25 SWD debugging interface

This series uses ARM Cortex-M0+ core, which has hardware debug module SWD, which supports complex debugging

operate. The hardware debug module allows the core to stop when an instruction is fetched (instruction breakpoint) or data is accessed (data breakpoint). kernel halt

When it stops, both the internal state of the kernel and the external state of the system can be queried in the IDE. After completing the inquiry, the

The core and peripherals can be restored and the program will continue to execute. When the HC32L110 microcontroller is connected to the debugger and starts

When debugging, the debugger will use the kernel's hardware debug module for debugging operations.

Notice:

– SWD cannot work in DeepSleep mode, please perform debugging operation in Active and Sleep mode.

25.1 SWD debugging additional functions

This product uses an ARM Cortex-M0+ CPU, which contains hardware extensions for advanced debugging

The debugging function of this product is the same as that of Cortex-M0+. The debug extension allows the kernel to

breakpoint) or stop the kernel when accessing data (data breakpoint). When the kernel is stopped, you can query the internal state of the kernel

state and the external state of the system. After the query is complete, the kernel and system are restored and program execution resumes.

The debug function is used when the debug host is connected to the MCU and debugged.

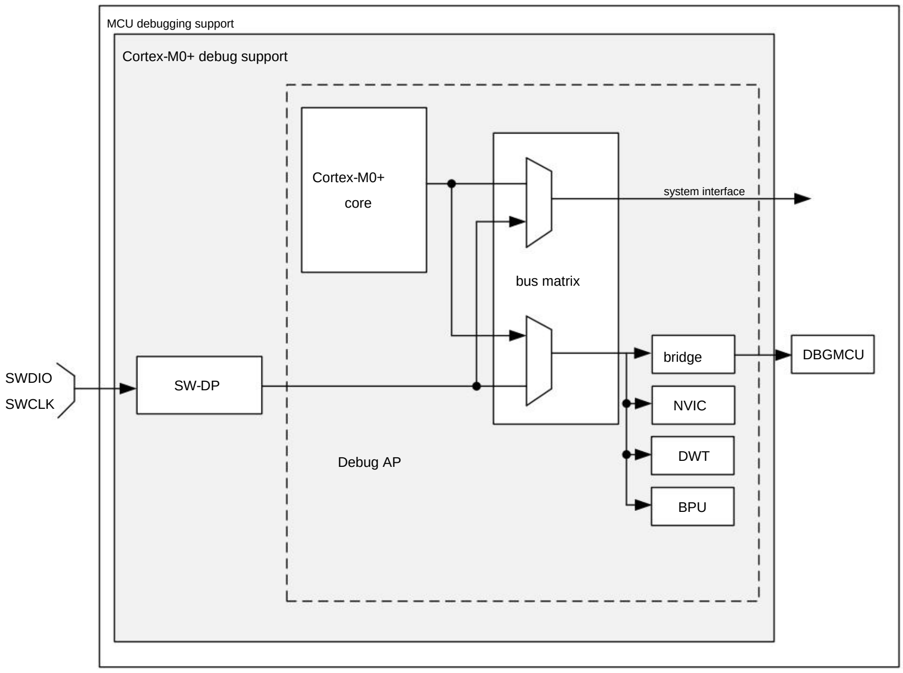

Figure 25-1 Debug Support Block Diagram

The debug capabilities built into the Cortex®-M0+ core are part of the ARM® CoreSight Design Suite.

The ARM® Cortex®-M0+ core provides integrated on-chip debug support. it includes:

ÿ SW-DP: Serial Line

ÿ BPU: Breakpoint Unit

ÿ DWT: Data Watch Point Trigger

Note:

– For details on the debug features supported by the ARM® Cortex®-M0+ core, see Cortex®-M0+

M0+ Technical Reference Manual.

25.2 ARM® Reference Documentation

ÿ Cortex®-M0+ Technical Reference Manual (TRM)

Available from www.infocenter.arm.com.

