Desarrollo de sistemas embebidos en framework multiplataforma

Autor: Mg. Lic. Santiago Germino.

embedul.ar

Disertante

Mg. Lic. Santiago Germino (UBA/CONICET)

https://www.linkedin.com/in/royconejo

- De donde vengo.
- A que me dedico.

Introducción

Sistemas embebidos
Framework multiplataforma

Sistemas embebidos

- Conjunto de hardware y software.
- Resuelven un problema o necesidad puntual.
- Generalmente restringidos en costo, potencia y/o consumo.
- Eficiencia altamente valorada.
- Sistemas críticos embebidos (safety, mission, business, security).

Framework multiplataforma

Breve historia:

- Origen.
- Necesidad.
- Objetivos.

Primeros trabajos

RETRO-CIAA - Defensa Trabajo Final del Posgrado en Sistemas Embebidos (CESE FI-UBA) - 18/12/2018

https://youtu.be/Z3_VP16FbQA

Implementación de un adaptador de video por software en microcontrolador de doble núcleo

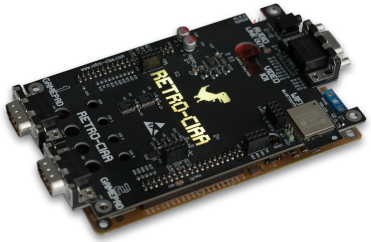
http://elektron.fi.uba.ar/index.php/elektron/article/view/91

Hardware inicial

Computadora Industrial Abierta Argentina

EDU-CIAA-NXP

http://www.proyecto-ciaa.com.ar/

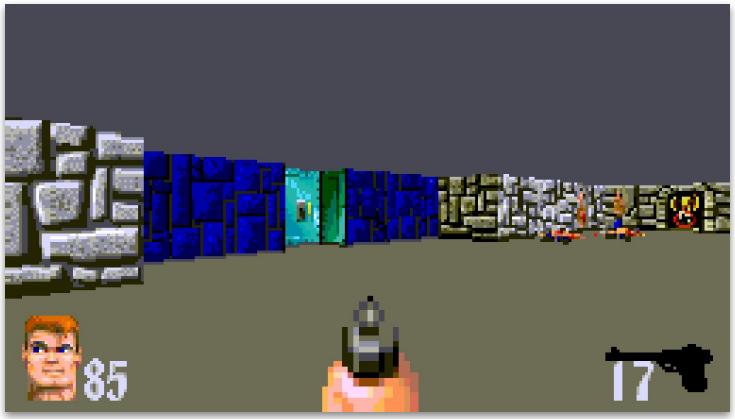


Poncho con conector de video, DAC de sonido y joysticks.

https://www.retro-ciaa.com

Primer caso de uso (2019)

- Aplicación compleja.
- Altos requerimientos de memoria, almacenamiento y procesamiento.
- En el límite de lo posible.



Entorno de desarrollo

- Linux.
- VSCodium.

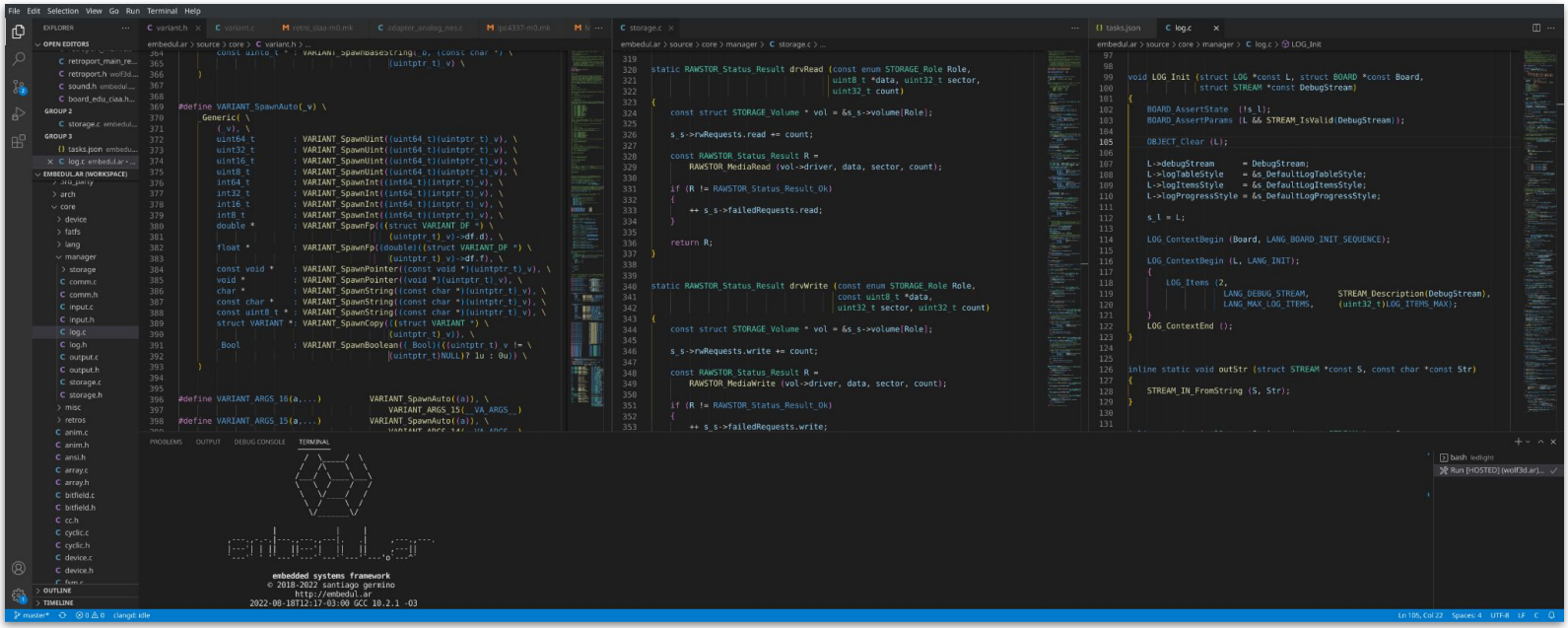
  Versión libre de Visual Studio Code.
- GCC. Arm® GCC Toolchain. LLVM/Clang.
- Python 3.

