

# EE3220 System-on-Chip Design

## Tutorial Note 3

### ARM-based 32-bit MCUs Debug & UART

# Objective

- In this tutorial, you will be able to learn:
  - The basic feature and design tools of ARM Cortex-M architecture.
  - Debug system architecture and experiment with Keil Studio.
  - Serial communication bus– UART and USART.

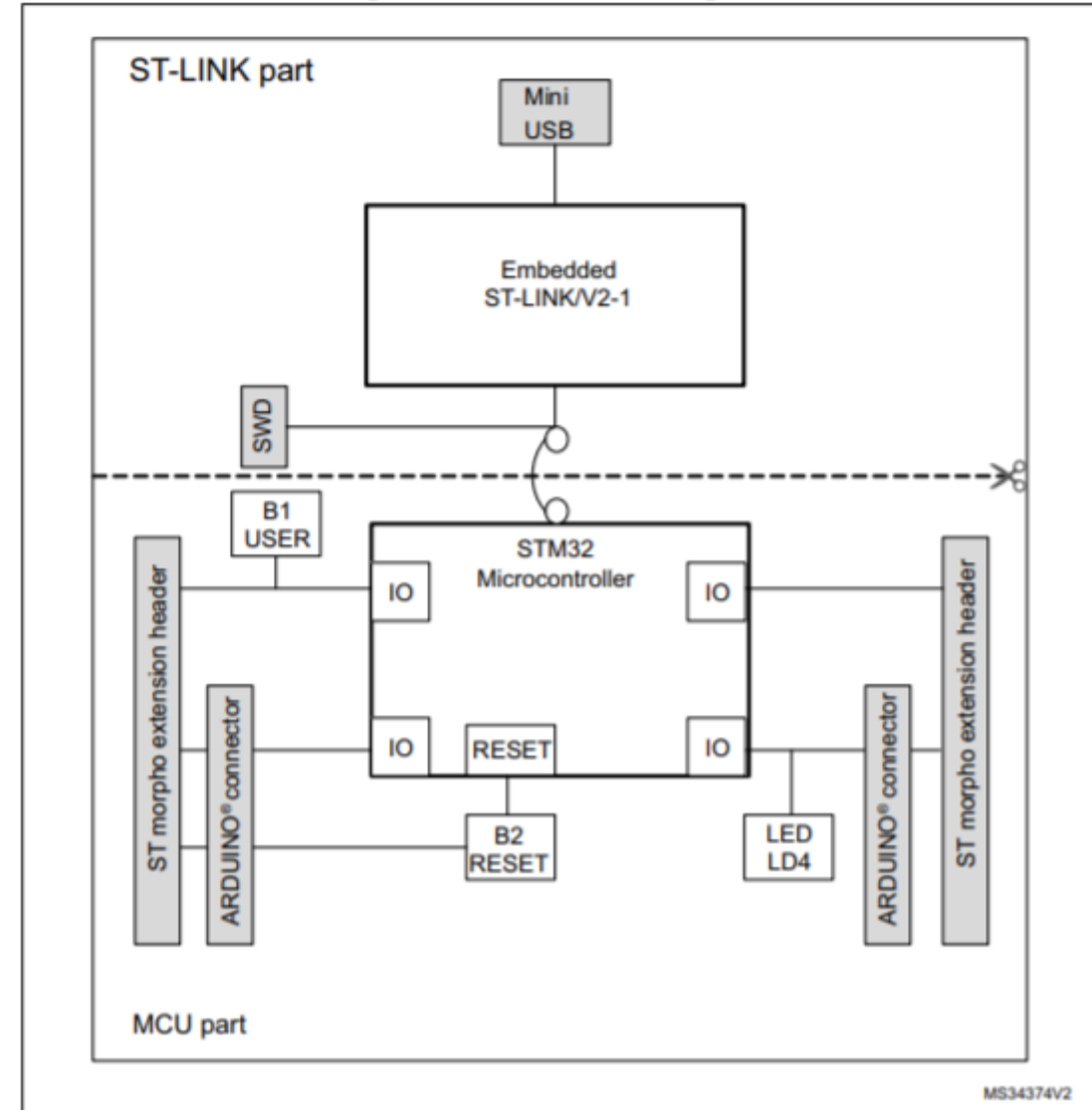
# Overview of NUCLEO F401-RE

## Common features

- STM32 microcontroller in LQFP64 package
- 1 user LED shared with ARDUINO®
- 1 user and 1 reset push-buttons
- 32.768 kHz crystal oscillator
- Board connectors:
  - ARDUINO® Uno V3 expansion connector
  - ST morpho extension pin headers for full access to all STM32 I/Os
- Flexible power-supply options: ST-LINK, USB VBUS, or external sources
- On-board ST-LINK debugger/programmer with USB re-enumeration capability: mass storage, Virtual COM port and debug port
- Comprehensive free software libraries and examples available with the STM32Cube MCU Package
