

## 38 Debug support (DBG)

This section applies to the whole STM32F4xx family, unless otherwise specified.

## 38.1 Overview

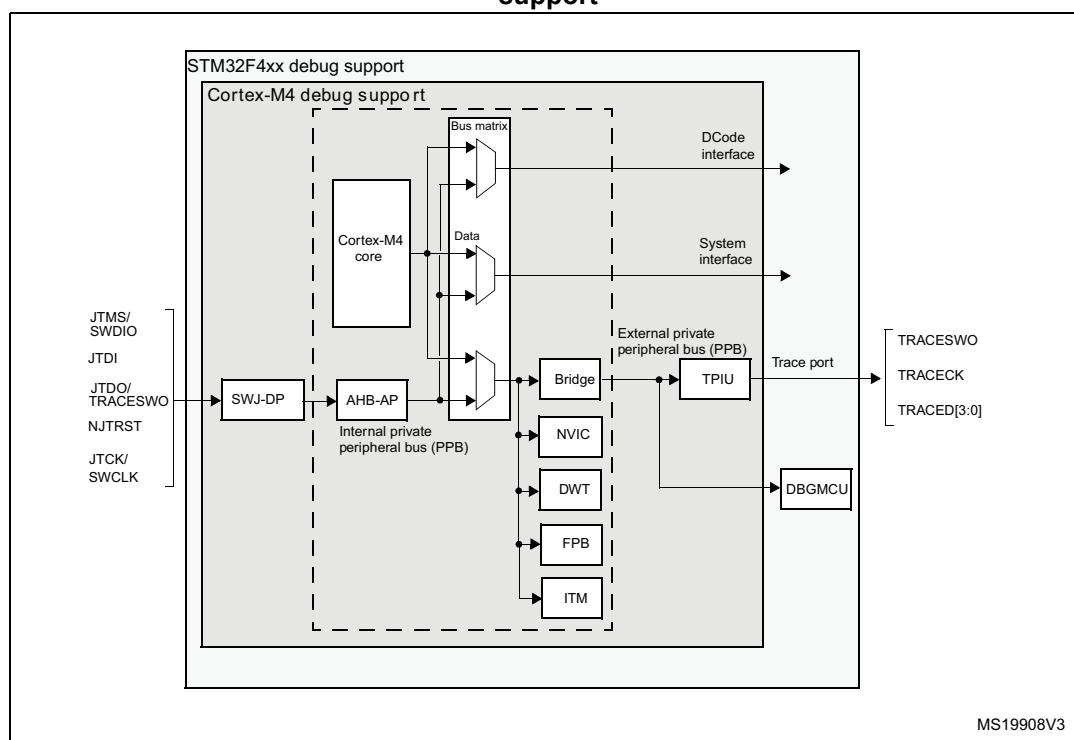
The STM32F4xx are built around a Cortex®-M4 with FPU core, which contains hardware extensions for advanced debugging features. The debug extensions allow the core to be stopped either on a given instruction fetch (breakpoint), or on data access (watchpoint). When stopped, the core's internal state and the system's external state may be examined. Once examination is complete, the core and the system may be restored and program execution resumed.

The debug features are used by the debugger host when connecting to and debugging the STM32F4xx MCUs.

Two interfaces for debug are available:

- Serial wire
- JTAG debug port

**Figure 485. Block diagram of STM32 MCU and Cortex®-M4 with FPU-level debug support**



**Note:** The debug features embedded in the Cortex®-M4 with FPU core are a subset of the Arm® CoreSight Design Kit.

The Arm® Cortex®-M4 with FPU core provides integrated on-chip debug support. It is comprised of:

- SWJ-DP: Serial wire / JTAG debug port
- AHP-AP: AHB access port
- ITM: Instrumentation trace macrocell
- FPB: Flash patch breakpoint
- DWT: Data watchpoint trigger
- TPUI: Trace port unit interface (available on larger packages, where the corresponding pins are mapped)
- ETM: Embedded Trace Macrocell (available on larger packages, where the corresponding pins are mapped)

It also includes debug features dedicated to the STM32F4xx:

- Flexible debug pinout assignment
- MCU debug box (support for low-power modes, control over peripheral clocks, etc.)

*Note:* For further information on debug functionality supported by the Arm® Cortex®-M4 with FPU core, refer to the Cortex®-M4 with FPU -r0p1 Technical Reference Manual and to the CoreSight Design Kit-r0p1 TRM (see [Section 38.2](#)).

## 38.2 Reference Arm® documentation

- Cortex®-M4 with FPU r0p1 Technical Reference Manual (TRM)  
(see Related documents on page 1)
- Arm® Debug Interface V5
- Arm® CoreSight Design Kit revision r0p1 Technical Reference Manual

## 38.3 SWJ debug port (serial wire and JTAG)

The core of the STM32F4xx integrates the Serial Wire / JTAG Debug Port (SWJ-DP). It is an Arm® standard CoreSight debug port that combines a JTAG-DP (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 the SWJ-DP, the two JTAG pins of the SW-DP are multiplexed with some of the five JTAG pins of the JTAG-DP.

Figure 486. SWJ debug port

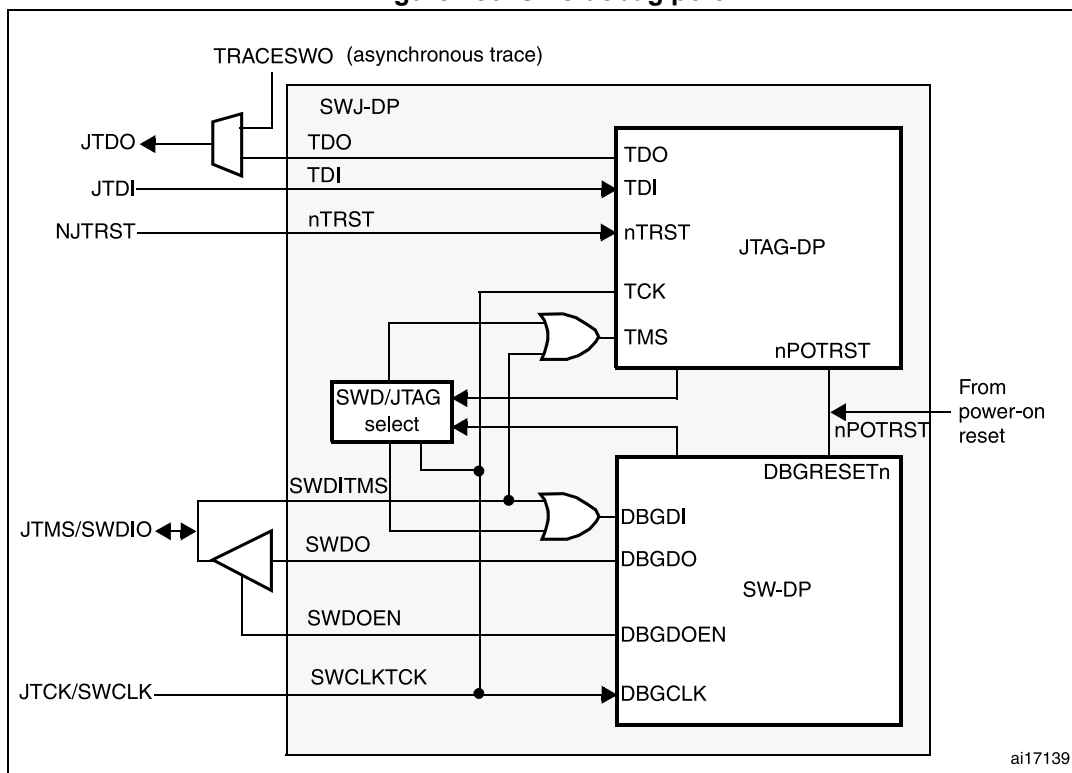


Figure 486 shows that the asynchronous TRACE output (TRACESWO) is multiplexed with TDO. This means that the asynchronous trace can only be used with SW-DP, not JTAG-DP.

### 38.3.1 Mechanism to select the JTAG-DP or the SW-DP

By default, the JTAG-Debug Port is active.

