

31 Debug support (DBG)

31.1 Overview

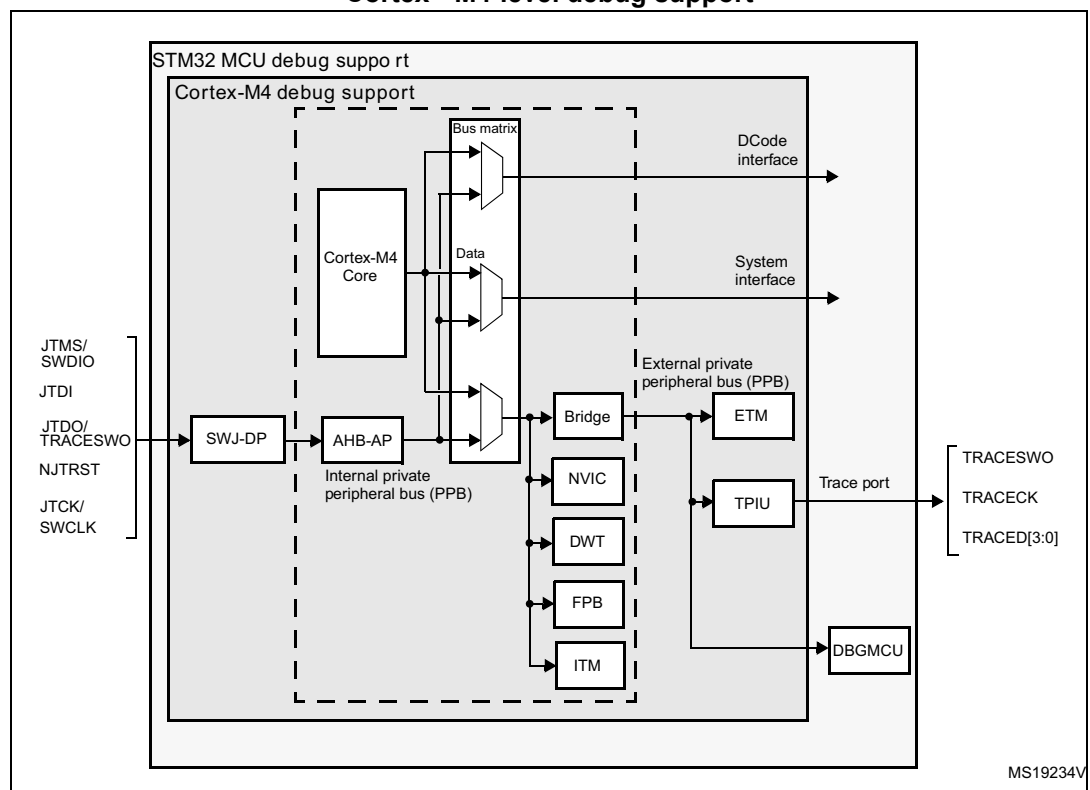
The STM32F334xx devices are built around a Cortex®-M4 core which contains hardware extensions for advanced debugging features. The debug extensions allow the core to be stopped either on a given instruction fetch (breakpoint) or data access (watchpoint). When stopped, the core's internal state and the system's external state may be examined. Once examination is complete, the core and the system may be restored and program execution resumed.

The debug features are used by the debugger host when connecting to and debugging the STM32F334xx MCUs.

Two interfaces for debug are available:

- Serial wire
- JTAG debug port

Figure 411. Block diagram of STM32 MCU and Cortex®-M4-level debug support



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

The Cortex®-M4 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)

It also includes debug features dedicated to the STM32F334xx:

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

Note: For further information on the debug feature supported by the Cortex®-M4 core, refer to the Cortex®-M4 with FPU-r0p1 Technical Reference Manual and to the CoreSight Design Kit-r0p1 TRM (see [Section 31.2: Reference Arm documentation](#)).