- Support of a wide choice of Integrated Development Environments (IDEs) including IAR Embedded Workbench®, MDK-ARM, and STM32CubeIDE

## Board-specific features

- External SMPS to generate Vcore logic supply
- 24 MHz HSE
- Board connectors:
  - External SMPS experimentation dedicated connector
  - Micro-AB or Mini-AB USB connector for the ST-LINK
  - MIPI® debug connector
- Arm® Mbed Enabled™ compliant



# Development tools

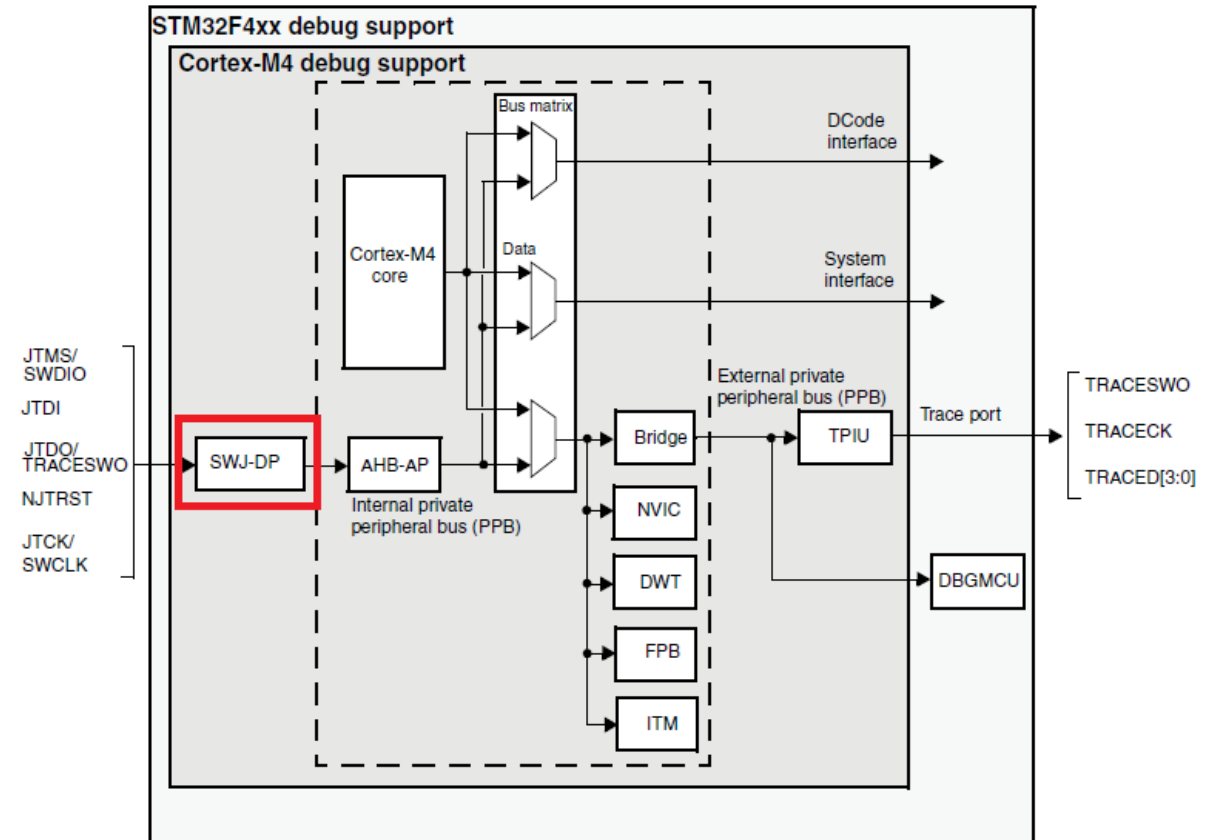
- The board is supported by many tools and IDEs many vendors
  - ARM Development Studio (licensed)
  - ARM® Mbed online
  - ARM Keil Studio Cloud
  - STM32CubeIDE (free)
  - IAR Systems - IAR Embedded Workbench® ( licensed)
  - Keil®: MDK-ARM(a) (licensed)
  - GCC-based IDEs
  - CrossWorks
  - SEGGER Embedded Studio (commercial with free-trial)
  - SW4STM32 (free)
  - TrueSTUDIO (free)
  - MATLAB® and Simulink® (licensed)