Entorno de desarrollo

VSCodium
Arm® GCC Toolchain
Herramientas

Entorno de desarrollo

VSCodium


File Edit Selection View Go Run Terminal Help
EXPLORER
OPEN EDITORS
EMBEDUL.AR (WORKSPACE)
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
embedded systems framework
© 2018-2022 santiago germino
http://embedul.ar
2022-08-18T12:17-03:00 GCC 10.2.1 -03

Instalación

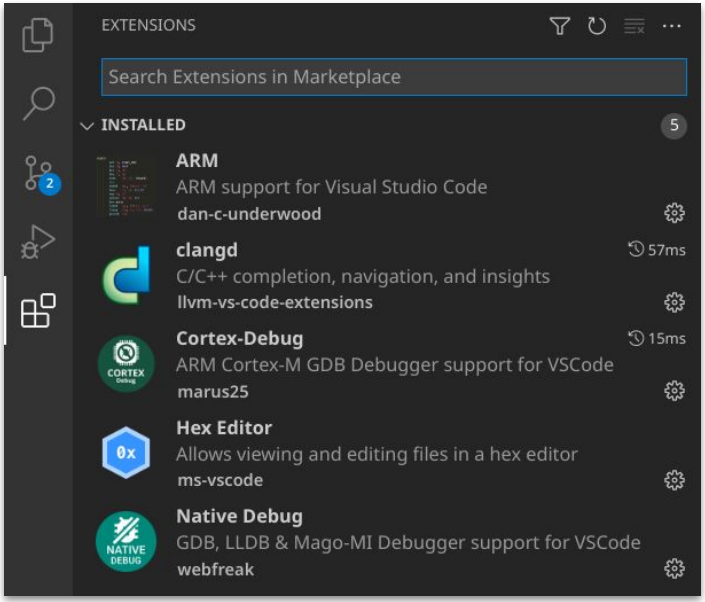
https://vscodium.com/

Seguir instrucciones en “Debian / Ubuntu (deb package)”.

VSCodium logo author: Christophe Larsonneur (MIT License)

Requerimientos

- Distribución Linux actualizada.
  Debian 11, Ubuntu 22.04 LTS
- Extensiones: ARM, clangd, Cortex-Debug, Hex Editor, Native Debug.



Entorno de desarrollo

Arm® GCC Toolchain

Arm® GNU Toolchain

arm Developer

https://developer.arm.com/downloads/-/arm-gnu-toolchain-downloads

x86_64 Linux hosted cross toolchains
AArch32 bare-metal target (arm-none-eabi)

- Descomprimir.
  sudo tar -xvf arm-gnu-toolchain-{...}.tar.xz -C /opt
- Agregar ruta al directorio *bin* de *arm-none-eabi-gcc*.
  export PATH=$PATH:/opt/arm-gnu-toolchain-{...}/bin

Entorno de desarrollo

Herramientas

Herramientas

Para build, flash y debug:

- apt install git make bear gcc openocd
  openocd versión 0.11 en adelante.

Para generar documentación:

- apt install python3-pip
- pip3 install sphinx hawkmoth sphinx_rtd_theme

VSCodium en sistemas embebidos

VSCodium logo author: Christophe Larsonneur (MIT License)

Conceptos básicos

- Cada proyecto *es una carpeta* (directorio).
- Cada carpeta contiene:

    - Archivo *Makefile* del proyecto.
    - Archivos de código.
    - Carpeta oculta .vscode.

VSCodium logo author: Christophe Larsonneur (MIT License)

Conceptos básicos (cont'd)

VSCode debe invocar al Makefile del proyecto.

¿Como se logra?

VSCodium logo author: Christophe Larsonneur (MIT License)

Subcarpeta .vscode

Contiene al menos tres archivos en formato JSON.