31.2 Reference Arm documentation

- Cortex®-M4 r0p1 Technical Reference Manual (TRM)
It is available from: <http://infocenter.arm.com>
- Arm Debug Interface V5
- Arm CoreSight Design Kit revision r0p1 Technical Reference Manual

31.3 SWJ debug port (serial wire and JTAG)

The STM32F334xx core 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 412. SWJ debug port

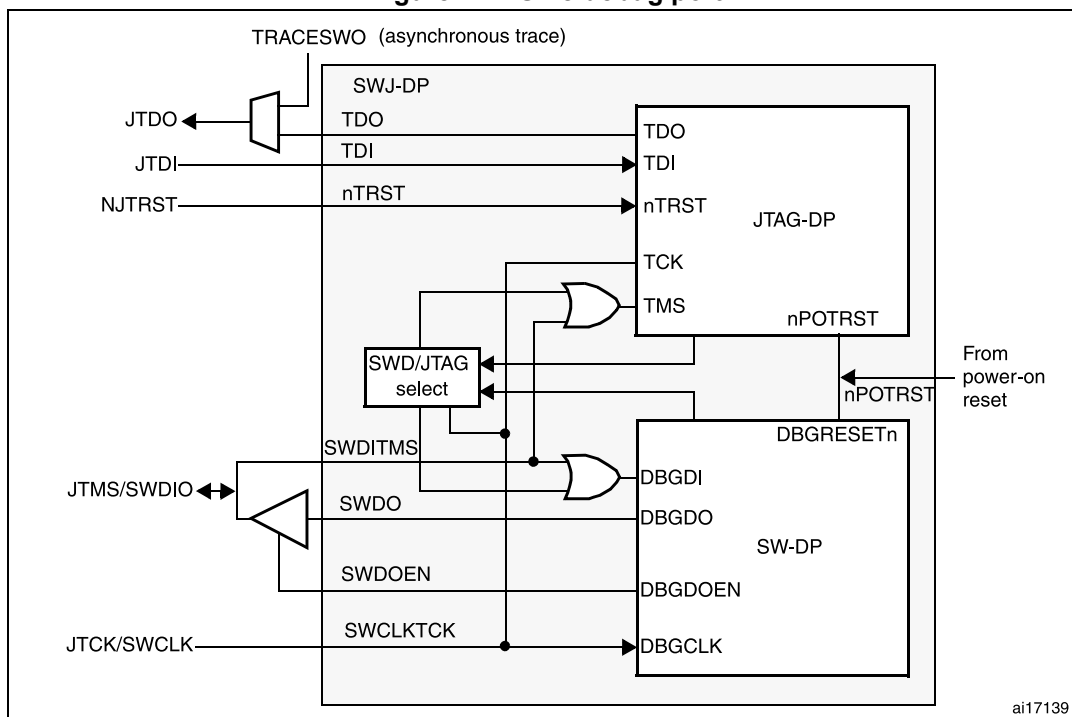


Figure 412 shows that the asynchronous TRACE output (TRACESWO) is multiplexed with TDO. This means that the asynchronous trace can only be used with SW-DP, not JTAG-DP.

31.3.1 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

31.4 Pinout and debug port pins

The STM32F334xx 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.

31.4.1 SWJ debug port pins

Five pins are used as outputs from the STM32F334xx for the SWJ-DP as *alternate functions* of general-purpose I/Os. These pins are available on all packages.

Table 150. 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

31.4.2 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, it is possible to disable some or all of the SWJ-DP ports and so, to release (in gray in the table below) the associated pins for general-purpose I/O(GPIO) usage. For more details on how to disable SWJ-DP port pins, please refer to [Section 9.3.2: I/O pin alternate function multiplexer and mapping](#).

Table 151. 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

Note:

When the APB bridge write buffer is full, it takes one extra APB cycle when writing the AFIO_MAPR register. This is because the deactivation of the JTAGSW pins is done in two cycles to guarantee a clean level on the nTRST and TCK input signals of the core.

- Cycle 1: the JTAGSW input signals to the core are tied to 1 or 0 (to 1 for nTRST, TDI and TMS, to 0 for TCK)
- Cycle 2: the GPIO controller takes the control signals of the SWJTAG IO pins (like controls of direction, pull-up/down, Schmitt trigger activation, etc.).

31.4.3 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: Input pull-up
- JTDI: Input pull-up
- JTMS/SWDIO: Input pull-up
- JTCK/SWCLK: Input pull-down
- JTDO: Input 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.

31.4.4 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.