# Debug system overview

- The STM32F401RE is built around a Cortex®-M4 with FPU core which contains hardware extensions for advanced debugging features:
  - The debug extensions allow the core to be stopped either on a given instruction fetch (breakpoint) or data access (watchpoint).
  - When stopped, the core's internal state and the system's external state may be examined.
  - Once examination is complete, the core and the system may be restored and program execution resumed.
- Two interfaces for debug are available:
  - Serial wire
  - JTAG debug port

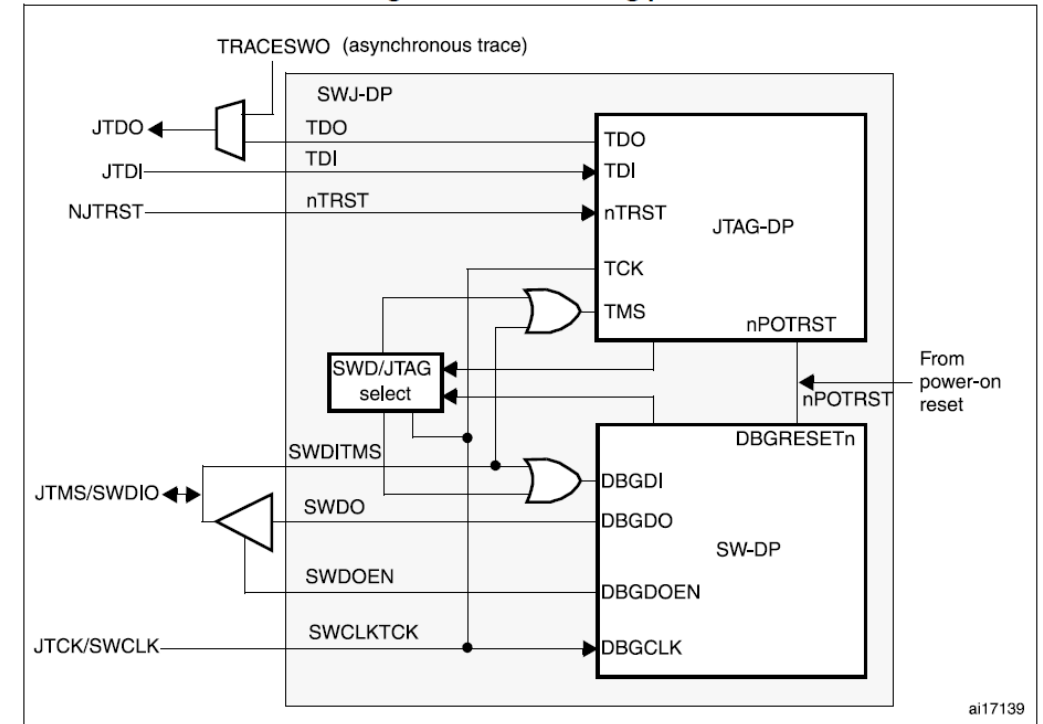
# STM32 MCU and Cortex®-M4 debug block

- The Arm® Cortex®-M4 with FPU core provides integrated on-chip debug support. It is comprised of:
  - SWJ-DP: Serial wire / JTAG debug port
  - AHB-AP: AHB access port
  - ITM: Instrumentation trace microcell
  - FPB: Flash patch breakpoint
  - DWT: Data watchpoint trigger
  - TPUI: Trace port unit interface
  - ETM: Embedded Trace Macrocell



# SWJ debug port

- The core of the STM32F401RE integrates the Serial Wire / JTAG Debug Port (SWJ-DP). It is an Arm® standard CoreSight debug port that combines a JTAGDP (5-pin) interface and a SW-DP (2-pin) interface.
  - The JTAG Debug Port (JTAG-DP) provides a 5-pin standard JTAG interface to the AHP-AP port.
  - The Serial Wire Debug Port (SW-DP) provides a 2-pin (clock + data) interface to the AHP-AP port.
- In STM32F401RE, debug is performed using 2 pins only instead of 5 required by the JTAG (JTAG pins could be re-use as GPIO with alternate function): the JTAG TMS and TCK pins are shared with SWDIO and SWCLK, respectively, and a specific sequence on the TMS pin is used to switch between JTAG-DP and SW-DP.