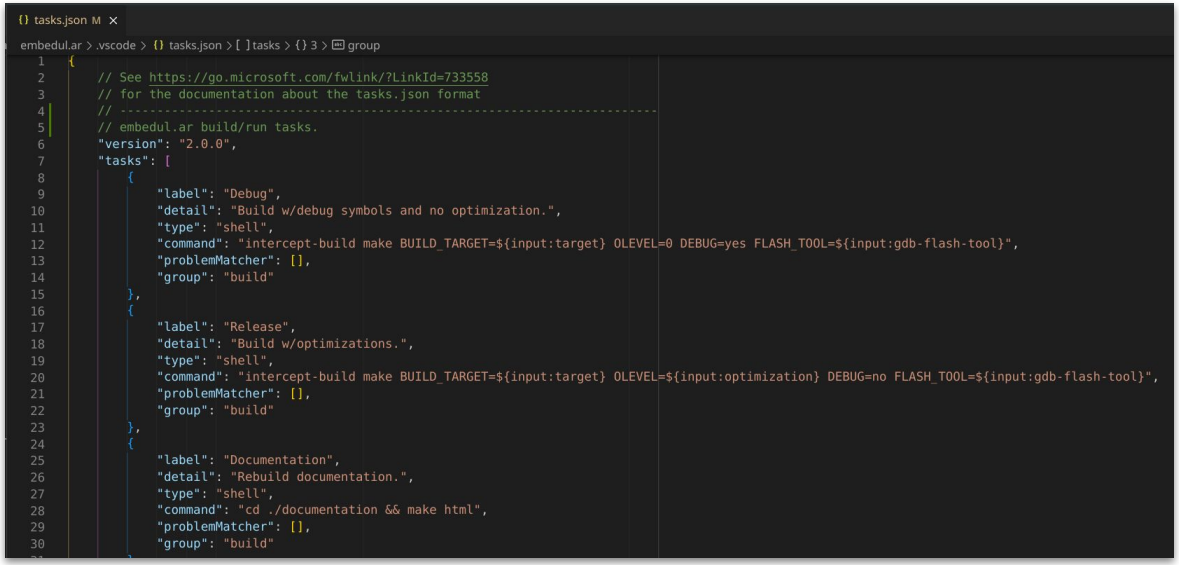
- tasks.json
- launch.json
- settings.json

tasks.json

- Específica tareas a ejecutar.
- Generalmente se invocan programas.
  Por ejemplo, make.
- Posibilidad de parámetros opcionales.
  Target hardware, nivel de optimización, etc.

VSCodium logo author: Christophe Larsonneur (MIT License)

task.json (cont'd)



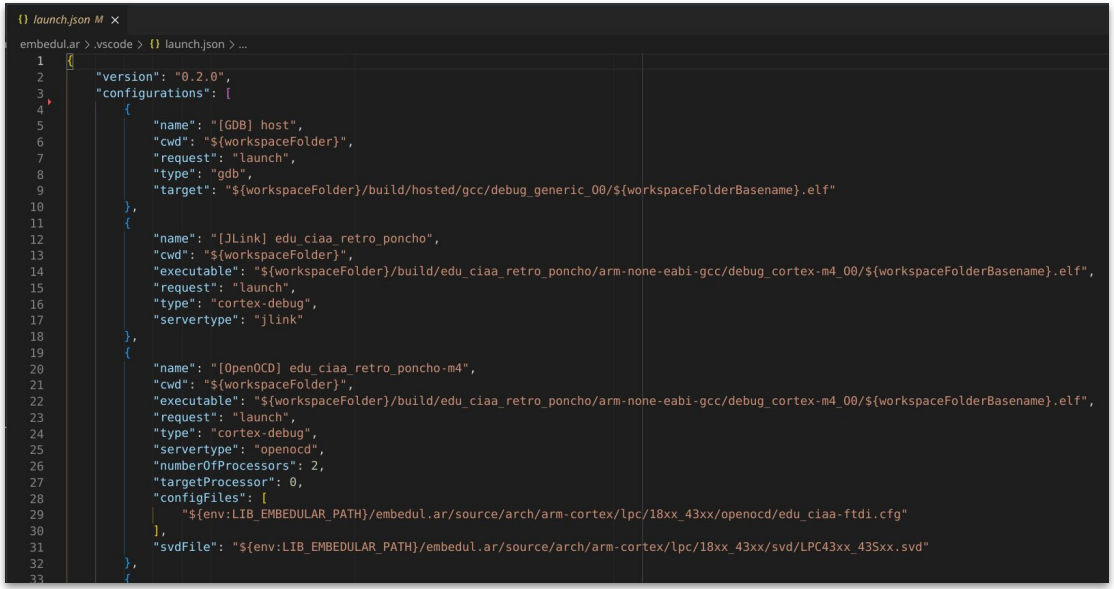
launch.json

- Configura sesiones de depuración.
- Se especifica tipo de sesión.

  GDB, cortex-debug…
- Parámetros según tipo de sesión.

  Por ejemplo en cortex-debug, servidor GDB: openocd, JLink...
- Ejecutable.

Launch.json (cont'd)



settings.json

- Preferencias del editor.

  Por ejemplo, el código de embedul.ar sigue un formato de 80 columnas y 4 espacios por tab.
- Configuración de las extensiones.

¿Qué es un *workspace*?

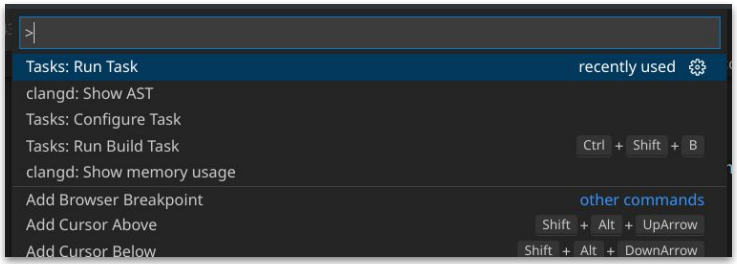
- Se pueden abrir varios proyectos (varias carpetas) en el mismo *workspace*.
- Archivos abiertos en el editor.
- Configuración de la vista.