If the debugger host wants to switch to the SW-DP, it must provide a dedicated JTAG sequence on TMS/TCK (respectively mapped to SWDIO and SWCLK) which disables the JTAG-DP and enables the SW-DP. This way it is possible to activate the SWDP using only the SWCLK and SWDIO pins.

This sequence is:

1. Send more than 50 TCK cycles with TMS (SWDIO) = 1
2. Send the 16-bit sequence on TMS (SWDIO) = 0111100111100111 (MSB transmitted first)
3. Send more than 50 TCK cycles with TMS (SWDIO) = 1

## 38.4 Pinout and debug port pins

The STM32F4xx MCUs are available in various packages with different numbers of available pins. As a result, some functionality (ETM) related to pin availability may differ between packages.

### 38.4.1 SWJ debug port pins

Five pins are used as outputs from the STM32F4xx for the SWJ-DP as *alternate functions* of general-purpose I/Os. These pins are available on all packages.

**Table 299. SWJ debug port pins**

SWJ-DP pin name	JTAG debug port		SW debug port		Pin assignment
	Type	Description	Type	Debug assignment	
JTMS/SWDIO	I	JTAG Test Mode Selection	IO	Serial Wire Data Input/Output	PA13
JTCK/SWCLK	I	JTAG Test Clock	I	Serial Wire Clock	PA14
JTDI	I	JTAG Test Data Input	-	-	PA15
JTDO/TRACESWO	O	JTAG Test Data Output	-	TRACESWO if async trace is enabled	PB3
NJTRST	I	JTAG Test nReset	-	-	PB4

### 38.4.2 Flexible SWJ-DP pin assignment

After RESET (SYSRESETn or PORESETn), all five pins used for the SWJ-DP are assigned as dedicated pins immediately usable by the debugger host (note that the trace outputs are not assigned except if explicitly programmed by the debugger host).

However, the STM32F4xx MCUs offers the possibility of disabling some or all of the SWJ-DP ports and so, of releasing the associated pins for general-purpose IO (GPIO) usage. For more details on how to disable SWJ-DP port pins, please refer to [Section 8.3.2: I/O pin multiplexer and mapping](#).

**Table 300. Flexible SWJ-DP pin assignment**

Available debug ports	SWJ IO pin assigned				
	PA13 / JTMS / SWDIO	PA14 / JTCK / SWCLK	PA15 / JTDI	PB3 / JTDO	PB4 / NJTRST
Full SWJ (JTAG-DP + SW-DP) - Reset State	X	X	X	X	X
Full SWJ (JTAG-DP + SW-DP) but without NJTRST	X	X	X	X	
JTAG-DP Disabled and SW-DP Enabled	X	X			
JTAG-DP Disabled and SW-DP Disabled	Released				

### 38.4.3 Internal pull-up and pull-down on JTAG pins

It is necessary to ensure that the JTAG input pins are not floating since they are directly connected to flip-flops to control the debug mode features. Special care must be taken with the SWCLK/TCK pin which is directly connected to the clock of some of these flip-flops.

To avoid any uncontrolled IO levels, the device embeds internal pull-ups and pull-downs on the JTAG input pins:

- NJTRST: Internal pull-up
- JTDI: Internal pull-up
- JTMS/SWDIO: Internal pull-up
- TCK/SWCLK: Internal pull-down

Once a JTAG IO is released by the user software, the GPIO controller takes control again. The reset states of the GPIO control registers put the I/Os in the equivalent state:

- NJTRST: AF input pull-up
- JTDI: AF input pull-up
- JTMS/SWDIO: AF input pull-up
- JTCK/SWCLK: AF input pull-down
- JTDO: AF output floating

The software can then use these I/Os as standard GPIOs.

*Note: The JTAG IEEE standard recommends to add pull-ups on TDI, TMS and nTRST but there is no special recommendation for TCK. However, for JTCK, the device needs an integrated pull-down.*

*Having embedded pull-ups and pull-downs removes the need to add external resistors.*

### 38.4.4 Using serial wire and releasing the unused debug pins as GPIOs

To use the serial wire DP to release some GPIOs, the user software must change the GPIO (PA15, PB3 and PB4) configuration mode in the GPIO\_MODER register. This releases PA15, PB3 and PB4 which now become available as GPIOs.

When debugging, the host performs the following actions:

- Under system reset, all SWJ pins are assigned (JTAG-DP + SW-DP).
- Under system reset, the debugger host sends the JTAG sequence to switch from the JTAG-DP to the SW-DP.
- Still under system reset, the debugger sets a breakpoint on vector reset.
- The system reset is released and the Core halts.
- All the debug communications from this point are done using the SW-DP. The other JTAG pins can then be reassigned as GPIOs by the user software.

*Note:* For user software designs, note that:

*To release the debug pins, remember that they are first configured either in input-pull-up (nTRST, TMS, TDI) or pull-down (TCK) or output tristate (TDO) for a certain duration after reset until the instant when the user software releases the pins.*

*When debug pins (JTAG or SW or TRACE) are mapped, changing the corresponding IO pin configuration in the IOPORT controller has no effect.*

## 38.5 STM32F4xx JTAG TAP connection

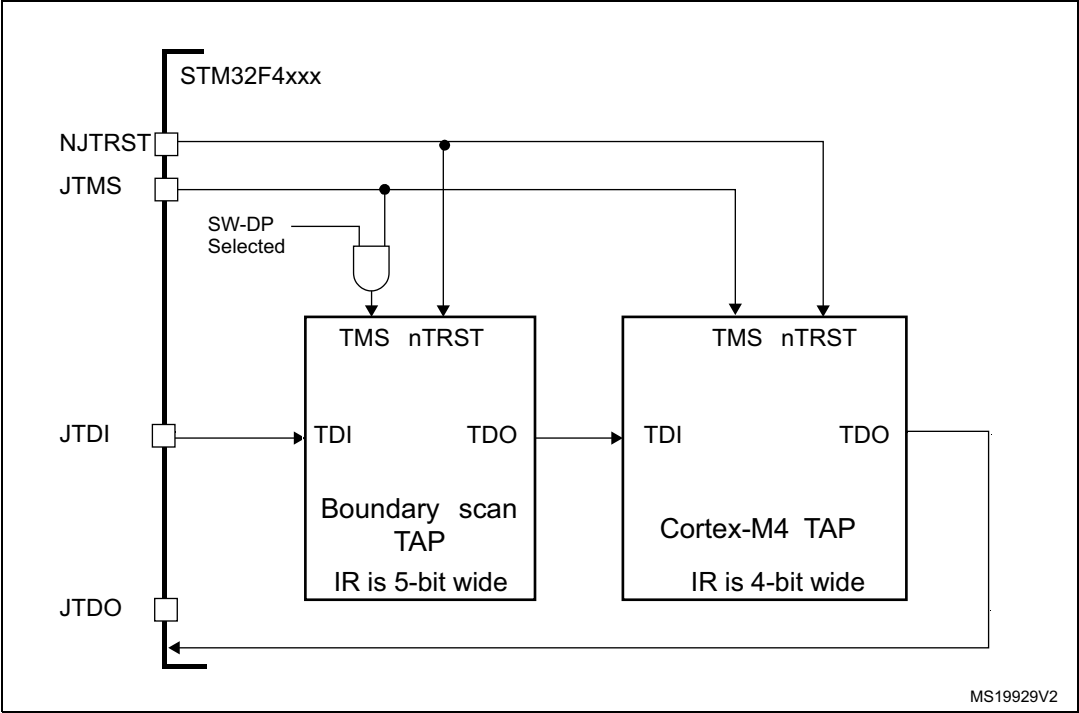
The STM32F4xx MCUs integrate two serially connected JTAG TAPs, the boundary scan TAP (IR is 5-bit wide) and the Cortex®-M4 with FPU TAP (IR is 4-bit wide).

To access the TAP of the Cortex®-M4 with FPU for debug purposes:

1. First, it is necessary to shift the BYPASS instruction of the boundary scan TAP.
2. Then, for each IR shift, the scan chain contains 9 bits (=5+4) and the unused TAP instruction must be shifted in using the BYPASS instruction.
3. For each data shift, the unused TAP, which is in BYPASS mode, adds 1 extra data bit in the data scan chain.