ÿ ARM® Debug Interface V5

ÿ ARM® CoreSight Design Suite Version r1p1 Technical Reference Manual

25.3 Debug port pins

25.3.1 SWD port pins

The SWD interface of this series needs to use 2 pins, as shown in the table below.

SWD port name	Debug function	pin assignment
SWCLK	serial clock	P31
SWDIO	Serial data input/output	P27

25.3.2 SW-DP pin assignment

If the [Encryption Chip] option is enabled when programming the program, the SWD debugging function will be disabled after power-on. If burned

If the [Encryption Chip] option is not enabled in the program, the P27/P31 pins are initialized to be debugged after power-on

dedicated pins used. The user can set the SYSCTRL1.SWD_USE_IO register to disable the SWD pin

Debug function, the SWD pin will be released for normal GPIO. The configuration and function of SWD pins are summarized in the following table

shown:

ÿEncryption ChipÿOption	SWD_USE_IO configuration	P27/P31 function
encryption	0	NA
encryption	1	GPIO
not encrypted	0	SWD
not encrypted	1	GPIO

25.3.3 INTERNAL PULL-UP ON SWD PIN

After the user software releases the SW I/O, the GPIO controller takes control of these pins. Reset of GPIO Control Registers

The state puts the I/O into the equivalent state:

ÿ SWDIO: input pull-up

ÿ SWCLK: input pull-up

No external resistors are required due to built-in pull-up and pull-down resistors.

25.4 SWD port

25.4.1 Introduction to SWD Protocol

This synchronous serial protocol uses two pins:

ÿ SWCLK: clock from host to target

ÿ SWDIO: Bidirectional

Using this protocol, two sets of register banks (DPACC register bank and APACC register bank) can be read and written simultaneously.

device group). When transferring data, the LSB comes first.

For SWDIO bidirectional management, the line must be pulled up on the board (ARM® recommends 100 kÿ).

These pull-up resistors are internally configurable. No external pull-up resistors are required.

Every time the direction of the SWDIO is changed in the protocol, the transition time is inserted and the line is not driven by the host

Nor is it goal driven. By default, this conversion time is one bit time, but can be set by configuring the SWCLK frequency

rate to adjust.

25.4.2 SWD protocol sequence

Each sequence consists of three phases:

1. Packet request sent by host (8 bits)

2. The acknowledgment response sent by the target (3 bits)

3. Phase of data transfer sent by host or target (33 bits)

bit name start		illustrate
0		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 DP or AP register
5	Parity Stop	Unit parity of first few digits
6	Resident	0
7		Not driven by the host. Must be read as 1 by target due to pull-up

See the Cortex®-M0+ TRM for a detailed description of the DPACC and APACC registers.

The packet request is always followed by a transition time (default 1 bit), at which point neither the host nor the target will drive.

bit name		illustrate
0	ACK	001: FAULT 010: WAIT 100: OK

MUST be followed by an ACK response only if a READ transaction occurs or a WAIT or FAULT acknowledgement is received

conversion time.

bit name		illustrate
0:31	WDATA or RDATA write	or read data
32 Parity		Single parity of 32 data bits

Conversion time must follow the DATA transfer only when a READ transaction occurs.

25.4.3 **SW-DP** state machine (reset, idle state, ID code)

The state machine of the SW-DP has an internal ID code for identifying the SW-DP. The code complies with the JEP-106 standard

allow. This ID code is the default ARM® code and is set to **0x0BB11477 (equivalent to** Cortex®-M0+).

Notice:

– The SW-DP state machine is inactive until the target reads this ID code.

ÿ After power-on reset or after the line is high for more than 50 cycles, the SW-DP state machine is in reset state.

state.

ÿ If the line is low for at least two cycles after the reset state, the SW-DP state machine is in the idle state.