# Debug in Keil Studio

The screenshot displays the Keil Studio IDE interface. The top menu bar includes File, Edit, Selection, View, Go, and Help. The left sidebar contains icons for Threads, Call Stack, Variables, Registers, and Breakpoints. The main editor window shows the source code of `main.cpp`, which is a C++ program for an mbed Microcontroller Library. The program is paused on a breakpoint at line 20, which is highlighted in green. The breakpoint is located at the end of the `ThisThread::sleep_for(BLINKING_RATE);` line. The bottom panel shows the Debug Console with messages indicating the program has halted on a breakpoint.

**Threads:** Main (PAUSED ON BREAKPOINT)

**Call Stack:** main (main.cpp 20:31)

**Variables:**

- Local: `led`: DigitalOut
- Global: `us ticker.c: /home/studio/workspac...`

**Registers:**

- Core: R0: 0x200014a4, R1: 0x00000000, R2: 0x00200000, R3: 0x0000000c, R4: 0x200014a4, R5: 0x20001498, R6: 0x00000000, R7: 0x00000000

**Breakpoints:**

- main.cpp mbed-os-example-blinky (16)
- main.cpp mbed-os-example-blinky (19)
- main.cpp mbed-os-example-blinky (20)

**Source Code (main.cpp):**

```
1  /* mbed Microcontroller Library
2  * Copyright (c) 2019 ARM Limited
3  * SPDX-License-Identifier: Apache-2.0
4  */
5
6  #include "mbed.h"
7
8
9  // Blinking rate in milliseconds
10 #define BLINKING_RATE 500ms
11
12
13 int main()
14 {
15     // Initialise the digital pin LED1 as an output
16     DigitalOut led(LED1);
17
18     while (true) {
19         led = !led;
20         ThisThread::sleep_for(BLINKING_RATE);
21     }
```