*Note:* **Important:** Once Serial-Wire is selected using the dedicated Arm® JTAG sequence, the boundary scan TAP is automatically disabled (JTMS forced high).

Figure 487. JTAG TAP connections



## 38.6 ID codes and locking mechanism

There are several ID codes inside the STM32F4xx MCUs. ST strongly recommends tools designers to lock their debuggers using the MCU DEVICE ID code located in the external PPB memory map at address 0xE0042000.

### 38.6.1 MCU device ID code

The STM32F4xx MCUs integrate an MCU ID code. This ID identifies the ST MCU part-number and the die revision. It is part of the DBG\_MCU component and is mapped on the external PPB bus (see [Section 38.16](#)). This code is accessible using the JTAG debug port (four to five pins) or the SW debug port (two pins) or by the user software. It is even accessible while the MCU is under system reset.

Only the DEV\_ID[11:0] must be used for identification by the debugger/programmer tools.

#### DBGMCU\_IDCODE

Address: 0xE004 2000

Only 32-bits access supported. Read-only.

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

Bits 31:16 **REV\_ID[15:0]** Revision identifier

This field indicates the revision of the device.

STM32F405xx/07xx and STM32F415xx/17xx devices:

0x1000 = Revision A

0x1001 = Revision Z

0x1003 = Revision 1

0x1007 = Revision 2

0x100F = Revision Y and 4

0x101F = Revision 5 and 6

STM32F42xxx and STM32F43xxx devices:

0x1000 = Revision A

0x1003 = Revision Y

0x1007 = Revision 1

0x2001 = Revision 3

0x2003 = Revision 4, 5 and B

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DEV\_ID[11:0]**: Device identifier (STM32F405xx/07xx and STM32F415xx/17xx)

The device ID is 0x413.

Bits 11:0 **DEV\_ID[11:0]**: Device identifier (STM32F42xxx and STM32F43xxx)

The device ID is 0x419



## 38.6.2 Boundary scan TAP

### JTAG ID code

The TAP of the STM32F4xx BSC (boundary scan) integrates a JTAG ID code equal to .

- 0x06413041 for STM32F405xx/07xx and STM32F415xx/17xx devices
- 0x06419041 for STM32F42xxx and STM32F43xxx devices

## 38.6.3 Cortex®-M4 with FPU TAP

The TAP of the Arm® Cortex®-M4 with FPU integrates a JTAG ID code. This ID code is the Arm® default one and has not been modified. This code is only accessible by the JTAG Debug Port, it is 0x4BA00477 (corresponds to Cortex®-M4 with FPU r0p1, see [Section 38.2](#)).

## 38.6.4 Cortex®-M4 with FPU JEDEC-106 ID code

The Arm® Cortex®-M4 with FPU integrates a JEDEC-106 ID code. It is located in the 4 KB ROM table mapped on the internal PPB bus at address 0xE00FF000\_0xE00FFFFF.

This code is accessible by the JTAG Debug Port (4 to 5 pins) or by the SW Debug Port (two pins) or by the user software.

## 38.7 JTAG debug port

A standard JTAG state machine is implemented with a 4-bit instruction register (IR) and five data registers (for full details, refer to the Cortex®-M4 with FPU r0p1 *Technical Reference Manual (TRM)*, for references, see [Section 38.2](#)).

**Table 301. JTAG debug port data registers**

IR(3:0)	Data register	Details
1111	BYPASS [1 bit]	-
1110	IDCODE [32 bits]	ID CODE 0x4BA00477 (Arm® Cortex®-M4 with FPU r0p1 ID Code)
1010	DPACC [35 bits]	<p>Debug port access register</p> <p>This initiates a debug port and allows access to a debug port register.</p> <ul style="list-style-type: none"> <li>– When transferring data IN: <ul style="list-style-type: none"> <li>Bits 34:3 = DATA[31:0] = 32-bit data to transfer for a write request</li> <li>Bits 2:1 = A[3:2] = 2-bit address of a debug port register.</li> <li>Bit 0 = RnW = Read request (1) or write request (0).</li> </ul> </li> <li>– When transferring data OUT: <ul style="list-style-type: none"> <li>Bits 34:3 = DATA[31:0] = 32-bit data which is read following a read request</li> <li>Bits 2:0 = ACK[2:0] = 3-bit Acknowledge: <ul style="list-style-type: none"> <li>010 = OK/FAULT</li> <li>001 = WAIT</li> <li>OTHER = reserved</li> </ul> </li> </ul> </li> </ul> <p>Refer to <a href="#">Table 302</a> for a description of the A[3:2] bits</p>

**Table 301. JTAG debug port data registers (continued)**

IR(3:0)	Data register	Details
1011	APACC [35 bits]	<p>Access port access register</p> <p>Initiates an access port and allows access to an access port register.</p> <ul style="list-style-type: none"> <li>– When transferring data IN: <ul style="list-style-type: none"> <li>Bits 34:3 = DATA[31:0] = 32-bit data to shift in for a write request</li> <li>Bits 2:1 = A[3:2] = 2-bit address (sub-address AP registers).</li> <li>Bit 0 = RnW= Read request (1) or write request (0).</li> </ul> </li> <li>– When transferring data OUT: <ul style="list-style-type: none"> <li>Bits 34:3 = DATA[31:0] = 32-bit data which is read following a read request</li> <li>Bits 2:0 = ACK[2:0] = 3-bit Acknowledge: <ul style="list-style-type: none"> <li>010 = OK/FAULT</li> <li>001 = WAIT</li> <li>OTHER = reserved</li> </ul> </li> </ul> </li> </ul> <p>There are many AP registers (see AHB-AP) addressed as the combination of:</p> <ul style="list-style-type: none"> <li>– The shifted value A[3:2]</li> <li>– The current value of the DP SELECT register</li> </ul>
1000	ABORT [35 bits]	<p>Abort register</p> <ul style="list-style-type: none"> <li>– Bits 31:1 = Reserved</li> <li>– Bit 0 = DAPABORT: write 1 to generate a DAP abort.</li> </ul>

**Table 302. 32-bit debug port registers addressed through the shifted value A[3:2]**

Address	A[3:2] value	Description
0x0	00	Reserved, must be kept at reset value.
0x4	01	<p>DP CTRL/STAT register. Used to:</p> <ul style="list-style-type: none"> <li>– Request a system or debug power-up</li> <li>– Configure the transfer operation for AP accesses</li> <li>– Control the pushed compare and pushed verify operations.</li> <li>– Read some status flags (overrun, power-up acknowledges)</li> </ul>
0x8	10	<p>DP SELECT register: Used to select the current access port and the active 4-words register window.</p> <ul style="list-style-type: none"> <li>– Bits 31:24: APSEL: select the current AP</li> <li>– Bits 23:8: reserved</li> <li>– Bits 7:4: APBANKSEL: select the active 4-words register window on the current AP</li> <li>– Bits 3:0: reserved</li> </ul>
0xC	11	<p>DP RDBUFF register: Used to allow the debugger to get the final result after a sequence of operations (without requesting new JTAG-DP operation)</p>

## 38.8 SW debug port

### 38.8.1 SW protocol introduction

This synchronous serial protocol uses two pins:

- SWCLK: clock from host to target
- SWDIO: bidirectional

The protocol allows two banks of registers (DPACC registers and APACC registers) to be read and written to.

Bits are transferred LSB-first on the wire.

For SWDIO bidirectional management, the line must be pulled-up on the board (100 K $\Omega$  recommended by Arm<sup>®</sup>).

Each time the direction of SWDIO changes in the protocol, a turnaround time is inserted where the line is not driven by the host nor the target. By default, this turnaround time is one bit time, however this can be adjusted by configuring the SWCLK frequency.

