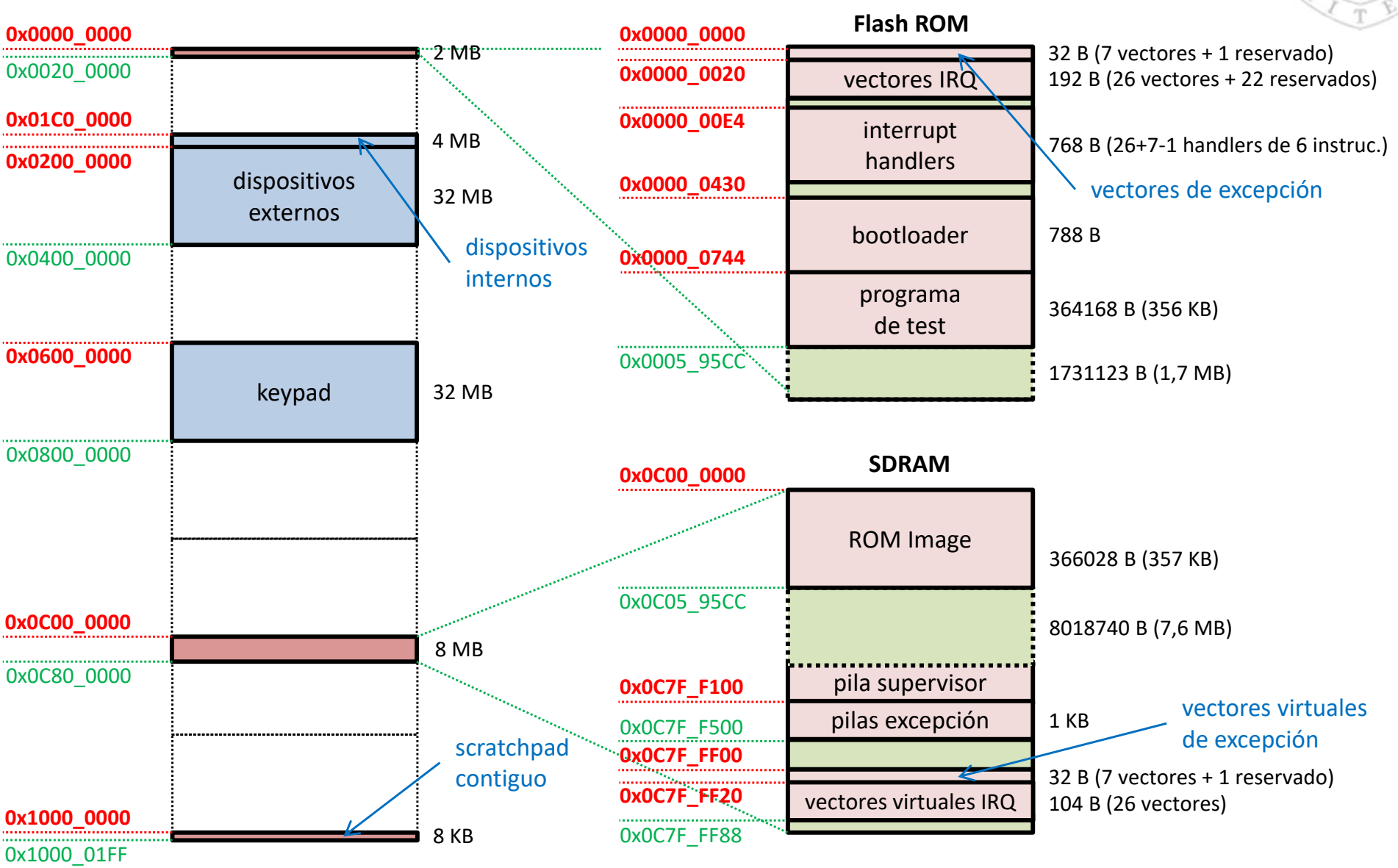
- El programa de demo desarrollado por Embest:
  - Enciende el display 7-segmentos.
  - Alterna el encendido del led izquierdo y derecho a una frecuencia de 1 Hz.
  - Muestra un menú de opciones en el LCD y lo envía por la UART0
    - Configuración UART0: 115200 baudios, 8 bits, sin paridad, 1 bit de stop.
  - En función de la tecla pulsada en el keypad o de lo recibido por la UART0 desde el ordenador ejecutará una rutina de demo de un elemento de la placa:
    - Leds, keypad, audio codec, temporizadores, disk, EEPROM, touchpad, ethernet





Aplicaciones volcadas en RAM

mapa lógico de memoria (i)



J.M. Mendías
2016

Aplicaciones volcadas en RAM

mapa lógico de memoria (ii)

tabla virtual vectores excepción

fuente	dir. vector
Reset	0x0C7F_FF00
Undefined instruction	0x0C7F_FF04
Software interrupt	0x0C7F_FF08
Abort (prefetch)	0x0C7F_FF0C
Abort (data)	0x0C7F_FF10
Reservado	0x0C7F_FF14
IRQ	0x0C7F_FF18
FIQ	0x0C7F_FF1C

tabla virtual vectores IRQ

fuente	dir. vector
ADC	0x0C7F_FF20
RTC	0x0C7F_FF24
UTxD1	0x0C7F_FF28
UTxD0	0x0C7F_FF2C
SIO	0x0C7F_FF30
IIC	0x0C7F_FF34
URxD1	0x0C7F_FF38
URxD0	0x0C7F_FF3C
TIMER5	0x0C7F_FF40
TIMER4	0x0C7F_FF44
TIMER3	0x0C7F_FF48
TIMER2	0x0C7F_FF4C
TIMER1	0x0C7F_FF50

