ARM922T™ with AHB

System-on-Chip Platform OS Processor

Product Overview

ARM®

DVI0025B

ARM922T with AHB
Product Overview

Release Information

The following changes have been made to this book.

Change History

Date	Issue	Confidentiality	Change
19 Jan 2001	B	Non-Confidential	Original issue
22 Jun 2007	B	Non-Confidential	Reformatted for InfoCenter, no change to contents.

Proprietary Notice

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by ARM and the party that ARM delivered this document to.

Product Status

The information in this document is final, that is for a developed product.

Web Address

`http://www.arm.com`

Contents
ARM922T with AHB Product Overview

Chapter 1
ARM922T with AHB System-on-Chip Platform OS Processor

This product overview contains the following sections:

1.1 Applications and benefits

This section summarizes the applications and benefits of the ARM922T *System-on-Chip* (SoC) with AHB.

1.1.1 Applications

- Devices running:
  - EPOC
  - PalmOS
  - Linux
  - WindowsCE
  - QNX
- High-performance wireless devices
- Networking applications
- Digital set-top boxes
- Imaging systems
- Automotive control solutions
- Audio and video encode and decode.

1.1.2 Benefits

- Designed specifically for System-on-Chip integration
- Supports the Thumb® instruction set offering the same excellent code density as the ARM7TDMI cores
- High performance lets system designers integrate more functionality into price and power-sensitive applications
- Cached processor with an easy-to-use lower frequency on-chip system bus interface.

1.2 Key features

This section summarizes the key features of the ARM922T with AHB SoC.

1.2.1 The ARM922T (Rev 1) with AHB

The ARM922T macrocell is a high-performance 32-bit RISC integer processor combining an ARM9TDMI™ processor core with:

- 8KB instruction cache and 8KB data cache
- instruction and data *Memory Management Unit* (MMU)
- write buffer with 16 data words and 4 addresses
- *Advanced Microprocessor Bus Architecture* (AMBA™) AHB interface
- *Embedded Trace Macrocell* (ETM) interface
- EXTEST-compatible boundary scan chain.

1.2.2 High performance

The ARM922T macrocell provides a high-performance processor solution for open systems requiring full virtual memory management and sophisticated memory protection. The ARM922T processor core is capable of running at 200MHz on the TSMC 0.18µm process.

1.2.3 Low power consumption

The ARM922T custom-designed hard macrocell has a very small die size and very low power consumption. The integrated cache helps to significantly reduce memory bandwidth demands, improving performance and minimizing power consumption.

At 200MHz the ARM922T macrocell typically consumes as little as 180mW, making it ideal for many high-performance battery-powered consumer applications. This extremely low power consumption is essential for hand-held products supporting complex MP3, Java and video decoding, while concurrently executing baseband protocol software.

High levels of integration on-chip can be achieved because of the very low power dissipation characteristics.

1.2.4 Characteristics

Table 1-1 shows the ARM922T characteristics.

Table 1-1 ARM922T characteristics

ARM922T characteristics on typical 0.18μm process (preliminary figures)	
Die area	6.55mm^2
Power typical)	0.8mW/MHz @ 1.8V
Clock frequency and performance (worst-case on slow silicon, and 125°C)	220MIPS @ 200MHz

1.3 ARM922T with AHB overview

This section gives an overview of the ARM922T with AHB SoC.

1.3.1 Functional diagram

Figure 1-1 is a functional diagram of the ARM922T with AHB SoC.