### 38.8.2 SW protocol sequence

Each sequence consist of three phases:

1. Packet request (8 bits) transmitted by the host
2. Acknowledge response (3 bits) transmitted by the target
3. Data transfer phase (33 bits) transmitted by the host or the target

**Table 303. Packet request (8-bits)**

Bit	Name	Description
0	Start	Must be "1"
1	APnDP	0: DP Access 1: AP Access
2	RnW	0: Write Request 1: Read Request
4:3	A[3:2]	Address field of the DP or AP registers (refer to <a href="#">Table 302</a> )
5	Parity	Single bit parity of preceding bits
6	Stop	0
7	Park	Not driven by the host. Must be read as "1" by the target because of the pull-up

Refer to the Cortex<sup>®</sup>-M4 with FPU r0p1 *TRM* for a detailed description of DPACC and APACC registers.

The packet request is always followed by the turnaround time (default 1 bit) where neither the host nor target drive the line.

Table 304. ACK response (3 bits)

Bit	Name	Description
0..2	ACK	001: FAULT 010: WAIT 100: OK

The ACK Response must be followed by a turnaround time only if it is a READ transaction or if a WAIT or FAULT acknowledge has been received.

Table 305. DATA transfer (33 bits)

Bit	Name	Description
0..31	WDATA or RDATA	Write or Read data
32	Parity	Single parity of the 32 data bits

The DATA transfer must be followed by a turnaround time only if it is a READ transaction.

### 38.8.3 SW-DP state machine (reset, idle states, ID code)

The State Machine of the SW-DP has an internal ID code which identifies the SW-DP. It follows the JEP-106 standard. This ID code is the default Arm® one and is set to **0x2BA01477** (corresponding to Cortex®-M4 with FPU r0p1).

*Note:*

*Note that the SW-DP state machine is inactive until the target reads this ID code.*

- The SW-DP state machine is in RESET STATE either after power-on reset, or after the DP has switched from JTAG to SWD or after the line is high for more than 50 cycles
- The SW-DP state machine is in IDLE STATE if the line is low for at least two cycles after RESET state.
- After RESET state, it is **mandatory** to first enter into an IDLE state AND to perform a READ access of the DP-SW ID CODE register. Otherwise, the target issues a FAULT acknowledge response on another transactions.

Further details of the SW-DP state machine can be found in the Cortex®-M4 with FPU r0p1 *TRM* and the *CoreSight Design Kit r0p1 TRM*.

### 38.8.4 DP and AP read/write accesses

- Read accesses to the DP are not posted: the target response can be immediate (if ACK=OK) or can be delayed (if ACK=WAIT).
- Read accesses to the AP are posted. This means that the result of the access is returned on the next transfer. If the next access to be done is NOT an AP access, then the DP-RDBUFF register must be read to obtain the result.  
The READOK flag of the DP-CTRL/STAT register is updated on every AP read access or RDBUFF read request to know if the AP read access was successful.
- The SW-DP implements a write buffer (for both DP or AP writes), that enables it to accept a write operation even when other transactions are still outstanding. If the write buffer is full, the target acknowledge response is "WAIT". With the exception of

IDCODE read or CTRL/STAT read or ABORT write which are accepted even if the write buffer is full.

- Because of the asynchronous clock domains SWCLK and HCLK, two extra SWCLK cycles are needed after a write transaction (after the parity bit) to make the write effective internally. These cycles must be applied while driving the line low (IDLE state). This is particularly important when writing the CTRL/STAT for a power-up request. If the next transaction (requiring a power-up) occurs immediately, it fails.

### 38.8.5 SW-DP registers

Access to these registers are initiated when APnDP=0

**Table 306. SW-DP registers**

A[3:2]	R/W	CTRLSEL bit of SELECT register	Register	Notes
00	Read	-	IDCODE	The manufacturer code is not set to ST code. <b>0x2BA01477</b> (identifies the SW-DP)
00	Write	-	ABORT	-
01	Read/Write	0	DP-CTRL/STAT	Purpose is to: <ul style="list-style-type: none"> <li>request a system or debug power-up</li> <li>configure the transfer operation for AP accesses</li> <li>control the pushed compare and pushed verify operations.</li> <li>read some status flags (overrun, power-up acknowledges)</li> </ul>
01	Read/Write	1	WIRE CONTROL	Purpose is to configure the physical serial port protocol (like the duration of the turnaround time)
10	Read	-	READ RESEND	Enables recovery of the read data from a corrupted debugger transfer, without repeating the original AP transfer.
10	Write	-	SELECT	The purpose is to select the current access port and the active 4-words register window
11	Read/Write	-	READ BUFFER	This read buffer is useful because AP accesses are posted (the result of a read AP request is available on the next AP transaction). This read buffer captures data from the AP, presented as the result of a previous read, without initiating a new transaction

### 38.8.6 SW-AP registers

Access to these registers are initiated when APnDP=1

There are many AP registers (see AHB-AP) addressed as the combination of:

- The shifted value A[3:2]
- The current value of the DP SELECT register

## 38.9 AHB-AP (AHB access port) - valid for both JTAG-DP and SW-DP

### Features:

- System access is independent of the processor status.
- Either SW-DP or JTAG-DP accesses AHB-AP.
- The AHB-AP is an AHB master into the Bus Matrix. Consequently, it can access all the data buses (Dcode Bus, System Bus, internal and external PPB bus) but the ICode bus.
- Bitband transactions are supported.
- AHB-AP transactions bypass the FPB.

The address of the 32-bits AHB-AP registers are 6-bits wide (up to 64 words or 256 bytes) and consists of:

- c) Bits [7:4] = the bits [7:4] APBANKSEL of the DP SELECT register
- d) Bits [3:2] = the 2 address bits of A[3:2] of the 35-bit packet request for SW-DP.

The AHB-AP of the Cortex<sup>®</sup>-M4 with FPU includes 9 x 32-bits registers:

**Table 307. Cortex<sup>®</sup>-M4 with FPU AHB-AP registers**

Address offset	Register name	Notes
0x00	AHB-AP Control and Status Word	Configures and controls transfers through the AHB interface (size, hprot, status on current transfer, address increment type)
0x04	AHB-AP Transfer Address	-
0x0C	AHB-AP Data Read/Write	-
0x10	AHB-AP Banked Data 0	Directly maps the 4 aligned data words without rewriting the Transfer Address register.
0x14	AHB-AP Banked Data 1	
0x18	AHB-AP Banked Data 2	
0x1C	AHB-AP Banked Data 3	
0xF8	AHB-AP Debug ROM Address	Base Address of the debug interface
0xFC	AHB-AP ID register	-

Refer to the Cortex<sup>®</sup>-M4 with FPU *r0p1 TRM* for further details.

## 38.10 Core debug

Core debug is accessed through the core debug registers. Debug access to these registers is by means of the *Advanced High-performance Bus* (AHB-AP) port. The processor can access these registers directly over the internal *Private Peripheral Bus* (PPB).

It consists of 4 registers:

**Table 308. Core debug registers**

Register	Description
DHCSR	The 32-bit Debug Halting Control and Status register This provides status information about the state of the processor enable core debug halt and step the processor
DCRSR	The 17-bit Debug Core register Selector register: This selects the processor register to transfer data to or from.
DCRDR	The 32-bit Debug Core register Data register: This holds data for reading and writing registers to and from the processor selected by the DCRSR (Selector) register.
DEMCR	The 32-bit Debug Exception and Monitor Control register: This provides Vector Catching and Debug Monitor Control. This register contains a bit named <b>TRCENA</b> which enable the use of a TRACE.

*Note:* **Important:** these registers are not reset by a system reset. They are only reset by a power-on reset.

Refer to the Cortex<sup>®</sup>-M4 with FPU r0p1 TRM for further details.

To Halt on reset, it is necessary to:

- enable the bit0 (VC\_CORRESET) of the Debug and Exception Monitor Control register
- enable the bit0 (C\_DEBUGEN) of the Debug Halting Control and Status register.

### 38.11 Capability of the debugger host to connect under system reset

The reset system of the STM32F4xx MCU comprises the following reset sources:

- POR (power-on reset) which asserts a RESET at each power-up.
- Internal watchdog reset
- Software reset
- External reset

The Cortex®-M4 with FPU differentiates the reset of the debug part (generally PORRESETn) and the other one (SYSRESETn)

This way, it is possible for the debugger to connect under System Reset, programming the Core Debug registers to halt the core when fetching the reset vector. Then the host can release the system reset and the core immediately halts without having executed any instructions. In addition, it is possible to program any debug features under System Reset.