31.5 STM32F334xx JTAG TAP connection

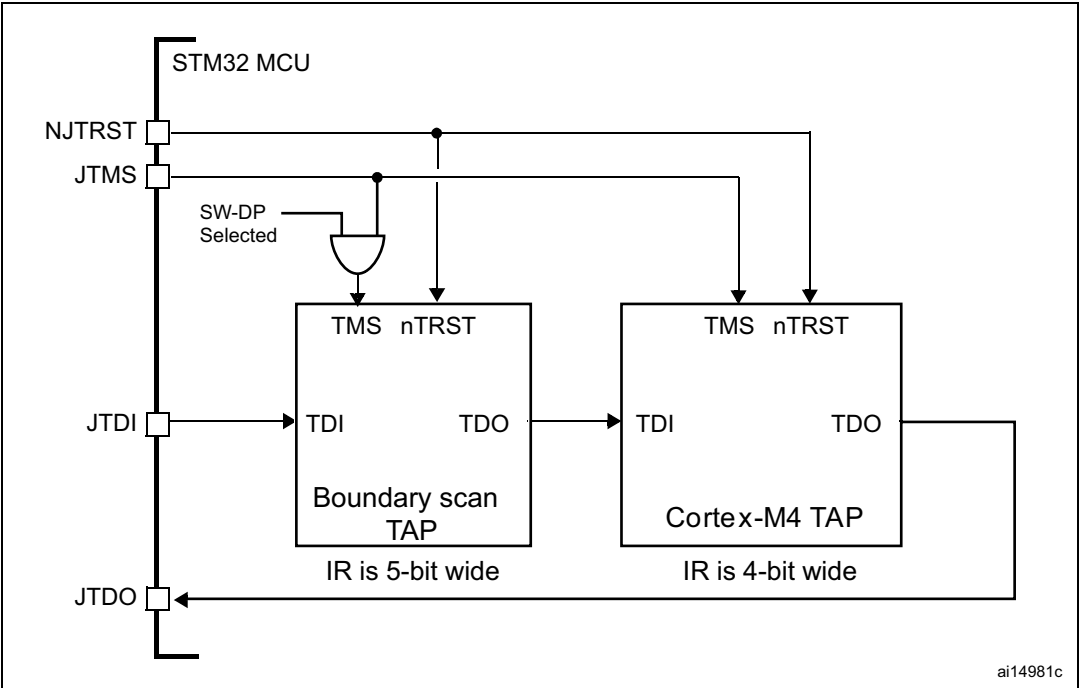
The STM32F334xx MCUs integrate two serially connected JTAG TAPs, the boundary scan TAP (IR is 5-bit wide) and the Cortex®-M4 TAP (IR is 4-bit wide).

To access the TAP of the Cortex®-M4 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 413. JTAG TAP connections



31.6 ID codes and locking mechanism

There are several ID codes inside the STM32F334xx 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.

31.6.1 MCU device ID code

The STM32F334xx 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 31.15 on page 1111](#)). This code is accessible using the JTAG debug port (4 to 5 pins) or the SW debug port (two pins) or by the user software. It is even accessible while the MCU is under system reset.

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															
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
Res	Res	Res	Res	DEV_ID											
				r	r	r	r	r	r	r	r	r	r	r	r

This code is read as 0x10000438 for Revision 1.0.

Bits 31:16 **REV_ID[15:0]** Revision identifier

This field indicates the revision of the device. For example, it is read as 0x1000 for Revision 1.

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

Bits 11:0 **DEV_ID[11:0]**: Device identifier

This field indicates the device and its revision.
The device ID is 0x438.

31.6.2 Boundary scan TAP

JTAG ID code

The TAP of the STM32F334xx BSC (boundary scan) integrates a JTAG ID code equal to 0x06432041.

31.6.3 Cortex[®]-M4 TAP

The TAP of the Cortex[®]-M4 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.

This code is **0x4BA00477** (corresponds to Cortex[®]-M4 r0p1, see [Section 31.2: Reference Arm documentation](#)).

Only the DEV_ID(11:0) should be used for identification by the debugger/programmer tools.

31.6.4 Cortex[®]-M4 JEDEC-106 ID code

The Cortex[®]-M4 integrates a JEDEC-106 ID code. It is located in the 4KB 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.

31.7 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[®]-M4r0p1 *Technical Reference Manual (TRM)*, for references, please see [Section 31.2: Reference Arm documentation](#)).