ÿ After the reset state, the state machine must first enter the idle state, and then update the DP-SW ID CODE register

Perform read access. Otherwise, the target will issue a FAULT confirmation response on another transaction.

For more details on the SW-DP state machine, see *Cortex®-M0+ TRM* and *CoreSight* design

kit *r1p0TRM.*

25.4.4 **DP** and **AP** read/write access

ÿ Do not delay read access to DP: target response can be sent immediately (if ACK=OK), or delayed

Send target response (if ACK=WAIT).

ÿ Delay read access to AP. This means that the access result will be returned on the next transfer. If you want to execute the next

If this access is not an AP access, the DP-RDBUFF register must be read to get the result.

ÿ The DP-CTRL/STAT register is updated every time an AP read access or RDBUFF read request is made.

READOK flag to know if AP read access was successful.

ÿ SW-DP has write buffer (for DP or AP write), so that even when other operations are not completed,

Write operations are also accepted. If the write buffer is full, the target acknowledges the response with a WAIT. but IDCODE

Except for reads, CTRL/STAT reads, or ABORT writes, which also work when the write buffer is full

been accepted.

ÿ Due to the existence of asynchronous clock domains SWCLK and HCLK, after the write operation (after the parity bit), the

Two additional SWCLK cycles for writes to take effect internally. should be driven low on the line

These cycles are applied when (idle state).

This is especially important when writing to the CTRL/STAT register to initiate a power-up request. else next action

(Operations that are not valid until the kernel is powered on) are executed immediately, which will cause the failure.

25.4.5 SW-DP register

These registers can be accessed when APnDP=0:

A[3:2] RW		**SELECT** register The **CTRLSEL** bit	register	Notes
00	read		IDCODE	Manufacturer code set to default ARM® for Cortex®-M0+ code. **0x0BB11477 (identifies** SW-DP)
00	Write		ABORT	
01	read /write	0	DP CTRL/STAT	Purpose: – Request system or debug power-up – Configure transport operations for AP access – Control compare and verify operations – read some status flags (overflow and power-up acknowledgment)
01	read /write	1	WIRE CONTROL	Used to configure the physical serial port protocol (such as the duration of the transition time time)
10	read		READ RESEND	Allows recovery of read data from corrupted debug software transfers, No need to repeat the original AP transmission.
10	Write		SELECT	4-word register window for selecting current access port and activity
11	read /write		READ BUFFER	This read buffer is useful since AP access has been issued (in The result of the read AP request is provided when the next AP transaction is performed). This read buffer captures the data in the AP and appears as

The result of the previous read, without starting a new operation.

25.4.6 SW-AP register

These registers can be accessed when APnDP=1:

There are multiple AP registers, which are addressed in the following combinations:

ÿ Shift value A[3:2]

ÿ Current value of DP SELECT register

address	A[3:2] Value Description	
0x0	00	Reserved, the reset value must be maintained.
0x4	01	DP CTRL/STAT register. Used for: – Request system or debug power-up – Configure transport operations for AP access – Control compare and verify operations – read some status flags (overflow and power-up acknowledgment)
0x8	10	DP SELECT register: 4-word register window for selecting the current access port and activity mouth. – Bit 31:24: APSEL: select the current AP – Bits 23:8: Reserved – Bits 7:4: APBANKSEL: Select the active 4-word register window on the current AP – Bits 3:0: reserved
0xC	11	DP RDBUFF register: used by the debugger to get the last result after a series of operations fruit

(without requesting a new JTAG-DP operation)

25.5 Kernel Debugging

Debug the kernel through the kernel debug registers. Debug access to these registers is done through the debug access port. it is sent by four

Register composition:

Register	Description
DHCSR	*32* -bit debug stop control and status register This register provides information about the state of the processor, enables the core to enter a debug stop state and provides processor stepping capabilities.
DCRSR	*17* -bit Debug Core Register Selector Register: This register selects the processor registers that need to be read and written.
DCRDR	*32* -bit debug core register data register: This register holds the data read and written between the register and the processor selected by the DCRSR (selector) register.
DEMCR	*32* -bit Debug Exception and Watch Control Register: This register provides vector capture and debug watch control.