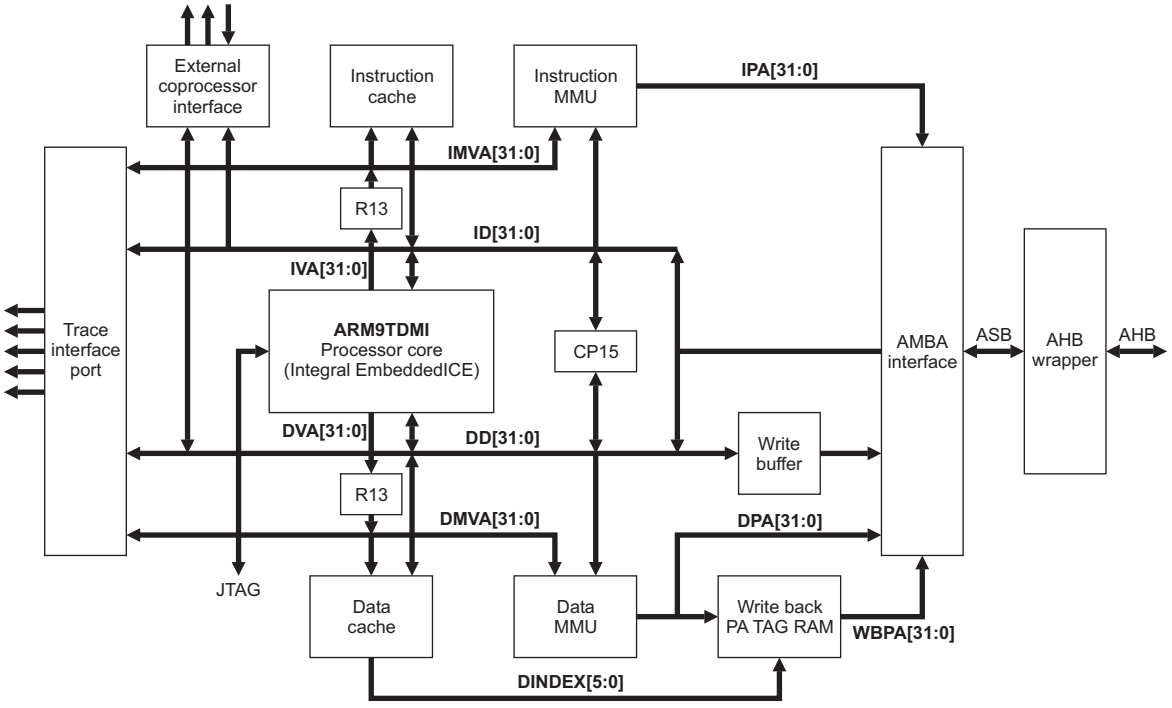

Figure 1-1 ARM922T Functional Diagram

1.3.2 ARM922T macrocell

The ARM922T macrocell is based on the ARM9TDMI Harvard architecture processor core, with an efficient five-stage pipeline.

To reduce the effect of main memory bandwidth and latency on performance, the ARM922T macrocell includes separate caches and MMUs for both instructions and data. It also has a write buffer and physical address TAG RAM.

1.3.3 Caches

Two 8KB caches are implemented, one for instructions, the other for data, both with an 8-word line size. Separate buses connect each cache to the ARM9TDMI core permitting a 32-bit instruction to be fetched and fed into the Decode stage of the pipeline at the same time as a 32-bit data access for the Memory stage of the pipeline.

Cache lock-down is provided to permit critical code sequences to be locked into the cache to ensure predictability for real-time code. The cache replacement algorithm can be selected by the operating system as either pseudo-random or round-robin. Both caches are 64-way set-associative. Lock-down operates on a per-way basis.

1.3.4 Write buffer

The ARM922T macrocell also incorporates a 16-data, 4-address write buffer to avoid stalling the processor when writes to external memory are performed.

1.3.5 PA TAG RAM

The ARM922T macrocell implements a *Physical Address Tag RAM* (PA TAG RAM) to perform write-backs from the data cache.

The physical addresses of all the lines held in the data cache are stored by the PA TAG memory, removing the requirement for address translation when evicting a line from the cache.

1.3.6 MMU

The ARM922T macrocell implements an enhanced ARMv4 MMU to provide translation and access permission checks for the instruction and data address ports of the ARM9TDMI core.

The MMU features are:

- standard ARMv4 MMU mapping sizes, domains, and access protection scheme
- mapping sizes are 1MB sections, 64KB large pages, 4KB small pages, and new 1KB tiny pages
- access permissions for sections
- access permissions for large pages and small pages can be specified separately for each quarter of the page (subpages)
- access permissions for tiny pages
- 16 domains implemented in hardware

- 64-entry instruction *Translation Lookaside Buffer* (TLB) and 64-entry data TLB
- hardware page table walks
- round-robin replacement algorithm (also called *cyclic*).

1.3.7 Control coprocessor (CP15)

The control coprocessor is provided for configuration of the caches, the write buffer, and other ARM922T options.

Eleven registers are available for program control:

- Register 1 controls system operation parameters including endianness, cache, and MMU enable
- Registers 2 and 3 configure and control MMU functions
- Registers 5 and 6 provide MMU status information
- Registers 7 and 9 are used for cache maintenance operations
- Registers 8 and 10 are used for MMU maintenance operations
- Register 13 is used for fast context switching
- Register 15 is used for test.

