Guía de iniciación al kit de evaluación

St STM32F4 Discovery



Revisión: 2012/11/09

Este documento se ha escrito con permiso de Stmicroelectronics y se basa en la guías: UM1467: Getting started with software and firmware environments for the STM32F4DISCOVERY Kit

UM1472: STM32F4DISCOVERY STM32F4 high-performance discovery board



http://armcortexm.blogs.upv.es

Índice de contenido

1	Introducción3
2	Características3
3	El microcontrolador STM32F407VGT64
4	Requisitos y conexión del kit7
5	El depurador ST-LINK/V27
	5.1 Uso en Keil MDK-ARM8
	5.1.1 Instalación8
	5.1.2 Configurando las opciones de depuración
	5.1.3 Depuración más avanzada10
6	Uso del entorno de desarrollo MDK-ARM (Keil ™)10
	6.1 Versión demo de Keil10
	6.2 Construir un proyecto ya existente10
	6.3 Ejecutar y depurar un proyecto MDK-ARM11
	6.4 Crear un proyecto con el entorno MDK-ARM12
	6.4.1 Crear el proyecto12
	6.4.2 Configurar el proyecto13
	6.4.3 Añadiendo archivos fuente14
7	El "STM32F4DISCOVERY board firmware package16
	7.1 Carpeta libraries16
	7.1.1 Subcarpeta CMSIS16
	7.1.2 STM32F4xx_StdPeriph_Driver17
	7.2 Carpeta de proyecto17
	7.2.1 Subcarpeta demonstration17
	7.2.2 Subcarpeta Peripheral_Examples17
	7.3 Carpeta Utilities17
8	Esquema eléctrico19



1 Introducción

La placa *STM32F4DISCOVERY* permite descubrir las características del STM32F4 y desarrollar aplicaciones.

Esta guía pretende simplificar al máximo la tarea de toma de contacto e iniciación con este kit. Una vez iniciados, lo aconsejable es desechar esta guía y basarse en los documentos originales de St referenciados en la portada.

Suerte.

2 Características

La placa STM32F4DISCOVERY ofrece las siguientes características:

- 1 Microcontrolador *STM32F407VGT6* con 1 MB de memoria flash, 192 KB de RAM, encapsulado *LQFP100*.
- 2 *ST-LINK/V2* incorporado con selector usar el kit como un *ST-LINK/V2* independiente (con conector *SWD* para programación y depuración).
- 3 Fuente de alimentación: a través del bus USB o desde una fuente de alimentación externa de 5V.
- 4 Sensor de movimiento ST MEMS LIS302DL, acelerómetro con salida digital de 3 ejes
- 5 Sensor de audio ST MEMS *MP45DT02*, micrófono digital omnidireccional
- 6 Audio DAC CS43L22 con controlador integrado de altavoz clase D
- 7 Ocho LEDs:
 - LD1 (rojo / verde) para la comunicación USB
 - LD2 (rojo) alimentación 3,3 V
 - Cuatro LEDs de usuario, LD3 (naranja), LD4 (verde), LD5 (rojo) y LD6 (azul)
 - 2 LEDs USB OTG LD7 (verde), VBus y LD8 (rojo)
- 8 Dos pulsadores (usuario y reset)
- 9 USB OTG con conector micro-AB





Nota: El pin 1 de CN2, CN3, JP1, y de los conectores P1 y P2 están indicados mediante un cuadro.

Figura 3. STM32F4DISCOVERY Layout

3 El microcontrolador STM32F407VGT6

Este procesador ARM Cortex-M4 32-bit con FPU tiene 210 DMIPS, 1 MB Flash, 196 KB RAM, USB OTG HS/FS, Ethernet, 17 TIMs y 3 ADCs.



1 Mbyte de memoria Flash 192 Kbytes de RAM

STM32F407VGT6

Este dispositivo proporciona las siguientes características:

• 168 MHz/210 DMIPS Cortex-M4 con un solo ciclo DSP MAC y unidad de coma flotante:



Ejecución mejorada de algoritmos de control Facilidad de uso Mejor eficiencia de código Reducción del *"time to market"*

• Diseñado para un alto rendimiento y muy elevada transferencia de datos; Acelerador ART, 32-bit, AHB bus matriz de 7 capas con 7 maestros y 8 esclavos, incluyendo 2 bloques de SRAM, controladores DMA: 2 de propósito general, uno para USB HS, uno para Ethernet. Un bloque de SRAM dedicado al core.

 Eficiencia energética, ultra-bajo consumo de energía, *RTC* <1 μA en modo *VBAT*, de 3,6 V hasta 1,7 V *VDD*, el regulador de tensión con capacidad de potencia adaptable, proporcionando una mayor flexibilidad para reducir el consumo de energía en aplicaciones que requieren tanto mucho procesamiento y baja consumo funcionando en baja tensión o con baterías recargables.

• Integración máxima: Hasta 1 Mbyte de memoria Flash on-chip, 192 Kbytes de SRAM, circuito de reset, RCs interno, PLLs, encapsulado *WLCSP* disponible, etc.