VSCodium en sistemas embebidos

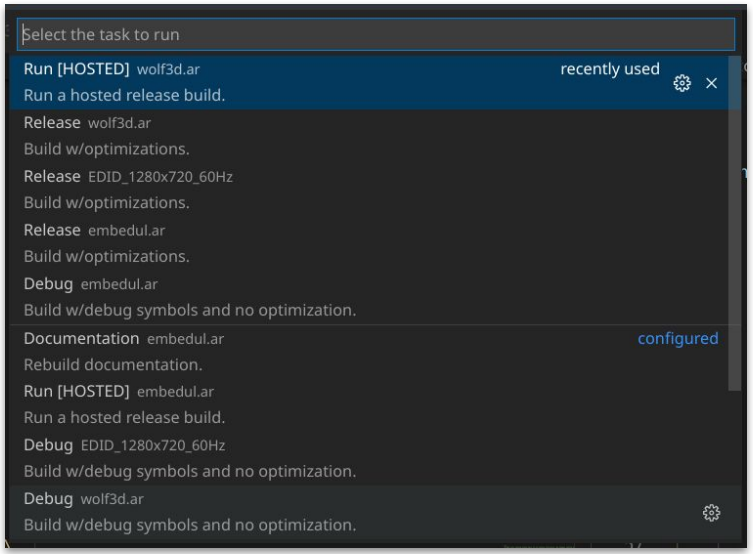
Ejecutar tareas y depuración

Ejecutar tareas

Command palette: Ctrl+Shift+P

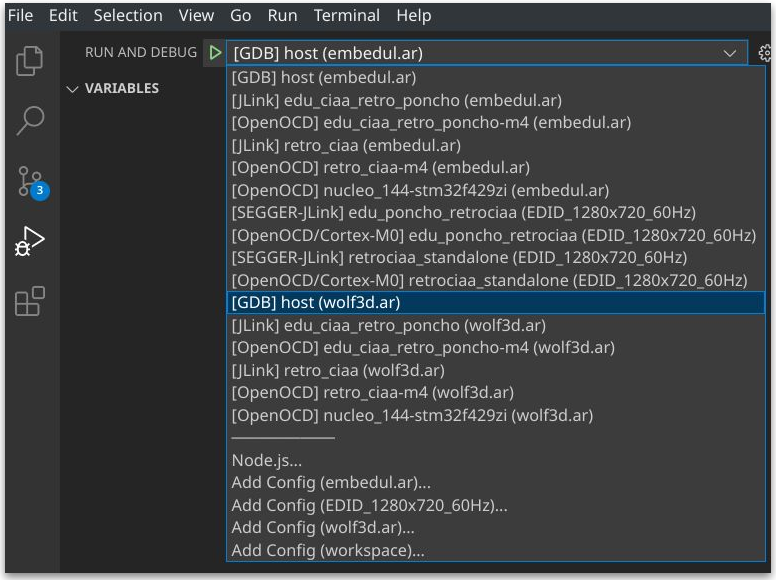


Build menu: Ctrl+Shift+B



Ejecutar sesión de depuración

- Ctrl+Shift+D
- Elegir la configuración deseada en el proyecto correcto.
  ¡Difícil en Workspaces con varios proyectos abiertos!



embedul.ar

embedul.ar logo author: Santiago Germino (CC BY-SA 4.0)

embedul.ar

- Framework open source para sistemas embebidos.
- Multiplataforma, targets diversos.
  (Nativo o cross-compiled para microcontroladores, web, etc...)
- Diseño modular orientado a objetos.
- Lenguaje de programación: C11.

http://github.com/royconejo/embedul.ar

embedul.ar (cont'd)

- Capa de abstracción transparente para la aplicación.
  Compila sin modificaciones en cualquier target.
- Licencia zlib/libpng.
  Considerada "free software" por la Debian Free Software Guidelines, Open Source Initiative y Free Software Foundation.
- Sistema de build anidado, construido sobre GNU Make.

embedul.ar (cont'd)

- Solo Stack. No utiliza memoria Heap.
  No malloc(), free(), etc.
- Soporta codificación UTF-8 en todas sus funciones.
  こんにちは世界!
- Programación defensiva.
  Por ejemplo, Bounds checking en cada función.

embedul.ar (cont'd)

- Apunta a cumplir rigurosos estándares de sistemas críticos. *MISRA-C*, *CERT C*.
- Contenedores básicos funcionan sobre memoria estática.
- Tipos de dato "Variant".
- Funciones con cantidad automática y tipo automático de parámetros, validados en tiempo de compilación (sí, en "C").

embedul.ar (cont'd)

- Documentación.


embedul.ar logo author: Santiago Germino (CC BY-SA 4.0)