1.3.8 ARM9TDMI Embedded Trace Macrocell (ETM9) interface

The ETM interface permits the connection of an ETM9 to provide the capacity to trace instructions and data accesses in real time.

Real-time trace is made up of three elements:

- an *Embedded Trace Macrocell* (ETM)
- a *Trace Port Analyzer* (TPA)
- the *Trace Debug Tools* (TDT) extensions to the ARM Developer Suite.

1.3.9 Debug features

The ARM9TDMI processor core incorporates an EmbeddedICE unit and EmbeddedICE-RT logic permitting both software tasks and external debug hardware to:

- set hardware and software breakpoints
- perform single-stepping
- enable access to registers and memory.

This functionality is implemented as a coprocessor and is accessible from hardware through the JTAG port.

Full-speed, real-time execution of the processor is maintained until a breakpoint is hit.

At this point control is passed either to a software handler or to JTAG control.

1.3.10 Coprocessors

The ARM9TDMI coprocessor interface permits full independent processing of the ARM execution pipeline and supports up to 16 independent coprocessors. Four of these coprocessors are available for designer-specified purposes, such as for floating-point math operations, and signal processing.

1.3.11 32-bit data buses

The ARM9TDMI core provides 32-bit data buses:

- between the processor core and the instruction cache
- between the processor core and the data cache
- between coprocessors and the processor core.

This enables the ARM9TDMI core to achieve very high performance on many code sequences, especially those that require data movement in parallel with data processing.

1.4 The ARMv4T Architecture

This section gives an overview of the ARMv4T architecture implemented on the ARM922T.

1.4.1 The ARM9TDMI core

The ARM9TDMI processor core implements the ARMv4T *Instruction Set Architecture* (ISA). The ARMv4T ISA is a superset of the ARMv4 ISA (implemented by the StrongARM® processor) with additional support for the Thumb 16-bit compressed instruction set. The ARMv4T ISA is implemented by the ARM7™ Thumb family. This gives full upward compatibility to the ARM9™ Thumb family.

1.4.2 Performance and code density

The ARM9TDMI core executes two instruction sets:

- 32-bit ARM instruction set
- 16-bit Thumb instruction set.

The ARM instruction set is designed so that a program can achieve maximum performance with the minimum number of instructions. Most ARM9TDMI instructions are executed in a single cycle.

The simpler Thumb instruction set offers much increased code density reducing code size and memory requirement.

Code can switch between the ARM and Thumb instruction sets on any procedure call.

1.4.3 ARM9TDMI integer pipeline stages

The integer pipeline consists of five stages to maximize instruction throughput in the ARM9TDMI core:

1. Fetch
2. Decode and register read
3. Execute shift and ALU operation, or address calculate, or multiply
4. Memory access and multiply
5. Write register.

By using a five-stage pipeline, the ARM922T delivers a throughput approaching one instruction per cycle.

1.4.4 Registers

The ARM9TDMI processor core consists of a 32-bit datapath and associated control logic. This datapath contains 31 general purpose registers, coupled to a full shifter, Arithmetic Logic Unit, and a multiplier. At any one time 16 registers are visible to the user. The remainder are mode-specific replacement registers (banked registers) used to speed up exception processing, and to make nested exceptions possible.

Register 15 is the *Program Counter* (PC) that can be used in all instructions to reference data relative to the current instruction. R14 holds the return address after a subroutine call. R13 is used (by software convention) as a stack pointer.

1.4.5 Exception types/Modes

The ARM9TDMI core supports five types of exception, and a privileged processing mode for each type. The types of exceptions are:

- fast interrupt (FIQ)
- normal interrupt (IRQ)
- memory aborts (used to implement memory protection or virtual memory)
- attempted execution of an undefined instruction
- software interrupts (SWIs).

All exceptions have banked registers for R14 and R13. After an exception, R14 holds the return address for exception processing. This address is used both to return after the exception is processed and to address the instruction that caused the exception.

R13 is banked across exception modes to provide each exception handler with a private stack pointer. The fast interrupt mode also banks registers 8 to 12 so that interrupt processing can begin without the need to save or restore these registers.