• Los periféricos innovadores que ofrecen nuevas posibilidades para conectar y comunicar datos a alta velocidad, así como mayor precisión debido a su alta resolución.

•Extensas herramientas y soluciones software que proporcionan una amplia gama dentro de los ecosistemas *STM32* para desarrollar tus aplicaciones





Figura 6. Diagrama de bloques del STM32F407VGT6



4 Requisitos y conexión del kit

Para ejecutar y desarrollar aplicaciones en tu placa *STM32F4DISCOVERY*, los requisitos mínimos son:

- PC con Windows (2000, XP, Vista, 7).
- cable USB tipo A a Mini-B, utilizado para alimentar la placa (a través del conector USB CN1) desde el PC y conectarse al *ST-LINK/V2* para depuración y programación.

La placa se debe conectar al PC de la siguiente manera. Recuerda que es necesario instalar los controladores para poder utilizarla.



Figura x. Entorno hardware

5 El depurador ST-LINK/V2

Una parte de la placa *STM32F4DISCOVERY* es un depurador *ST-LINK/V2* que es soportada por diversos entornos de desarrollo.

Antes de poder emplear la placa con su depurador es necesario instalar los controladores. Comprobar que los jumpers estén de la siguiente manera.





Figura x. Jumpers CN3

5.1 Uso en Keil MDK-ARM

5.1.1 Instalación

Keil viene con el controlador ST-LINK/V2. Para instalarlo debemos localizar la instalación de Keil que, por defecto, es en el directorio *C:* *Keil* del disco duro local y acudir al directorio ARM\STLink\USBDriver. Ejecutar el driver y aceptar la instalación aunque salgan mensajes sospechosos.

A continuación, conectar la placa y comprobar que el asistente de controladores Windows encuentra y, aceptando la busqueda automática, instala los controladores.

5.1.2 Configurando las opciones de depuración

Cada proyecto Keil se configura individualmente, así que suele ser necesario reconfigurarlo manualmente cada vez que se mueva de ordenador.

Para ello, seguir estos pasos:

1. En el menú **Project**, selecciona **Options for Target 1** para mostrar el cuadro de diálogo **Target Options**.

Abre la pestaña *Debug*, haz clic en *Use* y selecciona *ST-Link Debugger*. A continuación, haz clic en *Settings* y selecciona el protocolo SWD. Clic en OK para guardar la configuración.
 Marca la opción "Run to main()".



	-
	×
Linker Debug) Utilities	Linker (Debug) Utilities
Use: RDI Interface Driver Settings	Use: ST-Link Debugger
ULINK Cortex Debugger RDI Interface Driver Altera Blaster Cortex Debugger Stellaris ICDI Initializatic Signum Sustems JTAGjet ST-Link Debugger Cortex-M/R J-LINK/J-Trace Uctex-M/R J-LINK/J-Trace Cortex-M/R J-LINK/J-Trace Cortex-M/R J-LINK/J-Trace Cortex-M/R J-LINK/J-Trace Cortex-M/R J-LINK/J-Trace Cortex-M/R J-LINK/J-Trace Cortex-M/R J-LINK/J-Trace Cortex-M/R J-LINK/J-Trace	Load Application at Startup Initialization File: STLink Setup Protocol JTAG SWD OK Cancel

Figura x. Cuadro de diálogo Target Options - Pestaña Debug

4. Abre la pestaña *Utilities*, selecciona *Use Target Driver for Flash Programming* y elige *ST-Link Debugger* en el desplegable.

5. Verifica que la opción *Update Target before Debugging option* está seleccionada.

6. Haga clic en *OK* para guardar la configuración.

gure Flash Menu Command	
Jse Target Driver for Flash Programming	
ULINK Cortex Debugger Settings ULINK Cortex Debugger	Target before Debugging
Init File: RDI Interface Driver	Edit
Jse Extern Signum Systems JTAGjet	
Cortex-M/H J-LINK/J-Trace	
rouments:	
	gure Flash Menu Command Jse Target Driver for Flash Programming ULINK Cortex Debugger Init File: ULINK Cortex Debugger RDI Interface Driver Stellaris ICDI Signum Systems JTAGjet Contex:M7H J-LINK/J-Trace guments: Run Independent

Figura x. Cuadro de diálogo Target Options – Pestaña Utilities



5.1.3 Depuración más avanzada

Esto está aún verde, pero es importante y sustituirá a lo anterior un día de estos.

La guía provisional (en inglés) está en http://armcortexm.blogs.upv.es/stm32f4-discovery-and-printf-redirection-to-debug-viewer-in-keil/

6 Uso del entorno de desarrollo MDK-ARM (Keil ™)

6.1 Versión demo de Keil

En www.keil.com se puede descargar una versión limitada de MDK-ARM previo registro. Una limitaciones se basa en permitir un tamaño de código máximo de 32K, lo cual es suficiente probar todos los ejemplos.