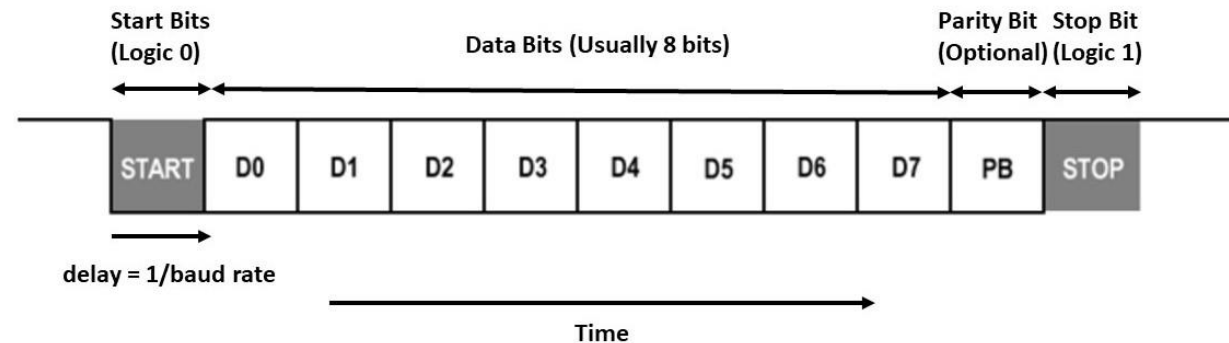
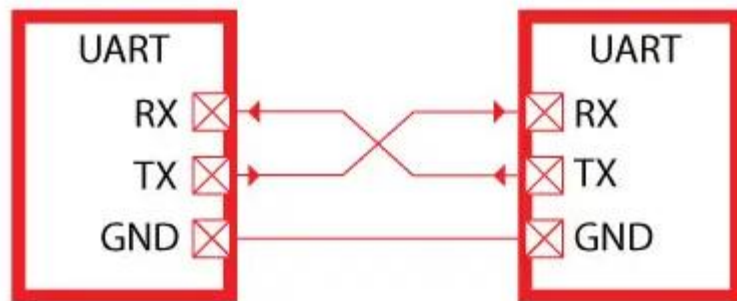
**Debug Console:**

```
go...
Halted on breakpoint
go...
Halted on breakpoint
go...
Halted on breakpoint
go...
Halted on breakpoint
```