These registers are not reset upon system reset. They can only be reset by a power-on reset. For more details,

See Cortex®-M0+ TRM.

In order to put the kernel into debug halted state immediately after reset, it is necessary to:

ÿ Enable debug and exception monitoring control register bit 0 (VC_CORRESET)

ÿ Enable Debug Stop Control and Status Register Bit 0 (C_DEBUGEN)

25.6 BPU (Breakpoint Unit)

The Cortex®-M0+ BPU implementation provides four breakpoint registers.

25.6.1 BPU function

Processor breakpoints implement PC-based breakpoint functionality.

For more information on the BPU CoreSight identification registers and their addresses and access types, see ARMv6-M

ARM® and ARM® CoreSight Components Technical Reference Manual.

25.7 DWT (Data Observation Point)

The Cortex®-M0+ DWT implementation provides two watchpoint register sets.

25.7.1 DWT function

Processor watchpoints implement data address and PC-based watchpoint functionality (ie, PC sample registers), and support

Comparator address mask, as described in ARMv6-M ARM®.

25.7.2 DWT Program Counter Sample Register

The processor implementing the data watchpoint unit also implements the ARMv6-M optional DWT program counter sample register

controller (DWT_PCSR). This register allows the debugger to periodically sample the PC without stopping the processor. This provides rough

Brief analysis. See ARMv6-M ARM® for more information.

Cortex®-M0+ DWT_PCSR records passing condition codes and instructions and instructions that fail condition codes.

25.8 MCU Debug Component (DBG)

The MCU debug component helps the debugger provide support for:

ÿ Low power mode

ÿ Timer and watchdog clock control during breakpoints

25.8.1 Debug Support for Low Power Modes

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

The MCU supports several low-power modes that disable the CPU clock or reduce CPU power consumption.

The kernel does not allow FCLK or HCLK to be turned off during a debug session. Since they need to be used during debugging

Debug connection, so it must remain active. The MCU integrates special methods that allow the user to

Download the debugging software.

25.8.2 Debugging support for timers and watchdogs

During a breakpoint, you must choose how the timers and watchdog's counters behave:

ÿ When a breakpoint occurs, the counter continues to count. For example, this approach is often required when PWM controls a motor

Mode.

ÿ When a breakpoint occurs, the counter stops counting. This method is required for watchdog use.

25.9 Debug mode module working state control (DEBUG_ACTIVE)

Reset value 0x00000FFF (this register setting only works in SWD debug mode)

Offset address: 0x038

31	30 29	28	27	26	25	24 23	twenty two	twenty one	20	19	18	17	16
Reserved													

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				LPTIM	Res.	RTC	WDT	PCA	TIM6	TIM5	TIM4	LPTIM	TIM2	TIM1	TIM0
				RW		RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

bit	mark	Function description
31:12	Reserved	reserve
11	LPTIM	When debugging, Timer3 count function configuration 1: In the SWD debugging interface, suspend the Timer3 counting function 0: In the SWD debugging interface, Timer3 normal counting function
10	Reserved	reserve
9	RTC	When debugging, the RTC count function is configured 1: In the SWD debugging interface, pause the RTC counting function 0: In the SWD debugging interface, the RTC normal counting function
8	WDT	When debugging, the WDT count function is configured 1: In the SWD debugging interface, suspend the WDT counting function 0: In the SWD debugging interface, the WDT normal counting function
7	PCA	When debugging, the PCA count function is configured 1: In the SWD debugging interface, suspend the PCA counting function 0: In the SWD debugging interface, PCA normal counting function
6	TIM6	When debugging, Timer6 count function configuration 1: In the SWD debugging interface, pause the Timer counting function 0: In the SWD debugging interface, the Timer normal counting function
5	TIM5	When debugging, Timer5 count function configuration 1: In the SWD debugging interface, pause the Timer counting function 0: In the SWD debugging interface, the Timer normal counting function
4	TIM4	When debugging, Timer4 count function configuration 1: In the SWD debugging interface, pause the Timer counting function 0: In the SWD debug interface, Timer normal counting function
3	LPTIM	When debugging, the LpTimer count function is configured 1: In the SWD debugging interface, pause the Timer counting function 0: In the SWD debug interface, Timer normal counting function
2	TIM2	When debugging, Timer2 count function configuration 1: In the SWD debugging interface, pause the Timer counting function