A seventh processing mode, System mode, uses the User mode registers. System mode runs tasks that require a privileged processor mode and enables them to invoke all classes of exceptions.

1.4.6 Status registers

All other processor states are held in status registers. The current operating processor status is in the *Current Program Status Register* (CPSR). The CPSR holds:

- four ALU flags (Negative, Zero, Carry, and Overflow)
- an interrupt disable bit for each of the IRQ and FIQ interrupts
- a bit to indicate ARM or Thumb execution state
- five bits to encode the current processor mode.

All five exception modes also have a *Saved Program Status Register* (SPSR) that holds the CPSR of the task immediately before the exception occurred.

1.4.7 Conditional execution

All ARM instructions can be executed conditionally and can optionally update the four condition code flags (Negative, Zero, Carry, and Overflow) according to their result. Fifteen conditions are implemented.

1.4.8 Classes of instruction

The ARM and Thumb instruction sets can be divided into four broad classes of instruction:

- data processing instructions
- load and store instructions
- branch instructions
- coprocessor instructions.

1.4.9 Data processing instructions

The data processing instructions operate on data held in general-purpose registers. Of the two source operands, one is always a register. The other has two basic forms:

- an immediate value
- a register value optionally shifted.

If the operand is a shifted register, the shift can be an immediate value or the value of another register. Four types of shift can be specified. Most data processing instructions can perform a shift followed by a logical or arithmetic operation.

There are two classes of multiply instructions:

- normal, 32-bit result
- long, 64-bit result variants.

Both types of multiply instruction can optionally perform an accumulate operation.

1.4.10 Load and store instructions

There are two main types of load and store instructions:

- load or store the value of a single register
- load or store multiple register values.

Load and store single register instructions can transfer a 32-bit word, a 16-bit halfword, or an 8-bit byte between memory and a register. Byte and halfword loads can be automatically zero extended or sign extended as they are loaded. These instructions have three primary addressing modes:

- offset
- pre-indexed
- post-indexed.

The address is formed by adding an immediate, or register- based, positive, or negative offset to a base register. Register-based offsets can also be scaled with shift operations. Pre-indexed and post-indexed addressing modes update the base register with the base plus offset calculation.

As the PC is a general-purpose register, a 32-bit value can be loaded directly into the PC to perform a jump to any address in the 4GB memory space.

Load and store multiple instructions perform a block transfer of any number of the general purpose registers to, or from, memory. Four addressing modes are provided:

- pre-increment addressing
- post-increment addressing
- pre-decrement addressing
- post-decrement addressing.

The base address is specified by a register value (that can be optionally updated after the transfer). As the subroutine return address and the PC values are in general-purpose registers, very efficient subroutine calls can be constructed.

1.4.11 Branch instructions

As well as letting data processing or load instructions change control flow (by writing the PC) a standard branch instruction is provided with 24-bit signed offset, providing for forward and backward branches of up to 32MB.

A branch with link (BL) instruction enables efficient subroutine calls. BL preserves the address of the instruction after the branch in R14 (the Link Register or LR). This lets a move instruction put the LR in to the PC and return to the instruction after the branch.

The branch and exchange (BX) instruction switches between ARM and Thumb instruction sets with the return address optionally preserving the operating mode of the calling routine.

1.4.12 Coprocessor instructions

There are three types of coprocessor instructions:

- coprocessor data processing instructions
- coprocessor register transfer instructions
- coprocessor data transfer instructions.

1.4.13 The ARMv4T instruction sets

Table 1-2 shows the ARMv4T ARM and Thumb instruction sets.

Table 1-2 The ARMv4T ARM and Thumb Instruction Sets

ARM	Thumb	Instruction	Operation
Data processing instructions			
●	●	MOV	Move
●	●	MVN	Move Not
●		MRS	Move from Status Register
●		MSR	Move to Status Register
●	●	LDR	Load Word
●	●	LDRB	Load Byte
●	●	LDRH	Load Halfword
●	●	LDRSB	Load Signed Byte
●	●	LDRSH	Load Signed Halfword
●	●	LDM	Load Multiple
●	●	STR	Store Word
●	●	STRB	Store Byte
●	●	STRH	Store Halfword
●	●	STM	Store Multiple
Arithmetic instructions			
●	●	ADD	Add
●	●	ADC	Add with Carry

Table 1-2 The ARMv4T ARM and Thumb Instruction Sets (continued)