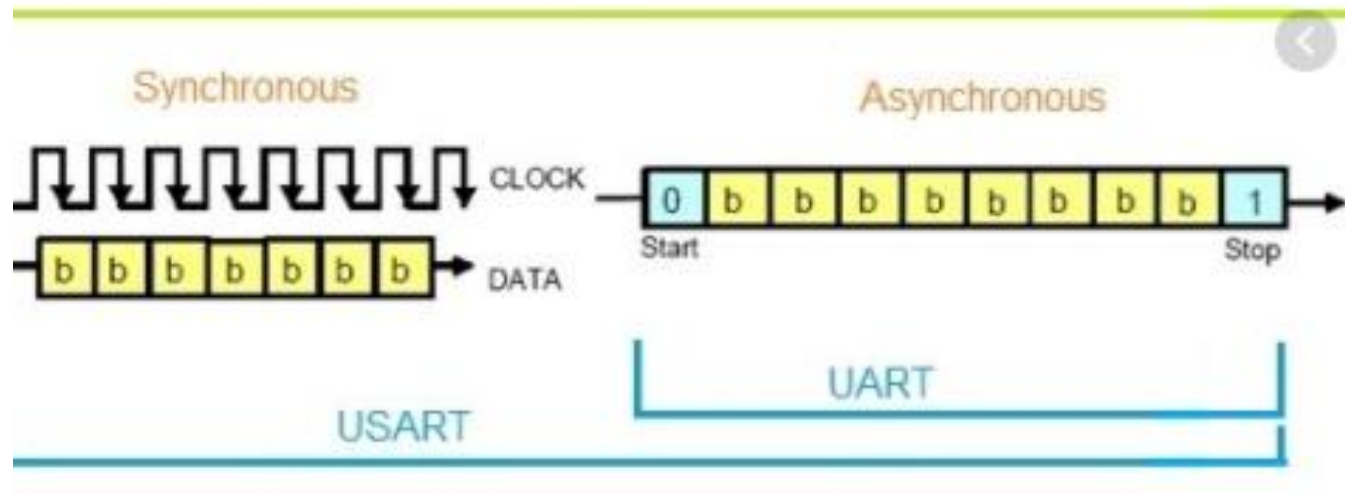
# UART

- Universal Asynchronous Receiver Transmitter (UART) is one of the simplest and the most common protocols for serial communication between hardware devices.
- A UART device requires only two wires, one for transmitting and the other for receiving, to support full duplex communication.
- The UART module converts data into a stream of serial bits and transmits in a packet form shown in the following figure. The data stream is transferred at a fixed **baud rate** in both direction.



# USART

- USART — a Universal **Synchronous**/Asynchronous Receiver/Transmitter
  - In synchronous mode, have a reference clock signal to receiver
  - Receiver could not know the baud rate



# USART Configure

- Baud rate register (USART\_BRR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIV_Mantissa[11:0]												DIV_Fraction[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

# USART Configure (cont'd)

- Control register 3 (USART\_CR3)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

- Set as 0x0000 if no flow control

# USART Configure (cont'd)

- Control register 2 (USART\_CR2)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LINEN	STOP[1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	Res.	ADD[3:0]			
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

- STOP[1:0]: STOP bits
- 00: 1 Stop bit

# USART Configure (cont'd)

## ■ Control register I (USART\_CR1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	Reserved	UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
rw	Res.	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

- Bit 2 RE: Receiver enable
  - 0: Receiver is disabled
  - 1: Receiver is enabled and begins searching for a start bit ← enable RX
- Bit 3 TE: Transmitter enable
  - 0: Transmitter is disabled
  - 1: Transmitter is enabled ← enable TX
- Bit 12 M: Word length.
  - 0: 1 Start bit, 8 Data bits, n Stop bit
  - 1: 1 Start bit, 9 Data bits, n Stop bit
- Bit 13 UE: USART enable
  - 0: USART prescaler and outputs disabled
  - 1: USART enabled
- Bit 7 TXEIE: TXE interrupt enable
  - 0: Interrupt is inhibited
  - 1: An USART interrupt is generated whenever TXE=1 in the USART\_SR register ← After NVIC setting done

# For more details

- Go to board website read more reference document.
  - <https://www.st.com/en/microcontrollers-microprocessors/stm32f401re.html#documentation>
- Reference document.
  - RM0368 Reference manual  
[https://www.st.com/resource/en/reference\\_manual/dm00096844-stm32f401xb-c-and-stm32f401xd-e-advanced-arm-based-32-bit-mcus-stmicroelectronics.pdf](https://www.st.com/resource/en/reference_manual/dm00096844-stm32f401xb-c-and-stm32f401xd-e-advanced-arm-based-32-bit-mcus-stmicroelectronics.pdf)

## Contents

1	Documentation conventions	34
1.1	List of abbreviations for registers	34
1.2	Glossary	35
1.3	Peripheral availability	35
2	Memory and bus architecture	36
2.1	System architecture	36
2.1.1	I-bus	37
2.1.2	D-bus	37
2.1.3	S-bus	37
2.1.4	DMA memory bus	37
2.1.5	DMA peripheral bus	37
2.1.6	BusMatrix	37
2.1.7	AHB/APB bridges (APB)	37
2.2	Memory organization	38
2.3	Memory map	38
2.3.1	Embedded SRAM	40
2.3.2	Flash memory overview	40
2.3.3	Bit banding	40
2.4	Boot configuration	41
3	Embedded Flash memory interface	44
3.1	Introduction	44
3.2	Main features	44
3.3	Embedded Flash memory in STM32F401xB/C and STM32F401xD/E	45
3.4	Read interface	46
3.4.1	Relation between CPU clock frequency and Flash memory read time	46
3.4.2	Adaptive real-time memory accelerator (ART Accelerator™)	47
3.5	Erase and program operations	49
3.5.1	Unlocking the Flash control register	49
3.5.2	Program/erase parallelism	50
3.5.3	Erase	50