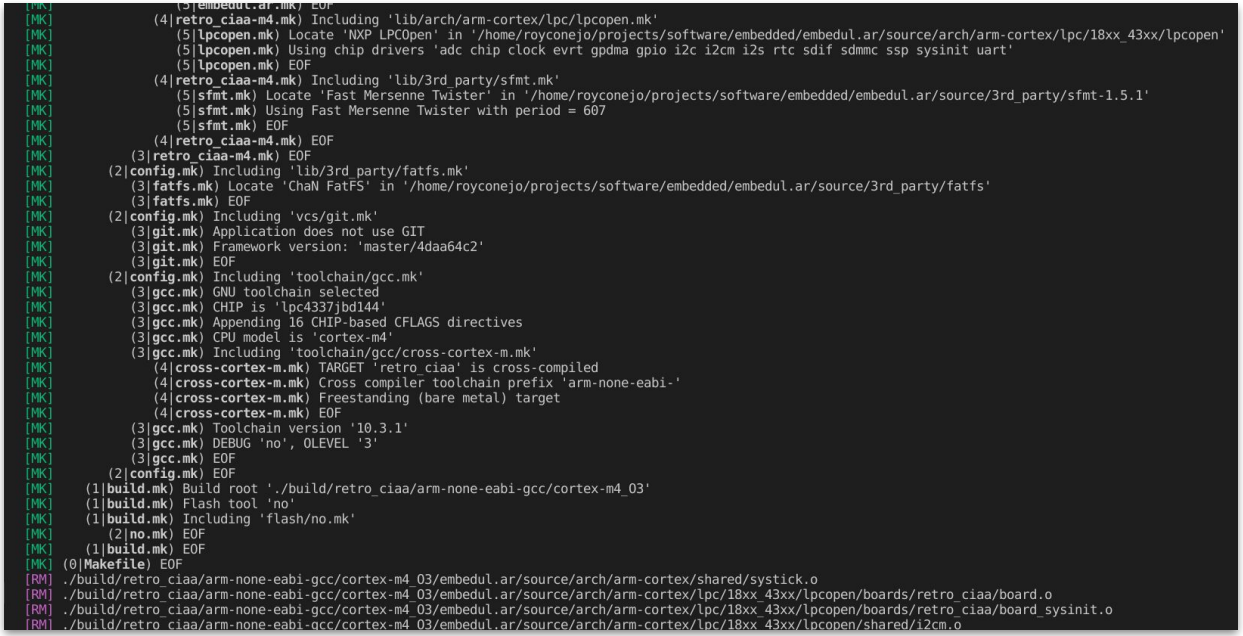
embedul.ar

Sistema de build

Sistema de build

```
[MK] (0|Makefile)
[MK]    (1|system.mk) Welcome to http://embedul.ar build system
[MK]    (1|system.mk) Date/time is '2022-08-19T06:01-03:00'
[MK]    (1|system.mk) Build system ready
[MK]    (1|system.mk) EOF
[MK] (0|Makefile) Including 'build.mk'
[MK]    (1|build.mk) Build not configured yet
[MK]    (1|build.mk) Including 'config.mk'
[MK]       (2|config.mk) App. name 'wolf3d.ar'
[MK]       (2|config.mk) Requested target shortcut 'retro_ciaa-m4'
[MK]       (2|config.mk) Using makefiles 'target/shortcut/retro_ciaa-m4 lib/3rd_party/fatfs vcs/git toolchain/gcc'
[MK]       (2|config.mk) Including 'target/shortcut/retro_ciaa-m4.mk'
[MK]          (3|retro_ciaa-m4.mk) Including 'lib/arch/arm-cortex/lpc/board/retro_ciaa-m4.mk'
[MK]             (4|retro_ciaa-m4.mk) Including 'lib/arch/arm-cortex/lpc/shared/lpc4337-m4.mk'
[MK]                (5|lpc4337-m4.mk) Including 'cpu/arch/arm-cortex/lpc/lpc4337jbd144_m4.mk'
[MK]                   (6|lpc4337jbd144_m4.mk) Declaring chip manufacturer 'lpc', family '43', model '37', variant 'jbd144'
[MK]                   (6|lpc4337jbd144_m4.mk) Including 'cpu/arch/arm-cortex/m4f.mk'
[MK]                      (7|m4f.mk) Including 'cpu/arch/arm-cortex/m4.mk'
[MK]                         (8|m4.mk) EOF
[MK]                      (7|m4f.mk) EOF
[MK]                   (6|lpc4337jbd144_m4.mk) EOF
[MK]                (5|lpc4337-m4.mk) Including 'lib/arch/arm-cortex/lpc/shared/lpc4337.mk'
[MK]                   (6|lpc4337.mk) Including 'target/check.mk'
[MK]                      (7|check.mk) Using target 'retro_ciaa'
[MK]                      (7|check.mk) EOF
[MK]                   (6|lpc4337.mk) EOF
[MK]                (5|lpc4337-m4.mk) EOF
[MK]             (4|retro_ciaa-m4.mk) Including 'lib/embedul.ar.mk'
[MK]                (5|embedul.ar.mk) Declaring library 'embedul.ar'
[MK]                (5|embedul.ar.mk) Locate 'embedul.ar' in '/home/royconejo/projects/software/embedded/embedul.ar/source/core'
[MK]                (5|embedul.ar.mk) Visual boot enabled
[MK]                (5|embedul.ar.mk) Using CORE modules 'anim array bitfield cyclic device main object sequence utf8 variant'
[MK]                (5|embedul.ar.mk) Using SUBSYSTEMS 'sound video'
[MK]                (5|embedul.ar.mk) Using FatFs custom disks and config
[MK]                (5|embedul.ar.mk) Exporting '2' user defined, '3' default config params
[MK]                (5|embedul.ar.mk) Config param 'VISUAL_BOOT=1'
[MK]                (5|embedul.ar.mk) Config param 'INPUT_SWITCH_ACTION=0'
[MK]                (5|embedul.ar.mk) Config param 'SPLASH_THEME_L1=lock_glyph'
[MK]                (5|embedul.ar.mk) Config param 'SPLASH_THEME_L2=none'
[MK]                (5|embedul.ar.mk) Config param 'SPLASH_THEME_L3=c64'
[MK]                (5|embedul.ar.mk) EOF
```