6.2 Construir un proyecto ya existente

1. Abrir el MDK-ARM µVision4 IDE,

La figura 3 muestra los nombres de las ventanas referenciadas en este apartado.



Figura 3. Entorno MDK-ARM µVision4 IDE

2. En el menú "**Project**", seleccione abrir proyecto (*Open Project* …) para mostrar el cuadro de diálogo de selección de fichero de proyecto. Seleccionar el archivo de proyecto *STM32F4-Discovery.uvproj* y haga clic en abrir (*Open*).

3. En el menú *Project*, selecciona *Rebuild all target files* para compilar el proyecto.



4. Si tu proyecto se ha compilado con éxito, aparecerá en pantalla la siguiente ventana *Build Output* (figura 4):



Figura 4. Build Output - MDK-ARM µVision4. Proyecto compilado con éxito.

6.3 Ejecutar y depurar un proyecto MDK-ARM

(Aço hi haurà que amliar-ho a clase)

En el entorno *MDK-ARM* μ *Vision4 IDE*, haga clic en la lupa para programar la memoria Flash e iniciar la depuración, como se muestra a continuación en la figura 5.



Figura 5. Inicio de una sesión de depuración en MDK-ARM µVision4

El depurador en el *MDK-ARM IDE* se puede utilizar para depurar el código fuente en *C*, establecer puntos de ruptura (*breakpoints*), monitorizar variables individuales y observar eventos durante la ejecución del código, como se muestra en la figura 6.