*Note: It is highly recommended for the debugger host to connect (set a breakpoint in the reset vector) under system reset.*

### 38.12 FPB (Flash patch breakpoint)

The FPB unit:

- implements hardware breakpoints
- patches code and data from code space to system space. This feature gives the possibility to correct software bugs located in the Code Memory Space.

The use of a Software Patch or a Hardware Breakpoint is exclusive.

The FPB consists of:

- 2 literal comparators for matching against literal loads from Code Space and remapping to a corresponding area in the System Space.
- 6 instruction comparators for matching against instruction fetches from Code Space. They can be used either to remap to a corresponding area in the System Space or to generate a Breakpoint Instruction to the core.



## 38.13 DWT (data watchpoint trigger)

The DWT unit consists of four comparators. They are configurable as:

- a hardware watchpoint or
- a trigger to an ETM or
- a PC sampler or
- a data address sampler

The DWT also provides some means to give some profiling informations. For this, some counters are accessible to give the number of:

- Clock cycle
- Folded instructions
- Load store unit (LSU) operations
- Sleep cycles
- CPI (clock per instructions)
- Interrupt overhead

## 38.14 ITM (instrumentation trace macrocell)

### 38.14.1 General description

The ITM is an application-driven trace source that supports *printf* style debugging to trace *Operating System* (OS) and application events, and emits diagnostic system information. The ITM emits trace information as packets which can be generated as:

- **Software trace.** Software can write directly to the ITM stimulus registers to emit packets.
- **Hardware trace.** The DWT generates these packets, and the ITM emits them.
- **Time stamping.** Timestamps are emitted relative to packets. The ITM contains a 21-bit counter to generate the timestamp. The Cortex<sup>®</sup>-M4 with FPU clock or the bit clock rate of the *Serial Wire Viewer* (SWV) output clocks the counter.

The packets emitted by the ITM are output to the TPIU (Trace Port Interface Unit). The formatter of the TPIU adds some extra packets (refer to TPIU) and then output the complete packets sequence to the debugger host.

The bit TRCEN of the Debug Exception and Monitor Control register must be enabled before programming or using the ITM.

### 38.14.2 Time stamp packets, synchronization and overflow packets

Time stamp packets encode time stamp information, generic control and synchronization. It uses a 21-bit timestamp counter (with possible prescalers) which is reset at each time stamp packet emission. This counter can be either clocked by the CPU clock or the SWV clock.

A synchronization packet consists of 6 bytes equal to 0x80\_00\_00\_00\_00\_00 which is emitted to the TPIU as 00 00 00 00 00 80 (LSB emitted first).

A synchronization packet is a timestamp packet control. It is emitted at each DWT trigger.

For this, the DWT must be configured to trigger the ITM: the bit CYCCNTENA (bit0) of the DWT Control register must be set. In addition, the bit2 (SYNCENA) of the ITM Trace Control register must be set.

*Note: If the SYNENA bit is not set, the DWT generates Synchronization triggers to the TPIU which sends only TPIU synchronization packets and not ITM synchronization packets.*

An overflow packet consists is a special timestamp packets which indicates that data has been written but the FIFO was full.

**Table 309. Main ITM registers**

Address	Register	Details
@E0000FB0	ITM lock access	Write 0xC5ACCE55 to unlock Write Access to the other ITM registers
@E0000E80	ITM trace control	Bits 31-24 = Always 0
		Bits 23 = Busy
		Bits 22-16 = 7-bits ATB ID which identifies the source of the trace data.
		Bits 15-10 = Always 0
		Bits 9:8 = TSPrescale = Time Stamp Prescaler
		Bits 7-5 = Reserved
		Bit 4 = SWOENA = Enable SWV behavior (to clock the timestamp counter by the SWV clock).
		Bit 3 = DWTENA: Enable the DWT Stimulus
		Bit 2 = SYNCENA: this bit must be to 1 to enable the DWT to generate synchronization triggers so that the TPIU can then emit the synchronization packets.
		Bit 1 = TSENA (Timestamp Enable)
		Bit 0 = ITMENA: Global Enable Bit of the ITM
@E0000E40	ITM trace privilege	Bit 3: mask to enable tracing ports31:24
		Bit 2: mask to enable tracing ports23:16
		Bit 1: mask to enable tracing ports15:8
		Bit 0: mask to enable tracing ports7:0
@E0000E00	ITM trace enable	Each bit enables the corresponding Stimulus port to generate trace.
@E0000000- E000007C	Stimulus port registers 0-31	Write the 32-bits data on the selected Stimulus Port (32 available) to be traced out.

### Example of configuration

To output a simple value to the TPIU:

- Configure the TPIU and assign TRACE I/Os by configuring the DBGMCU\_CR (refer to [Section 38.17.2](#) and [Section 38.16.3](#))
- Write 0xC5ACCE55 to the ITM Lock Access register to unlock the write access to the ITM registers
- Write 0x00010005 to the ITM Trace Control register to enable the ITM with Sync enabled and an ATB ID different from 0x00
- Write 0x1 to the ITM Trace Enable register to enable the Stimulus Port 0
- Write 0x1 to the ITM Trace Privilege register to unmask stimulus ports 7:0
- Write the value to output in the Stimulus Port register 0: this can be done by software (using a printf function)

## 38.15 ETM (Embedded Trace Macrocell™)

### 38.15.1 ETM general description

The ETM enables the reconstruction of program execution. Data are traced using the Data Watchpoint and Trace (DWT) component or the Instruction Trace Macrocell (ITM) whereas instructions are traced using the Embedded Trace Macrocell (ETM).

The ETM transmits information as packets and is triggered by embedded resources. These resources must be programmed independently and the trigger source is selected using the Trigger Event register (0xE0041008). An event could be a simple event (address match from an address comparator) or a logic equation between 2 events. The trigger source is one of the fourth comparators of the DWT module. The following events can be monitored:

- Clock cycle matching
- Data address matching

For more informations on the trigger resources refer to [Section 38.13](#).

The packets transmitted by the ETM are output to the TPIU (Trace Port Interface Unit). The formatter of the TPIU adds some extra packets (refer to [Section 38.17](#)) and then outputs the complete packet sequence to the debugger host.

### 38.15.2 ETM signal protocol and packet types

This part is described in the chapter 7 ETMv3 Signal Protocol of the Arm® IHI 0014N document.

### 38.15.3 Main ETM registers

For more information on registers refer to the chapter 3 of the Arm® IHI 0014N specification.

**Table 310. Main ETM registers**

Address	Register	Details
0xE0041FB0	ETM Lock Access	Write 0xC5ACCE55 to unlock the write access to the other ETM registers.
0xE0041000	ETM Control	This register controls the general operation of the ETM, for instance how tracing is enabled.
0xE0041010	ETM Status	This register provides information about the current status of the trace and trigger logic.
0xE0041008	ETM Trigger Event	This register defines the event that controls trigger.
0xE004101C	ETM Trace Enable Control	This register defines which comparator is selected.
0xE0041020	ETM Trace Enable Event	This register defines the trace enabling event.
0xE0041024	ETM Trace Start/Stop	This register defines the traces used by the trigger source to start and stop the trace, respectively.

### 38.15.4 ETM configuration example

To output a simple value to the TPIU:

- Configure the TPIU and enable the I/O\_TRACEN to assign TRACE I/Os in the STM32F4xx debug configuration register.
- Write 0xC5AC CE55 to the ETM Lock Access register to unlock the write access to the ITM registers
- Write 0x0000 1D1E to the ETM control register (configure the trace)
- Write 0x0000 406F to the ETM Trigger Event register (define the trigger event)
- Write 0x0000 006F to the ETM Trace Enable Event register (define an event to start/stop)
- Write 0x0200 0000x0000 0001 to the ETM Trace Start/stop register (enable the trace)
- Write 0x0000191E to the ETM Control register (end of configuration)

## 38.16 MCU debug component (DBGMCU)

The MCU debug component helps the debugger provide support for:

- Low-power modes
- Clock control for timers, watchdog, I2C and bxCAN during a breakpoint
- Control of the trace pins assignment

### 38.16.1 Debug support for low-power modes

To enter low-power mode, the instruction WFI or WFE must be executed.