fuente	dir. vector
TIMER0	0x0C7F_FF54
UERR0/1	0x0C7F_FF58
WDT	0x0C7F_FF5C
BDMA1	0x0C7F_FF60
BDMA0	0x0C7F_FF64
ZDMA1	0x0C7F_FF68
ZDMA0	0x0C7F_FF6C
TICK	0x0C7F_FF70
EINT4/5/6/7	0x0C7F_FF74
EINT3	0x0C7F_FF78
EINT2	0x0C7F_FF7C
EINT1	0x0C7F_FF80
EINT0	0x0C7F_FF84

pilas del sistema *

modo de operación	dir. base de pila	capacidad **
User / System (USR / SYS)	no inicializada (0x0)	---
Supervisor (SVC)	0x0C7F_F100	indefinida
Undefined (UND)	0x0C7F_F200	256B
Abort (ABT)	0x0C7F_F300	256B
IRQ	0x0C7F_F400	256B
FIQ	0x0C7F_F500	256B

Cada entrada de las tabla debe contener la **dirección de inicio** de la ISR que corresponda.

(*) Según el ARM Architecture Procedure Call Standard las pilas crecen hacia direcciones bajas usando como puntero el R13.
(**) 1 contexto completo ocupa 64B

Volcado de aplicaciones

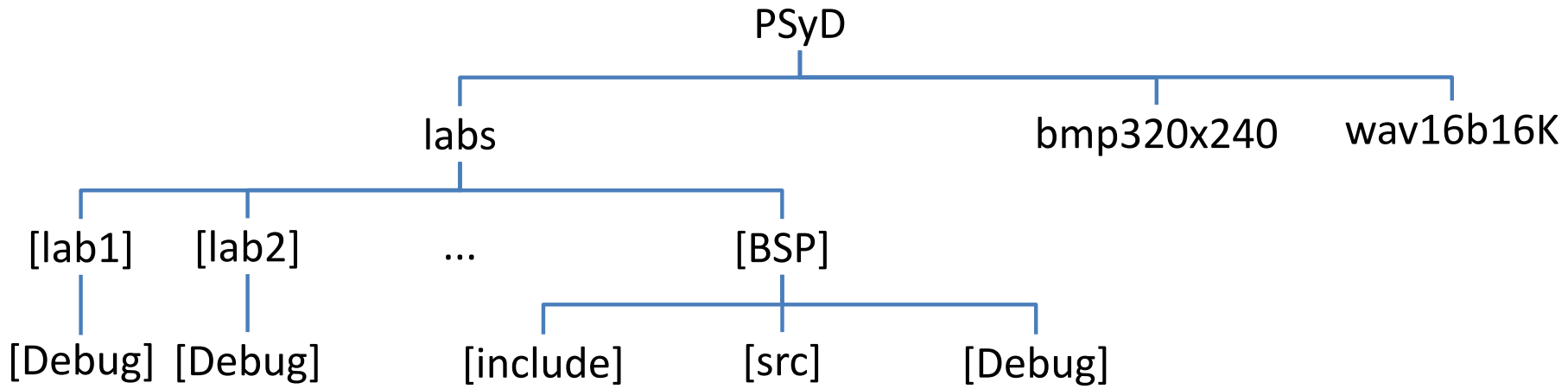
flujo de tareas genérico

1. Crear proyecto: **`File > New > C Project`**
2. Configurar las opciones de compilación y enlazado: **`File > Properties`**
   - Las opciones básicas están en el documento checklist-desarrollo.pdf
3. Copiar ficheros en la carpeta del proyecto
   - Código fuente y script de enlazador
4. Refrescar el proyecto: **`File > Refresh`**
5. Compilar proyecto: **`Project > Built project`**
6. Crear una configuración de depuración: **`Run > Debug Configurations...`**
   - Las opciones básicas están en el documento checklist-desarrollo.pdf
7. Conectar físicamente el IDE y la placa
8. Conectar lógicamente el IDE y la placa: **`Run > External Tools > OpenOCD`**
9. Descargar en RAM el programa ejecutable: **`Run > Debug`**
10. Arrancar la ejecución / depuración: **`Debug > Resume / Step ...`**

Para volcado en ROM, tras 7 ejecutar el comando indicado en checklist-desarrollo.pdf

Organización del workspace

- La estructura de directorios la asignatura será:



- Para cada laboratorio existe un proyecto en el workspace:
  - Carpeta raíz: contiene los ficheros fuente de la aplicación.
  - Debug: ubicada dentro de la anterior, conteniendo los ficheros objeto y los ejecutables.
- Progresivamente iremos desarrollando una biblioteca de firmware denominada BSP (Board Support Package), con los siguientes directorios:
  - include: contiene los ficheros cabecera de cada uno de los módulos de la biblioteca.
  - src: contiene los ficheros fuente de cada uno de los módulos de la biblioteca.
  - Debug: contiene el código objeto de la biblioteca.

Acerca de *Creative Commons*

- Licencia CC (Creative Commons)

  - Ofrece algunos derechos a terceras personas bajo ciertas condiciones. Este documento tiene establecidas las siguientes:

Reconocimiento (*Attribution*):
En cualquier explotación de la obra autorizada por la licencia hará falta reconocer la autoría.

No comercial (*Non commercial*):
La explotación de la obra queda limitada a usos no comerciales.

Compartir igual (*Share alike*):
La explotación autorizada incluye la creación de obras derivadas siempre que mantengan la misma licencia al ser divulgadas.

Más información: https://creativecommons.org/licenses/by-nc-sa/4.0/