Ele Edit View Project Fligh Debug Pergherals Extension A Extension B Iools 2VCS Window Help File Edit View Project Fligh Debug Pergherals Extension A Extension B Iools 2VCS Window Help Registers • 0 × Registers • 0 × Registers • 0 × Conservation 57: (Signed Value 57: (Base Debug Pergherals Extension A Extension B Iools 2VCS Window Help Project • 0 × Registers • 0 × Project • 0 ×	DISCOVER - µVision4	
Registers	<u>File Edit View Project Flash</u>	Debug Peripherals Extension A Extension B Tools SVCS Window Help
Register Value Project Value P1 0.00000000 P1 P1 0.00000000 P1 P2 0.00000000 P3 P3 0.00000000 P3 P4 0.00000000 P3 P4 0.000000000 P3 P4 0.00000000 P3 P4 0.00000000 P4	🗋 🗃 🗑 🖉 🛛 🖓 🖄 🖎 🗠	○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○
Register Value Dessembly Image: Control of the second secon	8 E O O O O	> D 🖲 🖕 🖉 • 🔳 • 📝 • 🔜 • 🔛 • 🔜 • 🏹 • 🔜 •
Register Value 57: { Current 0x00000438 55: /* Enable PWR and GPIOx Clocks */ R0 0x00000438 55: RCC_APBIPeriphClockCmd(RCC_APBIPeriph_PWR, ENABLE); R2 0x0000000 55: RCC_APBIPeriphClockCmd(RCC_APBIPeriph_PWR, ENABLE); R3 0x0000000 0x08000462 0708 LSLS r0,r1,#28 0x08000462 0708 LSLS r0,r1,#28 0x08000462 0708 LSLS r0,r1,#28 0x08000000 R6 0x0000000 RCC_APBIPeriphClockCmd (0x08000 R6 0x0000000 R8 0x0000000 r1 #0x01 R10 0x0000000 R11 0x0000000 r1 #0x01 Pescription : Main program. R11 0x0000000 R12 0x0000000 r65 r Erbiton : None R13(FP) 0x0000000 r Erbiton : None r Erbiton : None PCC_APBIPeriphClockCmd(RCC_APBIPeriph_PWR, ENABLE); Project Registers /* Erbit PWR and GPIOx Clocks */ RCC_APBIPeriph_PWR, ENABLE); Project Registers /* Erbit PeriphClockCmd (BCC_APBIPeriph_PWR, ENABLE); PCC V200000000 0x0000000 0x0000000	Registers 🗸 🗸 🗸	X Disassembly
S8: /* Enable PWR and GPIOx Clocks */ R0 0x08000488 B570 PUSH (r4-r6,1r) R1 0x20000438 B570 PUSH (r4-r6,1r) S9: RCC_APB1PeriphClockCmd(RCC_APB1Periph_PWR, ENABLE); 0x08000482 PUSH RCC_APB1PeriphClockCmd(RCC_APB2Periph_GPIOx, ENABLE); 0x08000482 PUSH RCC_APB1PeriphClockCmd(RCC_APB2Periph_GPIOx, ENABLE); 0x080000000 RCC_APB1PeriphClockCmd(RCC_APB2Periph_GPIOx, ENABLE); No08000000 R6 0x0000000 RCC_APB1PeriphClockCmd(RCC_APB2Periph_GPIOx, ENABLE); R7 0x0000000 RCC_APB1PeriphClockCmd(RCC_APB2Periph_GPIOx, ENABLE); R10 0x0000000 RCC_APB1PeriphClockCmd(RCC_APB2Periph_GPIOx, ENABLE); R11 0x0000000 RCC_APB1PeriphClockCmd(RCC_APB1Periph_PWR, ENABLE); R12 0x00000000 S5 R13(SP) 0x0000000 R14(LR) 0x0000000 R15(PC) 0x0000000 S6 /* Enable PWR and GPIOx Clocks */ R13(SP) 0x0000000 S7 /* Enable PWR and GPIOx Clocks */ R20000000 0x0000000 0x0000000 S8 0x0000000 <td>Begister Value</td> <td>57: (</td>	Begister Value	57: (
R0 0.0000000 R1 0.0000000 R2 0.0000000 R3 0.0000000 R3 0.0000000 R4 0.00000000 R3 0.00000000 R4 0.00000000 R5 0.00000000 R6 0.00000000 R7 0.00000000 R8 0.00000000 R10 0.0000000 R11 0.0000000 R12 0.0000000 R13 (SP) 0.2000048 051 * Description * Main proget 056 int main(void) * 10 0.0000000 R12 0.0000000 054 * Enable PWR and GPIOx Clocks */ 057 0 058 N2000000 059 P/* Enable PWR and GPIOx Clocks */ 059 RCC_APBIPeriphClockCmd(RCC_APBIPeriph PWR, ENABLE); 050 Int main(void) * 0.2000000 0.0000000 050 Int main(void) * 0.20000000 0.00000000 020000	E Current	58: /* Enable PWR and GPIOx Clocks */
S9: RCC_APB1PeriphClockCmd (RCC_APB1Periph_PWR, ENABLE); R2 0x00000000 R3 0x0800047b R4 0x08000482 R5 0x08000482 R6 0x08000000 R7 0x0000000 R8 0x0000000 R7 0x0000000 R8 0x0000000 R9 0x0000000 R10 0x0000000 R11 0x0000000 R12 0x0000000 R13 0x0000000 R14 0x0000000 R13 0x0000000 R14 0x0000000 R13 0x0000000 R14 0x0000000 R13 0x0000000 R14 0x0000000 S9: RCC_APB1PeriphClockCmd (RCC_APB1Periph_PWR, ENABLE); S9: RCC_APB1PeriphClockCmd (RCC_APB1Periph_PW		□ C>0x08000488 B570 PUSH (r4-r6, lr)