		0: In the SWD debugging interface, the Timer normal counting function
1	TIM1	When debugging, Timer1 count function configuration 1: In the SWD debugging interface, pause the Timer counting function 0: In the SWD debugging interface, the Timer normal counting function
0	TIM0	When debugging, Timer0 count function configuration 1: In the SWD debugging interface, pause the Timer counting function 0: In the SWD debugging interface, the Timer normal counting function

26 Devices Electronic Signature

The electronic signature is stored in the system memory area of the flash memory module and can be read by SWD or CPU. it

The included chip identification information is written at the factory, and the user firmware or external device can read the electronic signature to

Automatically match different configurations of HC32Fxxx / HC32Lxxx microcontrollers.

26.1 Product Unique Identification (UID) Register (80 -bit)

Typical application scenarios of unique identifiers:

ÿ Used as serial number

ÿ UID is used as security when used in conjunction with software cryptographic primitives and protocols before programming the internal Flash

Keys to improve the security of code in Flash

ÿ Activate safe bootstrap process, etc.

The 80-bit Unique Device Identifier provides a reference number that is unique to any device and any context. use

The user can never change these bits. The 80-bit unique device identifier can also be in different formats such as single-byte/half-word/word

Read, then concatenate using a custom algorithm.

Base address: 0x0010 0E74

Offset address	description	UID Bit(80bit)
		7 6 5 4 3 2 1 0
0	X Coordinate on the wafer	UID[7:0]
1	Rev ID	UID[15:8]
2	Fixed value FFH	Reserved
3	Fixed value FFH	Reserved
4	Fixed value 00H	UID[23:16]
5	Fixed value 00H	UID[31:24]
6	Wafer Number	UID[39:32]
7	Y Coordinate on the wafer	UID[47:40]
8	Wafer Lot Number	UID[55:48]
9		UID[63:56]
10		UID[71:64]
11		UID[79:72]

26.2 Product model register

0x0010 0C60 ~ 0x0010 0C6F Stores the ASCII code of the product model. If the product model is less than 16 bytes,

It is filled with 0x00.

Example: 484333324C3133364B38544100000000 represents the product model HC32L136K8TA.

26.3 FLASH Capacity Register

Base address: 0x0010 0C70

31	30	29	28	27	26	25	twenty four	twenty three	twenty two	twenty one	20	19	18	17	16
FlashSize[31:16]															
R															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FlashSize[15:0]															
R															

Bit	tag	function description
31:0	FlashSize	The capacity of the built-in Flash of the product, in bytes 0x00008000 means the Flash capacity is 32K Byte

26.4 RAM Size Register

Base address: 0x0010 0C74

31	30	29	28	27	26	25	twenty four	twenty three	twenty two	twenty one	20	19	18	17	16
RamSize[31:16]															
R															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RamSize[15:0]															
R															

Bit tag function description		
31:0	RamSize	Product's built-in RAM capacity, in bytes 0x00000800 means the RAM capacity is 2K Byte

26.5 Pin Count Register

Base address: 0x0010 0C7A

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PinCount[15:0]															
R															

Bit tag function description		
15:0	PinCount	The number of product pins, in units 0x0020 means the number of product pins is 32