Table 152. JTAG debug port data registers

IR(3:0)	Data register	Details
1111	BYPASS [1 bit]	
1110	IDCODE [32 bits]	ID CODE 0x3BA00477 (Cortex [®] -M4 r0p1 ID Code)

Table 152. JTAG debug port data registers (continued)

IR(3:0)	Data register	Details
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"> When transferring data IN: <ul style="list-style-type: none"> Bits 34:3 = DATA[31:0] = 32-bit data to transfer for a write request Bits 2:1 = A[3:2] = 2-bit address of a debug port register. Bit 0 = RnW = Read request (1) or write request (0). When transferring data OUT: <ul style="list-style-type: none"> Bits 34:3 = DATA[31:0] = 32-bit data which is read following a read request Bits 2:0 = ACK[2:0] = 3-bit Acknowledge: <ul style="list-style-type: none"> 010 = OK/FAULT 001 = WAIT OTHER = reserved <p>Refer to Table 153 for a description of the A(3:2) bits</p>
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"> When transferring data IN: <ul style="list-style-type: none"> Bits 34:3 = DATA[31:0] = 32-bit data to shift in for a write request Bits 2:1 = A[3:2] = 2-bit address (sub-address AP registers). Bit 0 = RnW = Read request (1) or write request (0). When transferring data OUT: <ul style="list-style-type: none"> Bits 34:3 = DATA[31:0] = 32-bit data which is read following a read request Bits 2:0 = ACK[2:0] = 3-bit Acknowledge: <ul style="list-style-type: none"> 010 = OK/FAULT 001 = WAIT OTHER = reserved <p>There are many AP Registers (see AHB-AP) addressed as the combination of:</p> <ul style="list-style-type: none"> The shifted value A[3:2] The current value of the DP SELECT register
1000	ABORT [35 bits]	<p>Abort register</p> <ul style="list-style-type: none"> Bits 31:1 = Reserved Bit 0 = DAPABORT: write 1 to generate a DAP abort.

Table 153. 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"> Request a system or debug power-up Configure the transfer operation for AP accesses Control the pushed compare and pushed verify operations. Read some status flags (overrun, power-up acknowledges)

Table 153. 32-bit debug port registers addressed through the shifted value A[3:2] (continued)

Address	A(3:2) value	Description
0x8	10	DP SELECT register: Used to select the current access port and the active 4-words register window. – Bits 31:24: APSEL: select the current AP – Bits 23:8: reserved – Bits 7:4: APBANKSEL: select the active 4-words register window on the current AP – Bits 3:0: reserved
0xC	11	DP RDBUFF register: Used to allow the debugger to get the final result after a sequence of operations (without requesting new JTAG-DP operation)

31.8 SW debug port

31.8.1 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 kΩ recommended by Arm).

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.

31.8.2 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 154. 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

Table 154. Packet request (8-bits) (continued)

Bit	Name	Description
4:3	A(3:2)	Address field of the DP or AP registers (refer to Table 153)
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[®]-M4 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 155. 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 156. 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.

31.8.3 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 **0x1BA01477** (corresponding to Cortex[®]-M4 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 r0p1 TRM* and the *CoreSight Design Kit r0p1 TRM*.

31.8.4 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 should 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.

31.8.5 SW-DP registers

Access to these registers are initiated when APnDP=0

Table 157. 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 0x2BA01477 (identifies the SW-DP)
00	Write	-	ABORT	-
01	Read/Write	0	DP-CTRL/STAT	Purpose is to: <ul style="list-style-type: none"> – request a system or debug power-up – configure the transfer operation for AP accesses – control the pushed compare and pushed verify operations. – read some status flags (overrun, power-up acknowledges)
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.

Table 157. SW-DP registers (continued)

A(3:2)	R/W	CTRLSEL bit of SELECT register	Register	Notes
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

31.8.6 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

31.9 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:

- g) Bits [7:4] = the bits [7:4] APBANKSEL of the DP SELECT register
- h) 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[®]-M4 includes 9 x 32-bits registers:

Table 158. Cortex®-M4 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®-M4 r0p1 TRM* for further details.

31.10 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 159. 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 TRCENA 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®-M4 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.

31.11 Capability of the debugger host to connect under system reset

The STM32F334xx MCUs' reset system 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 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 halt 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.

31.12 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.

31.13 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

31.14 ITM (instrumentation trace macrocell)

31.14.1 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[®]-M4 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.

31.14.2 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 160. 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 31.15.3: Debug MCU configuration register](#))
- 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)

31.15 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

31.15.1 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.

31.15.2 Debug support for timers, watchdog, bxCAN and I²C

During a breakpoint, it is necessary to choose how the counter of timers and watchdog should 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 I²C, the user can choose to block the SMBUS timeout during a breakpoint.

31.15.3 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

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
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	Res.	Res.	Res.	Res	DBG STAND BY	DBG STOP	DBG SLEEP	
												rw	rw	rw	

Bits 31:3 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).