R2 0x0000000 R3 0x0800048 R4 0x0800048 R5 0x0800048 R6 0x0800048 R7 0x0800048 R8 0x0800000 R8 0x0800000 R4 0x0800000 R7 0x0800000 R8 0x0000000 R9 0x0000000 R11 0x0000000 R12 0x0000000 R13 0x0000000 R14 0x0000000 R13 0x0000000 R14 0x0000000 R13 0x0000000 R4 0x0000000 R3 0x0000000 R4 0x0000000 R13 0x0000000 R4 0x0000000 R5 0x1000000 R5 0x1000000 R5 0x1000000 R14 0x0000000 R5 0x1000000 R5 0x1000000 R5 0x1000000 R5 0x1000000 R6 0x10	B1 0v20000438	59: RCC_APB1PeriphClockCmd(RCC_APB1Periph_PWR, ENABLE)
R5 0x00000047b R4 0x00000047b 0x00000048E F7FFFE1 BL.SLS r0,r1,#28 R5 0x0000000 R6 0x0000000 0x00000000 R5 0x0000000 0x00000000 R5 0x0000000 0x0000000 R6 0x00000000 R7 0x0000000 0x0000000 0x0000000 R8 0x0000000 051 0x0000000 052 140x75 r140x01 2 R7 0x0000000 R10 0x0000000 052 1main.c R11 0x0000000 R12 0x0000000 053 0utput : None 052 1mput : None 055 int main(void) 055 0x02000000 055 057 (Description 1main (void)	B2 0x0000000	0x0800048A 2101 MOVS r1,#0x01
R4 0x00000000000000000000000000000000000	B3 0x0800047b	0x0800048C 0708 LSLS r0,r1,#28
ASS 0x00000000 R6 60: RCC_APB1PeriphClockCmd(RCC_APB2Periph_GPIOC, ENABLE py08000402 2101 MONS r1 flow01 R6 0x00000000 R7 0x00000000 R8 0x00000000 00000000 main.c Imput r1 flow01 R8 0x00000000 R10 0x00000000 R11 0x00000000 053 Imput r1 flow01 R12 0x00000000 R13(SP) 0x20000438 R14(LR) 0x0000000 055 Imput r1 flow01 R12 0x00000000 R13(SP) 0x20000438 R14(LR) 0x0000000 055 Imput r1 flow01 R15(FP) 0x00000000 055 Imput r1 flow01 flow02 flow02 R14(LR) 0x00000000 0x10000000 flow02 flow02 flow02 flow02 R15(FP) 0x00000000 0x0000000 0x0000000 flow02 flow02 </td <td>B4 0x0800063c</td> <td>0x0800048E F7FFFEE1 BL.W RCC_APB1PeriphClockCmd (0x0800</td>	B4 0x0800063c	0x0800048E F7FFFEE1 BL.W RCC_APB1PeriphClockCmd (0x0800
R6 0x0000000 R7 0x0000000 R8 0x0000000 R9 0x0000000 R10 0x0000000 R11 0x0000000 R12 0x0000000 R13 (SP) 0x2000000 R14 (LR) 0x0000000 R15 (PC) 0x0000000 R14 (LR) 0x0000000 R15 (PC) 0x0000000 Note 056 R15 (PC) 0x0000000 Note 057 Ø58 /* Enable PWR and GPIOx Clocks */ RCC_APBIPeriphClockCond (RCC_APBIPeriph_PWR, ENABLE); DCC_ADBIDeriphClockCond (RCC_APBIPeriph_PWR, ENABLE); DCC_ADBIDeriphClockCond (RCC_APBIPeriph_PWR, ENABLE); DCC_ADBIDeriphClockCond (RCC_ADBIDeriph GDIOC_ENBERDeriph GDIOC_ENBERDeri	B5 0x0800063c	60: RCC_APB1PeriphClockCmd(RCC_APB2Periph_GPIOC, ENABL
R7 0x0000000 R8 0x0000000 R9 0x0000000 R10 0x0000000 R11 0x0000000 R12 0x0000000 R13(SP) 0x20000438 R14(LR) 0x08000175 R15(PC) 0x080001486 POST (* Enable PWR and GPIOx Clocks */ Project Registers Project Registers V20000000: 00000000 00000000 00000000 0x20000000: 00000000 00000000 00000000 00000000 0x20000000: 00000000 00000000 00000000 00000000 00000000 0x200000000: 00000000 <td>B6 0x0000000</td> <td>□ 0v08000402 2101 ₩0375 v1 #0v01</td>	B6 0x0000000	□ 0v08000402 2101 ₩0375 v1 #0v01
R8 0x0000000 R9 0x0000000 R10 0x0000000 R11 0x0000000 R12 0x0000000 R13 0x0000000 R14 0x0000000 R15 0x0000000 R12 0x0000000 R13 0x0000000 R14 0x0000000 R15 0x0000000 R14 0x0000000 R13 0x0000000 R14 0x0000000 R13 0x0000000 R14 0x0000000 R15 0x0000000 R	B7 0x0000000	
R8 0x0000000 R10 0x0000000 R11 0x0000000 R12 0x0000000 R13 (SP) 0x2000438 R14 (LR) 0x03000175 R15 (PC) 0x03000175 R15 (PC) 0x03000175 R15 (PC) 0x03000185 055 int main (void) 056 int main (void) 057 /* Enable PWR and GPIOx Clocks */ RCC_APB1PeriphClockCmd (RCC_APB1Periph PWR, ENABLE); Project Registers * PCC_APB1PeriphClockCmd (RCC_APB1Periph PWR, ENABLE); Project Registers * PCC_APB1PeriphClockCmd (RCC_APB1Periph PWR, ENABLE); Project Registers * PCC_APB1PeriphClockCmd (RCC_APB1Periph PWR, ENABLE); Project Registers * PCC_APB1Periph Ox000000 0x20000000: 00000000 00000000 0x20000000: 00000000 00000000 00000000 0x20000000: 00000000 00000000 00000000 00000000 0x20000000: 00000000 000000000 00000000 <td< td=""><td>B8 0x0000000</td><td>amain.c</td></td<>	B8 0x0000000	amain.c
All Description : Main program. All Output <td: none<="" td=""> All 0x00000000 R11 0x00000000 R12 0x00000000 R13(SP) 0x20000438 R14(LR) 0x0000000 R15(PC) 0x0000000 R15(PD) 0x0000000 R15(PD) 0x0000000 R15(PC) 0x0000000 RCC APBIPeriphClockCond(RCC_APB1Periph PWR, ENABLE); RCC APB1PeriphClockCond(RCC_APB1Periph PWR, ENABLE); Project Registers Memory 1 Image: Registers Address: 0x20000000 0x20000000: 00000000 00000000 0x20000000: 00000000 00000000 0x20000000: 00000000 00000000 00000000 0x200000000: 00000000 <t< td=""><td>B9 0x0000000</td><td></td></t<></td:>	B9 0x0000000	