Sistema de build (cont'd)



Makefile simple

embedul.ar > Makefile

```
include $(LIB_EMBEDULAR_PATH)/embedul.ar/makefiles/system.mk

LIB_EMBEDULAR_CONFIG := VISUAL_BOOT=1 \
                        SPLASH_THEME_L3=nyan_cat

BUILD_LIBS += 3rd_party/fatfs

APP_OBJS += \
    ./testing/main.o

$(BUILD)

```

Makefile “no tan simple”

wolf3d.ar > Makefile

```
include $(LIB_EMBEDULAR_PATH)/embedul.ar/makefiles/system.mk

LIB_EMBEDULAR_CONFIG := VISUAL_BOOT=1 \
                        INPUT_SWITCH_ACTION=0

BUILD_LIBS += 3rd_party/fatfs

# CFLAGS += -DSPANISH

lpc4337jbd144_CFLAGS += -DBOARD_RETRO_CIAA_DISABLE_TCP_SERVER

lpc4337jbd144_CFLAGS += -DRODATA_SECTION_GRCHUNK_PIC_332='CC_Section(".rodata.$$Flash2")'
lpc4337jbd144_CFLAGS += -DRODATA_SECTION_MAPSEGS_PLANE_0='CC_Section(".rodata.$$Flash2")'
lpc4337jbd144_CFLAGS += -DRODATA_SECTION_MAPSEGS_PLANE_1='CC_Section(".rodata.$$Flash2")'
#lpc4337jbd144_CFLAGS += -DRODATA_SECTION_AUDIOSEGS_MUSIC_ADLIB='CC_Section(".rodata.$$Flash2")'
lpc4337jbd144_CFLAGS += -DRODATA_SECTION_AUDIOSEGS_SOUND_ADLIB='CC_Section(".rodata.$$Flash2")'
#lpc4337jbd144_CFLAGS += -DRODATA_SECTION_TEXTURE_PAGE_332='CC_Section(".rodata.$$Flash2")'
lpc4337jbd144_CFLAGS += -DRODATA_SECTION_SPRITE_PAGE_332='CC_Section(".rodata.$$Flash2")'
#lpc4337jbd144_CFLAGS += -DBSS_SECTION_OBJLIST='CC_Section(".bss.$$RamAHB_ETB16")'
lpc4337jbd144_CFLAGS += -DBSS_SECTION_TILEMAP='CC_Section(".bss.$$RamAHB_ETB16")'
lpc4337jbd144_CFLAGS += -DBSS_SECTION_ACTORAT='CC_Section(".bss.$$RamLoc40")'
lpc4337jbd144_CFLAGS += -DBSS_SECTION_STATOBJLIST='CC_Section(".bss.$$RamAHB16")'
# lpc4337jbd144_CFLAGS += -DBSS_SECTION_VISLIST='CC_Section(".bss.$$RamAHB16")'
#lpc4337jbd144_CFLAGS += -DBSS_SECTION_AREACONNECT='CC_Section(".bss.$$RamAHB16")'
#lpc4337jbd144_CFLAGS += -DBSS_SECTION_WALLHEIGHT='CC_Section(".bss.$$RamAHB16")'
#lpc4337jbd144_CFLAGS += -DBSS_SECTION_DOOROBJLIST='CC_Section(".bss.$$RamAHB16")'
#lpc4337jbd144_CFLAGS += -DBSS_SECTION_SPOTVIS='CC_Section(".bss.$$RamAHB16")'
#lpc4337jbd144_CFLAGS += -DRODATA_SECTION_ASSETS_TILE_hud='CC_Section(".rodata.$$Flash2")'
#lpc4337jbd144_CFLAGS += -DRODATA_SECTION_ASSETS_TILEMAP_hud='CC_Section(".rodata.$$Flash2")'
```

Makefile “no tan simple” (cont’d)

```
30  lpc4337jbd144_CFLAGS += -DRODATA_SECTION_ASSETS_SFX_SoundChunks_0='CC_Section(".rodata.$$Flash2")'
31  lpc4337jbd144_CFLAGS += -DRODATA_SECTION_ASSETS_SFX_SoundChunks_1='CC_Section(".rodata.$$Flash2")'
32  lpc4337jbd144_CFLAGS += -DRODATA_SECTION_ASSETS_SFX_SoundChunks_2='CC_Section(".rodata.$$Flash2")'
33  lpc4337jbd144_CFLAGS += -DBSS_SECTION_RETROCIAA_SYSTEM='CC_Section(".bss.$$RamAHB16")'
34  lpc4337jbd144_CFLAGS += -DRODATA_SECTION_RETROCIAA_SPLASH_THEME='CC_Section(".rodata.$$Flash2")'
35  lpc4337jbd144_CFLAGS += -DRODATA_SECTION_ASSETS_TILEMAP_nyancat='CC_Section(".rodata.$$Flash2")'
36  lpc4337jbd144_CFLAGS += -DRODATA_SECTION_ASSETS_TILE_nyancat='CC_Section(".rodata.$$Flash2")'
37
38
39  # Project files to build
40  APP_OBJS += \
41      ./retroport.o \
42      ./retroport_main.o \
43      ./id_ca.o \
44      ./id_in.o \
45      ./id_pm.o \
46      ./id_sd.o \
47      ./id_us_1.o \
48      ./id_vh.o \
49      ./id_vl.o \
50      ./wl_act1.o \
51      ./wl_act2.o \
52      ./wl_agent.o \
53      ./wl_debug.o \
54      ./wl_draw.o \
55      ./wl_game.o \
56      ./wl_inter.o \
57      ./wl_main.o \
58      ./wl_menu.o \
59      ./wl_play.o \
60      ./wl_state.o \
61      ./wl_text.o \
62      ./wl_def.o \
63      ./rom/wl1/rom.o
64
65  $(BUILD)
```