The MCU implements several low-power modes which can either deactivate the CPU clock or reduce the power of the CPU.

The core does not allow FCLK or HCLK to be turned off during a debug session. As these are required for the debugger connection, during a debug, they must remain active. The MCU integrates special means to allow the user to debug software in low-power modes.

For this, the debugger host must first set some debug configuration registers to change the low-power mode behavior:

- In Sleep mode, DBG\_SLEEP bit of DBGMCU\_CR register must be previously set by the debugger. This feeds HCLK with the same clock that is provided to FCLK (system clock previously configured by the software).
- In Stop mode, the bit DBG\_STOP must be previously set by the debugger. This enables the internal RC oscillator clock to feed FCLK and HCLK in STOP mode.

### 38.16.2 Debug support for timers, watchdog, bxCAN and I<sup>2</sup>C

During a breakpoint, it is necessary to choose how the counter of timers and watchdog must behave:

- They can continue to count inside a breakpoint. This is usually required when a PWM is controlling a motor, for example.
- They can stop to count inside a breakpoint. This is required for watchdog purposes.

For the bxCAN, the user can choose to block the update of the receive register during a breakpoint.

For the I<sup>2</sup>C, the user can choose to block the SMBUS timeout during a breakpoint.

For timers having complementary outputs, when the counter is stopped (DBG\_TIMx\_STOP = 1), the outputs are disabled (as if the MOE bit was reset) for safety purposes.

### 38.16.3 Debug MCU configuration register

This register allows the configuration of the MCU under DEBUG. This concerns:

- Low-power mode support
- Timer and watchdog counter support
- bxCAN communication support
- Trace pin assignment

This DBGMCU\_CR is mapped on the External PPB bus at address 0xE0042004

It is asynchronously reset by the PORESET (and not the system reset). It can be written by the debugger under system reset.

If the debugger host does not support these features, it is still possible for the user software to write to these registers.

#### DBGMCU\_CR register

Address: 0xE004 2004

Only 32-bit access supported

POR Reset: 0x0000 0000 (not reset by system reset)

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								TRACE_MODE [1:0]		TRACE_IOEN	Reserved		DBG_STANDBY	DBG_STOP	DBG_SLEEP
								rw	rw	rw			rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:5 **TRACE\_MODE[1:0] and TRACE\_IOEN**: Trace pin assignment control

- With TRACE\_IOEN=0:
  - TRACE\_MODE=xx: TRACE pins not assigned (default state)
- With TRACE\_IOEN=1:
  - TRACE\_MODE=00: TRACE pin assignment for Asynchronous mode
  - TRACE\_MODE=01: TRACE pin assignment for Synchronous mode with a TRACEDATA size of 1
  - TRACE\_MODE=10: TRACE pin assignment for Synchronous mode with a TRACEDATA size of 2
  - TRACE\_MODE=11: TRACE pin assignment for Synchronous mode with a TRACEDATA size of 4

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **DBG\_STANDBY**: Debug Standby mode

0: (FCLK=Off, HCLK=Off) The whole digital part is unpowered.

From software point of view, exiting from Standby is identical than fetching reset vector (except a few status bit indicated that the MCU is resuming from Standby)

1: (FCLK=On, HCLK=On) In this case, the digital part is not unpowered and FCLK and HCLK are provided by the internal RC oscillator which remains active. In addition, the MCU generate a system reset during Standby mode so that exiting from Standby is identical than fetching from reset

Bit 1 **DBG\_STOP**: Debug Stop mode

0: (FCLK=Off, HCLK=Off) In STOP mode, the clock controller disables all clocks (including HCLK and FCLK). When exiting from STOP mode, the clock configuration is identical to the one after RESET (CPU clocked by the 8 MHz internal RC oscillator (HSI)). Consequently, the software must reprogram the clock controller to enable the PLL, the Xtal, etc.

1: (FCLK=On, HCLK=On) In this case, when entering STOP mode, FCLK and HCLK are provided by the internal RC oscillator which remains active in STOP mode. When exiting STOP mode, the software must reprogram the clock controller to enable the PLL, the Xtal, etc. (in the same way it would do in case of DBG\_STOP=0)

Bit 0 **DBG\_SLEEP**: Debug Sleep mode

0: (FCLK=On, HCLK=Off) In Sleep mode, FCLK is clocked by the system clock as previously configured by the software while HCLK is disabled.

In Sleep mode, the clock controller configuration is not reset and remains in the previously programmed state. Consequently, when exiting from Sleep mode, the software does not need to reconfigure the clock controller.

1: (FCLK=On, HCLK=On) In this case, when entering Sleep mode, HCLK is fed by the same clock that is provided to FCLK (system clock as previously configured by the software).

### 38.16.4 Debug MCU APB1 freeze register (DBGMCU\_APB1\_FZ)

The DBGMCU\_APB1\_FZ register is used to configure the MCU under Debug. It concerns APB1 peripherals. It is mapped on the external PPB bus at address 0xE004 2008.

The register is asynchronously reset by the POR (and not the system reset). It can be written by the debugger under system reset.

Address : 0xE004 2008

Only 32-bits access are supported.

Power-on reset (POR): 0x0000 0000 (not reset by system reset)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					DBG_CAN2_STOP	DBG_CAN1_STOP	Reserved	DBG_I2C3_SMBUS_TIMEOUT	DBG_I2C2_SMBUS_TIMEOUT	DBG_I2C1_SMBUS_TIMEOUT	Reserved				
					rw	rw		rw	rw	rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DBG_IWDG_STOP	DBG_WWDG_STOP	DBG_RTC_STOP	Reserved	DBG_TIM14_STOP	DBG_TIM13_STOP	DBG_TIM12_STOP	DBG_TIM7_STOP	DBG_TIM6_STOP	DBG_TIM5_STOP	DBG_TIM4_STOP	DBG_TIM3_STOP
				rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **DBG\_CAN2\_STOP**: Debug CAN2 stopped when Core is halted

- 0: Same behavior as in normal mode
- 1: The CAN2 receive registers are frozen

Bit 25 **DBG\_CAN1\_STOP**: Debug CAN2 stopped when Core is halted

- 0: Same behavior as in normal mode
- 1: The CAN2 receive registers are frozen

Bit 24 Reserved, must be kept at reset value.

Bit 23 **DBG\_I2C3\_SMBUS\_TIMEOUT**: SMBUS timeout mode stopped when Core is halted

- 0: Same behavior as in normal mode
- 1: The SMBUS timeout is frozen

Bit 22 **DBG\_I2C2\_SMBUS\_TIMEOUT**: SMBUS timeout mode stopped when Core is halted

- 0: Same behavior as in normal mode
- 1: The SMBUS timeout is frozen

Bit 21 **DBG\_I2C1\_SMBUS\_TIMEOUT**: SMBUS timeout mode stopped when Core is halted

- 0: Same behavior as in normal mode
- 1: The SMBUS timeout is frozen

Bit 20:13 Reserved, must be kept at reset value.

Bit 12 **DBG\_IWDG\_STOP**: Debug independent watchdog stopped when core is halted  
 0: The independent watchdog counter clock continues even if the core is halted  
 1: The independent watchdog counter clock is stopped when the core is halted

Bit 11 **DBG\_WWDG\_STOP**: Debug Window Watchdog stopped when Core is halted  
 0: The window watchdog counter clock continues even if the core is halted  
 1: The window watchdog counter clock is stopped when the core is halted

Bit 10 **DBG\_RTC\_STOP**: RTC stopped when Core is halted  
 0: The RTC counter clock continues even if the core is halted  
 1: The RTC counter clock is stopped when the core is halted

Bit 9 Reserved, must be kept at reset value.

Bits 8:0 **DBG\_TIMx\_STOP**: TIMx counter stopped when core is halted (x=2..7, 12..14)  
 0: The clock of the involved timer counter is fed even if the core is halted  
 1: The clock of the involved timer counter is stopped and the outputs are disabled when the core is halted

### 38.16.5 Debug MCU APB2 Freeze register (DBGMCU\_APB2\_FZ)

The DBGMCU\_APB2\_FZ register is used to configure the MCU under Debug. It concerns APB2 peripherals.