ARM	Thumb	Instruction	Operation
●	●	SUB	Subtract
●	●	SBC	Subtract with Carry
●	●	RSB	Reverse Subtract
●	●	RSC	Reverse Subtract with Carry
●	●	MUL	Multiply
●		MLA	Multiply Accumulate
●		UMULL	Unsigned Long Multiply
●		UMLAL	Unsigned Long Multiply Accumulate
●		SMULL	Signed Long Multiply
●		SMLAL	Signed Long Multiply Accumulate
Branch instructions			
●	●	B	Branch
●	●	BL	Branch with Link
●	●	BX	Branch and Exchange
Shift/rotate instructions			
	●	LSL	Logical Shift Left
	●	LSR	Logical Shift Right
	●	ASR	Arithmetic Shift Right
	●	ROR	Rotate Right
Logical and compare instructions			
●	●	AND	Logical AND
●	●	ORR	Logical (inclusive) OR
●	●	EOR	Logical Exclusive OR
●	●	BIC	Bit Clear
●	●	TST	Test

Table 1-2 The ARMv4T ARM and Thumb Instruction Sets (continued)

ARM	Thumb	Instruction	Operation
●		TEQ	Test Equivalence
	●	NEG	Negate
●	●	CMP	Compare
●	●	CMN	Compare Negated
Coprocessor instructions			
●	●	LDC	Load to Coprocessor
●	●	STC	Store from Coprocessor
●	●	MCR	Move to Coprocessor
●	●	MRC	Move from Coprocessor
●	●	CDP	Coprocessor Data Processing
Miscellaneous instructions			
●	●	SWI	Software Interrupt
●		SWPB	Swap Byte
●		SWP	Swap Word
	●	PUSH	Push Registers to Stack
	●	POP	Pop Registers from Stack

1.5 Interfaces on the ARM922T with AHB SoC

Table 1-3 shows the AHB interface signals used when the ARM922T is acting as the bus master.

Table 1-3 AHB interface signals used when ARM922T is the bus master

Name	Direction	Description
HADDRM[31:0] Address output bus	Output	The 32-bit system address bus.
HBURSTM[2:0] Burst type	Output	These signals indicate the type of burst transfer: 001 = INCR burst 011 = INCR4 burst 101 = INCR8 burst.
HBUSREQM Bus request	Output	When the processor requires the use of the bus, this is HIGH.
HGRANTM Bus grant	Input	When the processor is the granted bus master, this signal is HIGH. Ownership of the address/control signals changes at the end of a transfer when **HREADYM** is HIGH, so a bus master gains access to the bus when both **HREADYM** and **HGRANTM** are HIGH.
HLOCKM Locked transfers	Output	When the processor is the bus master and requires locked access to the bus, this signal is HIGH.
HPROTM[3:0] Protection control	Output	These signals provide information about the current memory access: **HPROTM[0]** is 0 if the transfer is an opcode fetch, or 1 for a data access **HPROTM[1]** is 0 if the transfer is a User access, or 1 for a Privileged access **HPROTM[2]** is 1 if the transfer is bufferable, or 0 if it is not **HPROTM[3]** is 1 if the transfer is cachable, or 0 if it is not.
HRDATAM[31:0] Read data bus	Input	The 32-bit data input bus.
HREADYM Transfer done	Input	This signal is driven HIGH by the selected slave to indicate that a bus transfer has finished. It is driven LOW to extend a bus transfer.
HRESPM[1:0] Transfer response	Input	These signals provide additional information on the status of a transfer: 00 = OKAY 01 = ERROR 10 = RETRY 11 = SPLIT.

Table 1-3 AHB interface signals used when ARM922T is the bus master (continued)

Name	Direction	Description
HSIZEM[2:0] Transfer size	Output	These signals indicate the size of the transfer: 000 = a byte (8-bit) 001 = half-word (16-bit) 010 = word (32-bit)
HTRANSM[1:0] Transfer type	Output	These signals indicate the type of the current transfer: 00 = IDLE 01 = BUSY 10 = NONSEQUENTIAL 11 = SEQUENTIAL.
HWDATAM[31:0] Write data bus	Output	The 32-bit data output bus.
HWRITEM Transfer direction	Output	During a write operation, this signal is HIGH. It is LOW during a read operation.

Table 1-4 shows the AHB interface signals used when the ARM922T is acting as the bus master or as the bus slave.