embedul.ar

Diseño modular orientado a objetos

Diseño modular orientado a objetos

- Objetos conceptualmente *devices* o dispositivos.
- El framework define la clase base y una interfaz (abstract base class).
- Los *drivers* implementan la interfaz privada del *device*.
- La implementación del *driver* se invoca a través del *device* (inheritance).

Diseño modular orientado a objetos (cont'd)

- Las implementaciones dependientes de la plataforma, o las dependientes de librerías puntuales, están encapsuladas en sus propias definiciones, -y además-
- La clase base de cualquier *device* debe ser tratada como estructuras opacas (encapsulation).

    En "C" hay formas de volverlas realmente opacas, pero no pregunten.

Diseño modular orientado a objetos (cont'd)

- Mediante un *pointer type cast* de un *driver* a su tipo de *device*, se accede igualmente a los servicios de ese objeto. También funciona al revés (polymorphism).
- Un tipo de device puede tener varias instancias, o una sola (singleton).
- Cualquier device puede tener múltiples implementaciones.

embedul.ar

Listado de devices

BOARD

Abstracción del sistema.

“El sistema” puede ser un microcontrolador, una aplicación corriendo en un sistema operativo o en una página web, etc.

Gracias a esto, embedul.ar soporta targets diversos.

PACKET

Abstracción de transmisión o recepción de datos estructurados.

Los “datos estructurados” pueden ser una comunicación por TCP/IP, SPI, I2C, etc.

Es decir, embedul.ar abstrae tanto protocolos de comunicación de software y de hardware, como SPI e I2C.

RAWSTOR

Abstracción de lectura y escritura de unidades de información (sectores) en dispositivos de almacenamiento.

Los “dispositivos de almacenamiento” pueden ser cualquier cosa.

embedul.ar abstrae el acceso a cualquier dispositivo de almacenamiento, incluso a través de fatfs.

SOUND

Abstracción de dispositivo de sonido.

Soporta efectos de sonido y música BGM leída mediante streaming desde un RAWSTOR.

No importa que el dispositivo implementado sea un DAC o la librería SDL.

VIDEO

Abstracción de dispositivo de video.

Además del driver dependiente de la plataforma, embedul.ar incluye:

- Funciones para convertir y dibujar sprites animados y con transparencias.
- Áreas de clipping de pantalla.
- Dibujo de texto UTF-8.
- Lo básico, líneas, rectángulos…

Todo esto independiente de la plataforma.

IO

Abstracción de dispositivo de entrada/salida.

Dispositivos con cualquier cantidad de entradas y/o salidas, digitales o analógicas. Se pueden implementar los clásicos:

- Teclado.
- Mouse.
- Joysticks.

Pero también cualquier tipo de sensores:

- Acelerómetro.
- Giróscopo.
- Temperatura.
- Humedad.
- Luz.

O actuadores:

- Motores.
- Luces LED, incluso SSL.

Todo en el mismo o en diferentes drivers.

RANDOM

Abstracción de dispositivo de generador de números aleatorios.

Por hardware o software, dependiente o no de la arquitectura o de librerías externas.

Como driver independiente de la plataforma, embedul.ar implementa el dispositivo "RANDOM_SFMT" (SIMD Fast Mersenne Twister).

STREAM

Abstracción de dispositivo de envío y recepción de datos serie byte a byte.

Puede ser “stdout/stdin” en sistemas hosted, o una implementación de una UART en sistemas embebidos.

De este modo, embedul.ar abstrae toda comunicación serie.

embedul.ar

Managers

Managers

- Se ocupan de gestionar los *drivers* de cada sistema.
- Durante la secuencia de inicio, el sistema o *board* registra sus *drivers* en los managers correspondientes.
- La aplicación invoca a los managers para obtener servicios de los *drivers*, generalmente de manera indirecta.

Managers

¿De manera indirecta?

- El board registra un driver y lo vincula a una etiqueta genérica de salida. Por ejemplo, la salida IO_BOARD_OUTB_LED_RGB_RED del driver IO_BOARD en la etiqueta OUTPUT_Bit_Warning del OUTPUT manager.
- La aplicación controla OUTPUT_Bit_Warning sin importar en qué sistema se ejecute.

embedul.ar

Listado de managers

INPUT

Input manager.

Gestiona las entradas del sistema (dispositivos IO).

Brinda a la aplicación información sobre eventos de press, click, doubleclick, hold, release para entradas digitales.

La aplicación puede obtener un valor guardado (buffered) hasta que el driver correspondiente se actualice, o uno inmediato, es decir, fuerza al driver a actualizarlo.

OUTPUT

Output manager.

Gestiona las salidas del sistema (dispositivos IO).

Brinda a la aplicación información sobre eventos de press, click, doubleclick, hold, release para entradas digitales.