This register is mapped on the external PPB bus at address 0xE004 200C

It is asynchronously reset by the POR (and not the system reset). It can be written by the debugger under system reset.

Address: 0xE004 200C

Only 32-bit access is supported.

POR: 0x0000 0000 (not reset by system reset)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													DBG_TIM11_STOP	DBG_TIM10_STOP	DBG_TIM9_STOP
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													DBG_TIM8_STOP	DBG_TIM1_STOP	
													rw	rw	

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **DBG\_TIMx\_STOP**: TIMx counter stopped when core is halted (x=9..11)  
 0: The clock of the involved timer counter is fed even if the core is halted  
 1: The clock of the involved timer counter is stopped and the outputs are disabled when the core is halted



Bits 15: Reserved, must be kept at reset value.

Bit 1 **DBG\_TIM8\_STOP**: TIM8 counter stopped when core is halted

0: The clock of the involved timer counter is fed even if the core is halted

1: The clock of the involved timer counter is stopped and the outputs are disabled when the core is halted

Bit 0 **DBG\_TIM1\_STOP**: TIM1 counter stopped when core is halted

0: The clock of the involved timer counter is fed even if the core is halted

1: The clock of the involved timer counter is stopped and the outputs are disabled when the core is halted

## 38.17 TPIU (trace port interface unit)

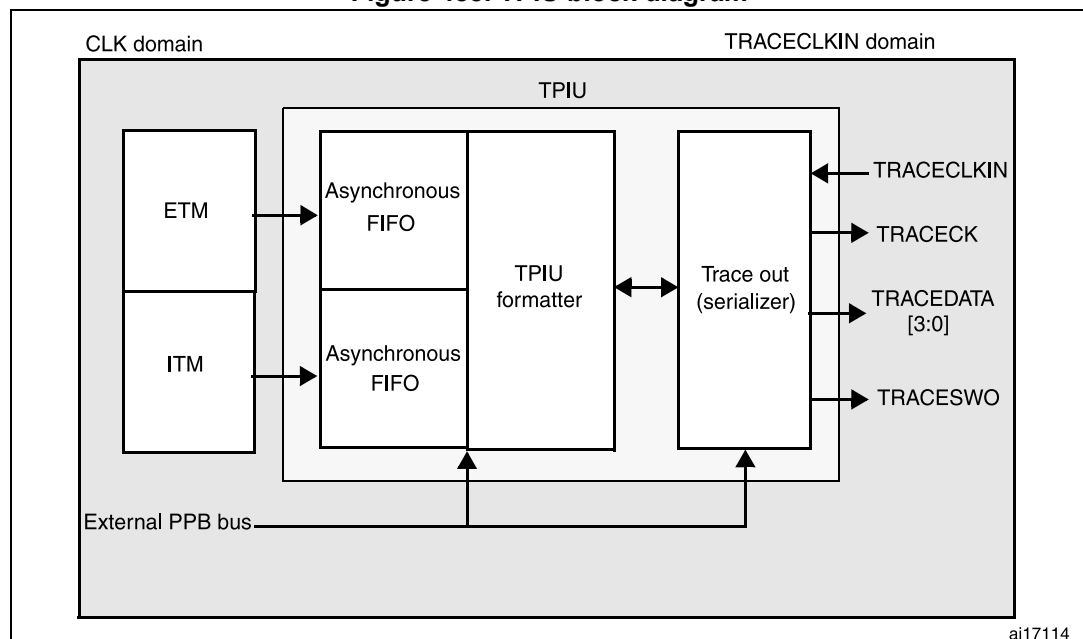
### 38.17.1 Introduction

The TPIU acts as a bridge between the on-chip trace data from the ITM and the ETM.

The output data stream encapsulates the trace source ID, that is then captured by a *trace port analyzer* (TPA).

The core embeds a simple TPIU, especially designed for low-cost debug (consisting of a special version of the CoreSight TPIU).

**Figure 488. TPIU block diagram**



### 38.17.2 TRACE pin assignment

- **Asynchronous mode**  
The asynchronous mode requires 1 extra pin and is available on all packages. It is only available if using Serial Wire mode (not in JTAG mode).

**Table 311. Asynchronous TRACE pin assignment**

TPUI pin name	Trace synchronous mode		STM32F4xx pin assignment
	Type	Description	
TRACESWO	O	TRACE Async Data Output	PB3

- **Synchronous mode**  
The synchronous mode requires from 2 to 6 extra pins depending on the data trace size and is only available in the larger packages. In addition it is available in JTAG mode and in Serial Wire mode and provides better bandwidth output capabilities than asynchronous trace.

**Table 312. Synchronous TRACE pin assignment**

TPUI pin name	Trace synchronous mode		STM32F4xxpin assignment
	Type	Description	
TRACECK	O	TRACE Clock	PE2
TRACED[3:0]	O	TRACE Sync Data Outputs Can be 1, 2 or 4.	PE[6:3]

#### TPUI TRACE pin assignment

By default, these pins are NOT assigned. They can be assigned by setting the TRACE\_IOEN and TRACE\_MODE bits in the **MCU Debug component configuration register**. This configuration has to be done by the debugger host.

In addition, the number of pins to assign depends on the trace configuration (asynchronous or synchronous).

- **Asynchronous mode:** 1 extra pin is needed
- **Synchronous mode:** from 2 to 5 extra pins are needed depending on the size of the data trace port register (1, 2 or 4):
  - TRACECK
  - TRACED(0) if port size is configured to 1, 2 or 4
  - TRACED(1) if port size is configured to 2 or 4
  - TRACED(2) if port size is configured to 4
  - TRACED(3) if port size is configured to 4

To assign the TRACE pin, the debugger host must program the bits TRACE\_IOEN and TRACE\_MODE[1:0] of the Debug MCU configuration register (DBGMCU\_CR). By default the TRACE pins are not assigned.

This register is mapped on the external PPB and is reset by the PORESET (and not by the SYSTEM reset). It can be written by the debugger under SYSTEM reset.

Table 313. Flexible TRACE pin assignment

DBGMCU_CR register		Pins assigned for:	TRACE IO pin assigned					
TRACE_IOEN	TRACE_MODE [1:0]		PB3 /JTDO/ TRACESWO	PE2/ TRACECK	PE3 / TRACED[0]	PE4 / TRACED[1]	PE5 / TRACED[2]	PE6 / TRACED[3]
0	XX	No Trace (default state)	Released <sup>(1)</sup>	-				
1	00	Asynchronous Trace	TRACESWO	-	-	Released (usable as GPIO)		
1	01	Synchronous Trace 1 bit	Released <sup>(1)</sup>	TRACECK	TRACED[0]	-	-	-
1	10	Synchronous Trace 2 bit		TRACECK	TRACED[0]	TRACED[1]	-	-
1	11	Synchronous Trace 4 bit		TRACECK	TRACED[0]	TRACED[1]	TRACED[2]	TRACED[3]

1. When Serial Wire mode is used, it is released. But when JTAG is used, it is assigned to JTDO.

**Note:** By default, the TRACECLKIN input clock of the TPIU is tied to GND. It is assigned to HCLK two clock cycles after the bit TRACE\_IOEN has been set.

The debugger must then program the Trace Mode by writing the PROTOCOL[1:0] bits in the SPP\_R (Selected Pin Protocol) register of the TPIU.

- PROTOCOL=00: Trace Port Mode (synchronous)
- PROTOCOL=01 or 10: Serial Wire (Manchester or NRZ) Mode (asynchronous mode). Default state is 01

It then also configures the TRACE port size by writing the bits [3:0] in the CPSPS\_R (Current Sync Port Size register) of the TPIU:

- 0x1 for 1 pin (default state)
- 0x2 for 2 pins
- 0x8 for 4 pins

### 38.17.3 TPUI formatter

The formatter protocol outputs data in 16-byte frames:

- seven bytes of data
- eight bytes of mixed-use bytes consisting of:
  - 1 bit (LSB) to indicate it is a DATA byte ('0') or an ID byte ('1').
  - 7 bits (MSB) which can be data or change of source ID trace.
- one byte of auxiliary bits where each bit corresponds to one of the eight mixed-use bytes:
  - if the corresponding byte was a data, this bit gives bit0 of the data.
  - if the corresponding byte was an ID change, this bit indicates when that ID change takes effect.