R11 0x0000000 R12 0x0000000 R13 0x0000000 R13 0x0000000 R13 0x0000000 R13 0x0000000 R13 0x0000000 R14 0x0000000 R15 RCC Address: 0x20000000 0x200000000 00000000 00000000 00000000 0x200000000 00000000 00000000 00000000 00000000 0x2000000000 00000000 00000000 00000000 00000000 00000000 0x20000000000 00000000 00000000 <td< td=""><td>B10 0x00000000</td><td>051 * Description : Main program.</td></td<>	B10 0x00000000	051 * Description : Main program.
R12 0x0000000 R13 0x20000438 R14 0x20000438 R15 0x20000438 R14 0x20000438 R14 0x20000438 R14 0x20000438 R15 0x20000438 R14 0x200000438 R15 0x20000000 R16 0x20000000 R17 RCC_APB1PeriphClockCmd (RCC_APB1Periph_PWR, ENABLE); R17 RCC_APB1PeriphClockCmd (RCC_APB1Periph_PWR, ENABLE); R17 RCC_APB1PeriphClockCmd (RCC_APB1Periph_GD1000000000000000000000000000000000000	B11 0x00000000	052 * Input : None
R13 (SP) 0x20000438 R14 (LR) 0x40000438 R14 (LR) 0x40000438 R15 (PC) 0x00000438 Internal 056 Improject Registers Improject Registers Improject 00000000 000000000 00000000 000000000 00000000 000000000 00000000 000000000 00000000 000000000 00000000 000000000 00000000 000000000 00000000 000000000 00000000 000000000 00000000 000000000 00000000 000000000 00000000 000000000 00000000 000000000 00000000 000000000 00000000 000000000 00000000 000000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 000000000 00000000 000000000 00000000 000000000 000000000 </td <td></td> <td>053 * Output : None</td>		053 * Output : None
Ald LLA 0x08000175 Al5 (PC) 0x08000175 (0x08000488) WPPSR 0x61000000 Internal 055 Project Registers Memory 1 Image: Constraint of the second sec		054 * Keturn : None
R15 (PC) 0x00000488 WSSR 0x51000000 Internal 058 Project Registers Memory 1 Image: Control of the control of	R14 (LR) 0x08000175	055 int main (maid)
Internal Ox61000000 O/* Enable PWR and GPIOx Clocks */ RCC_APB1PeriphClockCmd (RCC_APB1Periph_PWR, ENABLE); DCC_ADB1DerinhClockCmd (RCC_APB1Periph_PWR, ENABLE); Improject Registers DCC_ADB1DerinhClockCmd (RCC_APB1Periph_PWR, ENABLE); Address: Dx20000000 000000000 000000000 0000000	R15 (PC) 0x08000488	
Internal 059 RCC_APBIP eriph ClockCond (RCC_APBIPeriph_PWR, ENABLE); RCC_APBIP eriph ClockCond (RCC_APBIPeriph_GDTOC_ENBRDE); RCC_APBIP eriph ClockCond (RCC_APBIPeriph_GDTOC_ENBRDE); Wemory 1 Address: 0x20000000 00000000 00000000 00000000 0x20000000: 00000000 00000000 00000000 00000000 00000000 0x20000000: 00000000 00000000 00000000 00000000 00000000 00000000 0x20000000: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 000000000 00000000 <td< td=""><td>E XPSR 0x61000000</td><td>059 (f Enchie DET and CDTOR Cleaks f/</td></td<>	E XPSR 0x61000000	059 (f Enchie DET and CDTOR Cleaks f/
Improject Registers Registers <t< td=""><td>Internal</td><td>059 RCC APBIPerinhClockCmd(RCC APBIPerinh PWR, ENABLE):</td></t<>	Internal	059 RCC APBIPerinhClockCmd(RCC APBIPerinh PWR, ENABLE):
Improjet		NCC_ADB1DeriphClockCmd(DCC_ADB2Deriph_CDIOC_ENABLE);
Address: 0x200000000 00000000	E Project Registers	
Address: 0x20000000 00000000	femory 1	1 🔻
Dx20000000: 00000000 00000000 00000000 000000	Address: 0x20000000	
xx20000020: 04000000 00000000		
x2000026: 00000000 00000000 000000	20000000: 00000000 000	
xx2000040: 000000000 00000000 000000000	x20000020: 04030201 090	50706 40010600 40011000 40011000 0000000 00000000 00000000
x20000080: 000000000 00000000 000000000	20000040: 00000000 000	
X20000020: 000000000 00000000 000000000		
X20000020: 00000000		
Dx200000E0: 000000000 000000000 000000	x200000x0; 00000000 000	
Cal Stack Alocals Alocals Alocals Alocal and	X20000000; 00000000 000	
Call Stack Stack Stack Stack Stack Stack Symbols		
	Call Stack Locals Watch 1	Memory 1 JS Symbols