## Contents

18.6.5	I <sup>2</sup> C Data register (I2C_DR)	498
18.6.6	I <sup>2</sup> C Status register 1 (I2C_SR1)	498
18.6.7	I <sup>2</sup> C Status register 2 (I2C_SR2)	501
18.6.8	I <sup>2</sup> C Clock control register (I2C_CCR)	503
18.6.9	I <sup>2</sup> C TRISE register (I2C_TRISE)	504
18.6.10	I <sup>2</sup> C FLTR register (I2C_FLTR)	504
18.6.11	I2C register map	505
19	Universal synchronous/asynchronous receiver transmitter (USART)	506
19.1	USART introduction	506
19.2	USART main features	506
19.3	USART functional description	507
19.3.1	USART character description	510
19.3.2	Transmitter	511
19.3.3	Receiver	514
19.3.4	Fractional baud rate generation	519
19.3.5	USART receiver tolerance to clock deviation	529
19.3.6	Multiprocessor communication	530
19.3.7	Parity control	532
19.3.8	LIN (local interconnection network) mode	533
19.3.9	USART synchronous mode	535
19.3.10	Single-wire half-duplex communication	537
19.3.11	Smartcard	538
19.3.12	IrDA SIR ENDEC block	540
19.3.13	Continuous communication using DMA	542
19.3.14	Hardware flow control	544
19.4	USART interrupts	547
19.5	USART mode configuration	548
19.6	USART registers	548
19.6.1	Status register (USART_SR)	548
19.6.2	Data register (USART_DR)	551
19.6.3	Baud rate register (USART_BRR)	551
19.6.4	Control register 1 (USART_CR1)	551
19.6.5	Control register 2 (USART_CR2)	554
19.6.6	Control register 3 (USART_CR3)	555
19.6.7	Guard time and prescaler register (USART_GTPR)	557

## RM0368

## Contents

22.16.2	OTG_FS global registers	700
22.16.3	Host-mode registers	721
22.16.4	Device-mode registers	731
22.16.5	OTG_FS power and clock gating control register (OTG_FS_PCGCTL)	754
22.16.6	OTG_FS register map	755
22.17	OTG_FS programming model	764
22.17.1	Core initialization	764
22.17.2	Host initialization	765
22.17.3	Device initialization	765
22.17.4	Host programming model	766
22.17.5	Device programming model	782
22.17.6	Operational model	784
22.17.7	Worst case response time	800
22.17.8	OTG programming model	802
23	Debug support (DBG)	808
23.1	Overview	808
23.2	Reference Arm® documentation	809
23.3	SWJ debug port (serial wire and JTAG)	809
23.3.1	Mechanism to select the JTAG-DP or the SW-DP	810
23.4	Pinout and debug port pins	810
23.4.1	SWJ debug port pins	811
23.4.2	Flexible SWJ-DP pin assignment	811
23.4.3	Internal pull-up and pull-down on JTAG pins	812
23.4.4	Using serial wire and releasing the unused debug pins as GPIOs	813
23.5	STM32F401xB/C and STM32F401xD/E JTAG TAP connection	813
23.6	ID codes and locking mechanism	815
23.6.1	MCU device ID code	815
23.6.2	Boundary scan TAP	816
23.6.3	Cortex®-M4 with FPU TAP	816
23.6.4	Cortex®-M4 with FPU JEDEC-106 ID code	816
23.7	JTAG debug port	816
23.8	SW debug port	818
23.8.1	SW protocol introduction	818
23.8.2	SW protocol sequence	818