31.15.4 Debug MCU APB1 freeze register (DBGMCU_APB1_FZ)

The DBGMCU_APB1_FZ register is used to configure the MCU under DEBUG. It concerns the APB1 peripherals:

- Timer clock counter freeze
- I2C SMBUS timeout freeze
- Window watchdog and independent watchdog counter freeze support

This DBGMCU_APB1_FZ is mapped on the external PPB bus at address 0xE0042008.

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-bit 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
Res	Res	Res	Res	Res	Res	DBG_CAN_STOP	Res	Res	Res	DBG_I2C1_SMBUS_TIMEOUT	Res	Res	Res	Res	Res
						rw				rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	DBG_IWDG_STOP	DBG_WWDG_STOP	DBG_RTC_STOP	Res	Res	Res	Res	DBG_TIM7_STOP	DBG_TIM6_STOP	Res	Res	DBG_TIM3_STOP	DBG_TIM2_STOP
			rw	rw	rw					rw	rw			rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **DBG_CAN_STOP**: Debug CAN stopped when core is halted

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

Bits 24:22 Reserved, must be kept at reset value.

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

Bits 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**: Debug RTC stopped when core is halted
 0: The clock of the RTC counter is fed even if the core is halted
 1: The clock of the RTC counter is stopped when the core is halted

Bits 9:6 Reserved, must be kept at reset value.

Bit 5 **DBG_TIM7_STOP**: TIM7 counter stopped when core is halted
 0: The counter clock of TIM7 is fed even if the core is halted
 1: The counter clock of TIM7 is stopped when the core is halted

Bit 4 **DBG_TIM6_STOP**: TIM6 counter stopped when core is halted
 0: The counter clock of TIM6 is fed even if the core is halted
 1: The counter clock of TIM6 is stopped when the core is halted

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **DBG_TIM3_STOP**: TIM3 counter stopped when core is halted
 0: The counter clock of TIM3 is fed even if the core is halted
 1: The counter clock of TIM3 is stopped when the core is halted

Bit 0 **DBG_TIM2_STOP**: TIM2 counter stopped when core is halted
 0: The counter clock of TIM2 is fed even if the core is halted
 1: The counter clock of TIM2 is stopped when the core is halted

31.15.5 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:

- Timer clock counter freeze

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
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	DBG_HRTIM1_STOP	Res	Res	Res	DBG_TIM17_STOP	DBG_TIM16_STOP	DBG_TIM15_STOP	Res	DBG_TIM1_STOP
							rw				rw	rw	rw		rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **DBG_HRTIM1_STOP**: HRTIM1 counter stopped when core is halted

0: The clock of the HRTIM1 timer counters is fed even if the core is halted

1: The clock of the HRTIM1 timer counters is stopped when the core is halted

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBG_TIMx_STOP**: TIMx counter stopped when core is halted (x=1, 8,15..17)

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 when the core is halted

Note: Bit1 is reserved.

31.16 TPIU (trace port interface unit)

31.16.1 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).

Table 161. 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 ⁽¹⁾	-				
1	00	Asynchronous Trace	TRACESWO	-	-	Released (usable as GPIO)		
1	01	Synchronous Trace 1 bit	Released ⁽¹⁾	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.

31.17 DBG register map

Table 162. 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	
0xE0042000	DBGMCU_IDCODE	REV_ID																Res	Res	Res	Res	DEV_ID												
	Reset value ⁽¹⁾	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X					X	X	X	X	X	X	X	X	X	X	X		
0xE0042004	DBGMCU_CR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	
	Reset value																														0	0	0	0
0xE004 2008	DBGMCU_APB1_FZ	Res	Res	Res	Res	Res	Res	DBG_CAN_STOP	Res	Res	Res	DBG_I2C1_SMBUS_TIMEOUT	Res	Res	Res	Res	Res	Res	Res	Res	Res	DBG_IWDG_STOP	DBG_WWDG_STOP	DBG_RTC_STOP	Res	Res	Res	DBG_TIM7_STOP	DBG_TIM6_STOP	Res	Res	Res	Res	Res
	Reset value							0				0										0	0	0				0	0			0	0	0
0xE004 200C	DBGMCU_APB2_FZ	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	DBG_HRTIM1_STOP	Res	Res	Res	Res	Res	DBG_TIM17_STOP	DBG_TIM16_STOP	DBG_TIM15_STOP	Res	DBG_TIM1_STOP	0
	Reset value																						0					0	0	0			0	

1. The reset value is product dependent. For more information, refer to [Section 31.6.1: MCU device ID code](#).