Figura 6. Espacio de trabajo del IDE MDK-ARM

6.4 Crear un proyecto con el entorno MDK-ARM

6.4.1 Crear el proyecto

1. En el menú *Project*, selecciona *New μVision Project*... para visualizar el cuadro de diálogo de creación de proyecto *Create Project File*. Nombra el nuevo proyecto y haz clic en guardar (*Save*).

Project	Fl <u>a</u> sh <u>D</u> ebug Pe <u>r</u> ipherals
Ne	w µ⊻ision Project…
Net	w Multi-Project <u>W</u> orkspace
Op	en Project
⊆lo	ise Project
Ext	port 🕨
Ma	inage 🕨 🕨

Figura 7. Creación de un nuevo proyecto

2. Cuando se guarda un nuevo proyecto, el *IDE* muestra el cuadro de diálogo de selección de dispositivos, donde seleccionaremos nuestro dispositivo. En este caso vamos a utilizar el *STMicroelectronics* montado en la placa *STM32F4DISCOVERY*, con lo que habrá que hacer



doble clic en *STMicroelectronics*, seleccionar el dispositivo *STM32F407VGT6* y hacer clic en *OK* para guardar la configuración.

Vendor: STMicroelectronics
Device: STM32F407VGT6
Toolset: ARM
STMicroelectronics * Core: ARM 32-bit CortexM4 CPU with Adaptive real-time (ART) accelerator allowing

Figura 8. Cuadro de diálogo selección de dispositivo

3. Haz clic en el botón "**Yes**" para copiar el **STM32 Startup Code** a la carpeta del proyecto y agregar el archivo al proyecto como se muestra en la figura 9.

 · · ·
μVision
Copy STM32 Startup Code to Project Folder and Add File to Project ?
Ves No

Figura 9. Cuadro de diálogo STM32 Startup Code

Nota: El fichero por defecto del *STM32 Startup Code* incluye la función *SystemInit*. Se puede comentar este fichero para que no se utilice o añadir el archivo *system_stm32f4xx.c* desde la biblioteca*STM32f4xx firmware library*.

6.4.2 Configurar el proyecto

1. En el menú **Project**, selecciona **Options for Target 1** para mostrar el cuadro de diálogo **Target Options**.

2. Abre el *Target tab* e introduce la configuración de inicio (*start*) y tamaño (*size*) de *IROM1* e *IARM1* como se muestra en la figura 13.



				Stereor -	er andre der				
51 Microel	ectionacs 5	1M32L152HB	-		Code 0	ieneration			
		3	Ral (MHz): 8	10					
Operatin	g system:	None		-	ΓU	se Cross-Mo	dule Optimiza	lion	
					(PU	se MicroLIB	> r	Big Endian	
Read/0	inly Memor	y Areas		ľ	-Read/	write Memo	ıy Areas		
default	off-chip	Start	Size	Startup	default	off-chip	Start	Size	Nolni
Г	ROM1:			C	Г	RAM1:			Г
E	ROM2			C	Г	RAM2			Г
Г	ROM3		1		Г	RAM3			- г
-	-on-chip				1.00	on-chip			
	IBOMI	0.0000000	0x20000		(F	IRAMT)	\$20000000	0x4000	Sr
5	HIGHING &					F			-

Figura 13. Cuadro de diálogo Target Options - Pestaña Target

3. Siguiendo las instrucciones del apartado ST-Link/V", configura las opciones de depouración.

6.4.3 Añadiendo archivos fuente

-

Para crear un nuevo fichero fuente, selecciona *New* en el menu *File*, y se abrirá una ventana de edición vacía donde introducir el código fuente.

El entorno *MDK-ARM* resalta en color la sintaxis de C cuando se guarda el archivo con *(File> Save As...)* extensión *.c. En el ejemplo de la figura 10, el archivo se guarda como main.c.

i main.c
1
3 int main (void)
$4 \vdash (\\ 5 \mid \mathbf{return}(0); $
6 L)

Figura 10. Ejemplo de archivo *main.c*

El entorno *MDK-ARM* ofrece varias maneras de agregar ficheros fuente a un proyecto. Por ejemplo, se puede seleccionar el grupo de archivos en **Project Window > Files** y haciendo clic con el botón derecho del ratón para abrir el menú contextual. Selecciona la opción **Add Files...**, y selecciona finalmente el archivo main.c creado previamente.



Project	→ ↓ ×	
	Options for Group 'Source Group 1' Alt+I	+F7
	Open File	
	Open List File	
	Open <u>M</u> ap File	
2-22-2- 1-1-1-1	<u>R</u> ebuild all target files	
pages Internet	Build target	F7
	Tr <u>a</u> nslate File	
	Stop b <u>u</u> ild	
	Add Group	
	Add Files to Group 'Source Group 1'	
	Remo $\underline{v}e$ Group 'Source Group 1' and its Files	
*	Manage <u>C</u> omponents	
~	Show Include File Dependencies	

Figura 11. Añadir ficheros fuente

Si el archivo se agrega correctamente, se muestra la siguiente ventana (figura 12).

Project 🕑 🗙	
Target 1 Source Group 1 Startup_stm32f4xx	

Figura 12. Estructura nuevo proyecto

- 8. En el menú Project, selecciona Build Target.
- 9. Si el proyecto se compila con éxito, se abrirá la siguiente ventana (figura 16):



Figura 16. Proyecto compilado con éxito en MDK-ARM µVision4

- 10. Antes de ejecutar la aplicación, conecta la placa *STM32F4DISCOVERY* como se describe en la sección 1.
- 11. Para programar la memoria Flash y empezar a depurar, sigue las instrucciones que figuran en la sección 4.2: "depurar y ejecutar tu proyecto".