Igual que Input, permite bufferear cambios en las salidas hasta que el driver mismo actualice (deferred) o efectivizarse al instante.

COMM

Communication manager.

Gestiona la comunicación del sistema (dispositivos STREAM y PACKET).

Al igual que INPUT y OUTPUT, el board registra y vincula los dispositivos de comunicación a una etiqueta.

Por ejemplo:

- COMM_Stream_Log
- COMM_Packet_RS485
- COMM_Packet_SPI_Local

STORAGE

Storage manager.

Gestiona el acceso a los dispositivos de almacenamiento RAWSTOR.

El board registra y vincula los dispositivos de almacenamiento asignándoles una etiqueta (rol). También puede “presentar” un dispositivo al manager y este determine el tipo (raw, fatfs) detectando el MBR y las particiones.

Además, el manager brinda un servicio de “cached elements”: archivos convertidos a almacenamiento RAW contiguo para acelerar la lectura.

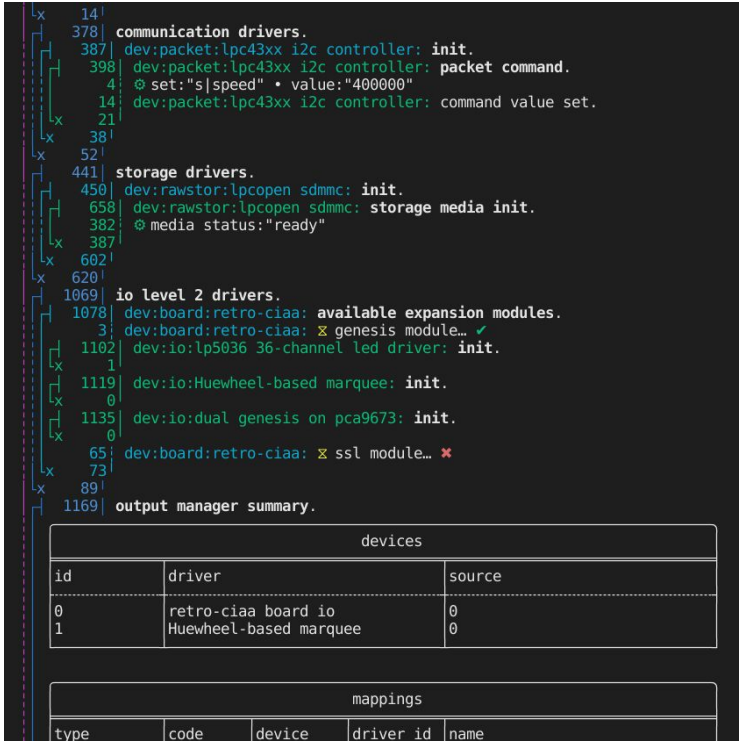
LOG

Logging manager.

Mensajes del sistema a través de un STREAM registrado por la implementación del objeto board.

Mensajes de log

Mensajes de log (cont'd)



embedul.ar

Programación defensiva

Programación defensiva

- Modo paranóico.
- Detecta errores y bugs causados por entradas inesperadas.
  Última línea de defensa, antes validación y testing!

```
33  void INPUT_Init (struct INPUT *const I)
34  {
35      BOARD_AssertState  (!s_i);
36      BOARD_AssertParams (I);
37
38      OBJECT_Clear (I);
39
40      LOG_ContextBegin (I, LANG_INIT);
41      {
```

```
72   * :param CodePoint: Unicode code point from the basic multilingual plan
73   */
74  uint8_t UTF8_SetCodePoint (uint8_t *const Data, const uint32_t Octets,
75                             const uint16_t CodePoint)
76  {
77      BOARD_AssertParams (Data && Octets);
78
79      // UTF-8 one octet, same as 7-bit ASCII.
80      if (CodePoint < 128)
81      {
```

embedul.ar

C11

C11

- Se utiliza ampliamente C11.
- Los autoparams tienen un costo
  _Generic, *_Static_assert*…
- Compila como C11, no como C++ estándar.
- Sin embargo, se puede invocar desde “C++”.

¿Por qué “C”?

- Indiscutido en bajo nivel (o sistemas con severas restricciones).
- Multiplataforma facil.
  Creado para portar sistemas operativos.
- Gente dedicada a bajo nivel lo entiende “bien”.
- Casi nadie usa C11.
- Se lleva mejor con bare metal, incluso sin *standard libraries*.
- Sus datos se importan o exportan fácil a cualquier otro lenguaje; muy fácil hacer bindings.

¿Y “C++”?

```
1  std::map<int,std::string> myMap;
2
3  if (auto [iter, succeeded] = myMap.insert(value); succeeded) {
4      useIter(iter);
```

Evalué que “C” era la herramienta ideal. Pero más allá de eso:

- Malentendidos sobre qué es o cómo aplicar “C++”.
  ¿Superset de “C”? Hace 40 años, tal vez. Hoy son lenguajes distintos.
  https://www.stroustrup.com/bs_faq.html#C-is-subset
- C++17 son jeroglíficos para programador “C”.

¿Y “C++”? (cont’d)

- Overpowered para esta aplicación.
- En embedul.ar no se usa Heap.
  ¿Qué hacer con el operador new?
- Escasa gente experta.
  tampoco abundan expertos en puro “C”, pero es manejable.

¿Preguntas?

¡Muchas gracias!

Auspiciantes

SAMCRO REMERAS