*Note: Refer to the Arm® CoreSight Architecture Specification v1.0 (Arm® IHI 0029B) for further information*

### 38.17.4 TPUI frame synchronization packets

The TPUI can generate two types of synchronization packets:

- The Frame Synchronization packet (or Full Word Synchronization packet)  
It consists of the word: 0x7F\_FF\_FF\_FF (LSB emitted first). This sequence can not occur at any other time provided that the ID source code 0x7F has not been used.  
It is output periodically **between** frames.  
In continuous mode, the TPA must discard all these frames once a synchronization frame has been found.
- The Half-Word Synchronization packet  
It consists of the half word: 0x7F\_FF (LSB emitted first).  
It is output periodically **between or within** frames.  
These packets are only generated in continuous mode and enable the TPA to detect that the TRACE port is in IDLE mode (no TRACE to be captured). When detected by the TPA, it must be discarded.

### 38.17.5 Transmission of the synchronization frame packet

There is no Synchronization Counter register implemented in the TPIU of the core. Consequently, the synchronization trigger can only be generated by the **DWT**. Refer to the registers DWT Control register (bits SYNCTAP[11:10]) and the DWT Current PC Sampler Cycle Count register.

The TPUI Frame synchronization packet (0x7F\_FF\_FF\_FF) is emitted:

- after each TPIU reset release. This reset is synchronously released with the rising edge of the TRACECLKIN clock. This means that this packet is transmitted when the TRACE\_IOEN bit in the DBGMCU\_CFG register is set. In this case, the word 0x7F\_FF\_FF\_FF is not followed by any formatted packet.
- at each DWT trigger (assuming DWT has been previously configured). Two cases occur:
  - If the bit SYNENA of the ITM is reset, only the word 0x7F\_FF\_FF\_FF is emitted without any formatted stream which follows.
  - If the bit SYNENA of the ITM is set, then the ITM synchronization packets follow (0x80\_00\_00\_00\_00\_00), formatted by the TPUI (trace source ID added).

### 38.17.6 Synchronous mode

The trace data output size can be configured to 4, 2 or 1 pin: TRACED(3:0)

The output clock is output to the debugger (TRACECK)

Here, TRACECLKIN is driven internally and is connected to HCLK only when TRACE is used.

*Note: In this synchronous mode, it is not required to provide a stable clock frequency.*

The TRACE I/Os (including TRACECK) are driven by the rising edge of TRACCLKIN (equal to HCLK). Consequently, the output frequency of TRACECK is equal to HCLK/2.

### 38.17.7 Asynchronous mode

This is a low cost alternative to output the trace using only 1 pin: this is the asynchronous output pin TRACESWO. Obviously there is a limited bandwidth.

TRACESWO is multiplexed with JTDO when using the SW-DP pin. This way, this functionality is available in all STM32F4xx packages.

This asynchronous mode requires a constant frequency for TRACECLKIN. For the standard UART (NRZ) capture mechanism, 5% accuracy is needed. The Manchester encoded version is tolerant up to 10%.

### 38.17.8 TRACECLKIN connection inside the STM32F4xx

In the STM32F4xx, this TRACECLKIN input is internally connected to HCLK. This means that when in asynchronous trace mode, the application is restricted to use to time frames where the CPU frequency is stable.

**Note:** ***Important:** when using asynchronous trace: it is important to be aware that:*

*The default clock of the STM32F4xx MCUs is the internal RC oscillator. Its frequency under reset is different from the one after reset release. This is because the RC calibration is the default one under system reset and is updated at each system reset release.*

*Consequently, the trace port analyzer (TPA) must not enable the trace (with the TRACE\_IOEN bit) under system reset, because a Synchronization Frame Packet is issued with a different bit time than trace packets which are transmitted after reset release.*

### 38.17.9 TPIU registers

The TPIU APB registers can be read and written only if the bit TRCENA of the Debug Exception and Monitor Control register (DEMCR) is set. Otherwise, the registers are read as zero (the output of this bit enables the PCLK of the TPIU).

**Table 314. Important TPIU registers**

Address	Register	Description
0xE0040004	Current port size	Allows the trace port size to be selected: Bit 0: Port size = 1 Bit 1: Port size = 2 Bit 2: Port size = 3, not supported Bit 3: Port Size = 4 Only 1 bit must be set. By default, the port size is one bit. (0x00000001)
0xE00400F0	Selected pin protocol	Allows the Trace Port Protocol to be selected: Bit1:0= 00: Sync Trace Port Mode 01: Serial Wire Output - manchester (default value) 10: Serial Wire Output - NRZ 11: reserved

Table 314. Important TPIU registers (continued)

Address	Register	Description
0xE0040304	Formatter and flush control	<p>Bits 31-9 = always '0            Bit 8 = TrgIn = always '1 to indicate that triggers are indicated            Bits 7-4 = always 0            Bits 3-2 = always 0            Bit 1 = EnFCont. In Sync Trace mode (Select_Pin_Protocol register bit1:0=00), this bit is forced to '1: the formatter is automatically enabled in continuous mode. In asynchronous mode (Select_Pin_Protocol register bit1:0 &lt;&gt; 00), this bit can be written to activate or not the formatter.            Bit 0 = always 0</p> <p>The resulting default value is 0x102</p> <p><b>Note:</b> In synchronous mode, because the TRACECTL pin is not mapped outside the chip, the formatter is always enabled in continuous mode -this way the formatter inserts some control packets to identify the source of the trace packets).</p>
0xE0040300	Formatter and flush status	Not used in Cortex <sup>®</sup> -M4 with FPU, always read as 0x00000008

### 38.17.10 Example of configuration

- Set the bit TRCENA in the Debug Exception and Monitor Control register (DEMCR)
- Write the TPIU Current Port Size register to the desired value (default is 0x1 for a 1-bit port size)
- Write TPIU Formatter and Flush Control register to 0x102 (default value)
- Write the TPIU Select Pin Protocol to select the sync or async mode. Example: 0x2 for async NRZ mode (UART like)
- Write the DBGMCU control register to 0x20 (bit IO\_TRACEN) to assign TRACE I/Os for async mode. A TPIU Sync packet is emitted at this time (FF\_FF\_FF\_7F)
- Configure the ITM and write the ITM Stimulus register to output a value

## 38.18 DBG register map

The following table summarizes the Debug registers.

**Table 315. DBG register map and reset values**

Addr.	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																
0xE004 2000	DBGMCU_IDCODE	REV_ID																Reserved		DEV_ID																													
	Reset value <sup>(1)</sup>	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X																																
0xE004 2004	DBGMCU_CR	Reserved												DBG_TIM7_STOP				Reserved												TRACE_MODE[1:0]				TRACE_IOEN		Reserved													
	Reset value													0	0	0	0													0	0				0					0	0								
0xE004 2008	DBGMCU_APB1_FZ	Reserved				DBG_CAN2_STOP		DBG_CAN1_STOP		Reserved		Reserved												DBG_IWDG_STOP		DBG_WWDG_STOP		Reserved		DBG_RTC_STOP		DBG_TIM14_STOP		DBG_TIM13_STOP		DBG_TIM12_STOP		DBG_TIM7_STOP		DBG_TIM6_STOP		DBG_TIM5_STOP		DBG_TIM4_STOP		DBG_TIM3_STOP		DBG_TIM2_STOP	
	Reset value					0	0	0	0	0	0													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xE004 200C	DBGMCU_APB2_FZ	Reserved												DBG_TIM11_STOP				DBG_TIM10_STOP				DBG_TIM9_STOP				Reserved												DBG_TIM8_STOP				DBG_TIM1_STOP							
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0													0	0	0	0	0	0	0	0	0	0	0	0

1. The reset value is product dependent. For more information, refer to [Section 38.6.1: MCU device ID code](#).