7 EI "STM32F4DISCOVERY board firmware package

St proporciona un paquete de software para el kit que debe descargarse de su web. Dicho paquete incorpora las bibliotecas CMSIS, las bibliotecas específicas de periféricos y un conjunto de ejemplos que permiten hacerse una idea de las capacidades del microcontrolador. El paquete viene en comprimido y, al descomprimir, el archivo zip se crea una carpeta, *STM32F4-Discovery_FW_VX.YZ*, que contiene las siguientes subcarpetas:



Figura x. Estructura directorios

7.1 Carpeta libraries

Esta carpeta contiene la capa de abstracción del hardware (HAL) para los dispositivos *STM32F4xx*.

7.1.1 Subcarpeta CMSIS

Esta subcarpeta contiene los archivos CMSIS STM32F4xx y Cortex-M4F.

Los archivos CMSIS Cortex-M4F consisten en:

<u>- Core Peripheral Access Layer</u>: que contiene las definiciones de nombres, direcciones y funciones de ayuda para acceder a los registros y periféricos Cortex-M4F. Además, define una interfaz independiente del dispositivo para *kernels RTOS* que incluye definiciones de canales de depuración.

<u>- CMSIS DSP Software Library</u>: cuenta con una suite de funciones de procesamiento de señal para su uso en dispositivos basados en el procesador *Cortex-M*. La biblioteca está completamente escrita en C y es totalmente compatible con *CMSIS*.

Los archivos STM32F4xx CMS/S consisten en:

<u>- stm32f4xx.h:</u> este archivo contiene las definiciones de todos los registros de periféricos, bits, y mapeo de memoria para dispositivos *STM32F4xx*. Este archivo es el único *include* utilizado en el código fuente de la aplicación, por ejemplo, *main.c.*



<u>- system_stm32f4xx.c/.h:</u> Este archivo contiene la configuración del reloj para los dispositivos *STM32F4xx*. Exporta la función *SystemInit()* que configura el reloj, los factores multiplicadores y divisores PLL, AHB / APBx *prescalers* y *Flash*. Esta función es invocada en la inicialización justo después del reset y antes de comenzar el programa principal. La llamada se realiza dentro del archivo *startup_stm32f4xx.s*.

<u>- startup_stm32f4xx.s:</u> Proporciona el código de inicialización del *Cortex-M4F* y los vectores de interrupción para todos los manejadores de interrupción del dispositivo *STM32F4xx*.

7.1.2 STM32F4xx_StdPeriph_Driver

Esta subcarpeta contiene los fuentes de los controladores de periféricos del *STM32F4xx* (excepto USB y Ethernet).

Cada *driver* consiste en un conjunto de rutinas y estructuras de datos que cubren (casi) todas las funcionalidades de los periféricos. El desarrollo de cada *driver* está guiado por una API común, que estandariza la estructura del *driver*, las funciones y los nombres de los parámetros.

Cada periférico tiene un archivo de código fuente, *stm32f4xx_ppp.c*, y un archivo de cabecera, *stm32f4xx_ppp.h*. El archivo *stm32f4xx_ppp.c* contiene todas las funciones firmware necesarias para utilizar el periférico "PPP".

7.2 Carpeta de proyecto

Esta carpeta contiene los fuentes de las aplicaciones STM32F4DISCOVERY.

7.2.1 Subcarpeta demonstration

Esta subcarpeta contiene los fuentes de una aplicación demo con el proyecto preconfigurado para los entornos IAR *EWARM, Keil MDK-ARM*, Atollic *TrueSTUDIO y TASKING*.

En la subcarpeta *Binary* se proporcionan los binarios (*.hex y *.dfu) de esta aplicación. Pudiéndose utilizar el *STM32F4xx Bootloader* o cualquier herramienta de programación para reprogramar la aplicación demo usando estos ficheros.

7.2.2 Subcarpeta Peripheral_Examples

Esta subcarpeta contiene ejemplos para algunos periféricos con proyectos preconfigurados para los entornos *EWARM, MDK-ARM, TrueSTUDIO y TASKING*.

7.3 Carpeta Utilities

Esta carpeta contiene una capa de abstracción específica para el *STM32F4DISCOVERY*. La idea de esta capa es prtoporcionar servicio para manejar los dispositivos disponibles. Proporciona los siguientes controladores:

- *stm32f4_discovery.c*: proporciona funciones para manejar el pulsador de usuario y 4 LEDs (*LD3.LD6*)



- *stm32f4_discovery_audio_codec.c/.h*: proporciona funciones para manejar el audio DAC (CS43L22)

- *stm32f4_discovery_lis302dl.c/.h*: proporciona funciones para manejar el acelerómetro *MEMS* (LIS302DL).



8 Esquema eléctrico

Figura x: STM32F4DISCOVERY











http://armcortexm.blogs.upv.es

Figura 13. MCU





Figura 14: Audio











Figura 16: Periféricos