Table 1-4 AHB interface signals used when ARM922T is the bus master or a bus slave

Name	Direction	Description
HCLK Bus clock	Input	This clock controls all bus transfers. All signal timings are related to the rising edge of this signal.
HRESETn Reset	Input	Put this signal LOW to reset the system and the bus. It can be asserted asynchronously, but must be de-asserted synchronously. **HRESETn** must be held LOW for a minimum of five clock cycles. To completely reset the system, **nTRST** must also be placed LOW.

Table 1-5 shows the AHB interface signals used when the ARM922T is acting as the bus slave.

Table 1-5 AHB interface signals used when ARM922T is a bus slave

Name	Direction	Description
HADDRS[11:2] Address input bus	Input	The address input bus used when the processor is a bus slave.
HRDATAS[31:0] Read data bus	Output	The 32-bit data output bus used to transfer data from the processor to the bus master during read operations.
HREADYOUTS Transfer done	Output	When a transfer is complete, this signal is HIGH. This signal can be driven LOW to extend a transfer.
HREADYS Transfer done	Input	When this signal is HIGH, it indicates that a transfer between the bus master (TIC, when the ARM922T is in test mode) and another slave has finished. The signal is driven LOW to extend a bus transfer. If both master and slave ports are connected to the same AHB, then this signal must be connected to **HREADYM**.
HRESPS[1:0] Transfer response	Output	The transfer response provides additional information on the status of a transfer. This will always indicate: 00 = OKAY
HSELS Slave select	Input	When the current transfer is intended for the processor, this signal must be placed HIGH. The signal is a combinatorial decode of the address bus. Each AHB slave has its own select signal.
HTRANS1S Transfer	Input	This signal indicates that a transfer is occurring: 0 = IDLE/BUSY 1 = SEQUENTIAL/NONSEQUENTIAL.
HWDATAS[31:0] Write data bus	Input	The 32-bit data input bus data used to transfer data from the bus master to the processor.
HWRITES Transfer direction	Input	This signal indicates a write transfer when it is HIGH and a read transfer when it is LOW.

Table 1-6 shows the coprocessor interface signals.

Table 1-6 Coprocessor interface signals

Name	Direction	Description
CHSDE[1:0] Coprocessor handshake decode	Input	The handshake signals from the Decode stage of the coprocessor pipeline follower.
CHSEX[1:0] Coprocessor handshake execute	Input	The handshake signals from the Execute stage of the coprocessor pipeline follower.
CPCLK Coprocessor clock	Output	This clock controls the operation of the coprocessor interface.
CPDOUT[31:0] Coprocessor data out	Output	The 32-bit data bus for transfers from the processor to the coprocessor.
CPDIN[31:0] Coprocessor data in	Input	The 32-bit data bus for transfers from the coprocessor to the processor.
CPEN Coprocessor data out enable	Input	When tied LOW, the **CPID** and **CPDOUT** buses are held stable. When tied HIGH, the **CPID** and **CPDOUT** buses are enabled. Normally this is a static configuration signal.
CPID[31:0] Coprocessor data in	Input	The 32-bit instruction data bus for transferring instructions to the pipeline follower in the coprocessor.
CPLATECANCEL Coprocessor late cancel	Output	When this signal is HIGH during the first Memory cycle, the coprocessor must cancel the instruction without updating the coprocessor state.
CPPASS Coprocessor pass	Output	When there is a coprocessor instruction in the Execute stage of the pipeline, and it must be executed, this signal is HIGH.
CPTBIT Coprocessor Thumb bit	Output	When the coprocessor interface is in Thumb state, this signal is HIGH.

Table 1-6 Coprocessor interface signals (continued)

Name	Direction	Description
nCPMREQ Not coprocessor memory request	Output	When the coprocessor should read an instruction from **CPID[31:0]**, this signal is LOW.
nCPTRANS Not coprocessor translate	Output	When the coprocessor interface is in a privileged mode, this signal is LOW.
nCPWAIT Not coprocessor wait	Output	When cycles must be extended on the coprocessor interface, this signal is LOW.

Table 1-7 shows the JTAG and TAP controller signals.

Table 1-7 JTAG and TAP controller signals

Name	Direction	Description
TAPID[31:0] Test access port identification	Input	Static configuration signals to determine the device identification (ID) code: Bits 31:28 are for the revision code Bits 27:12 are for the product code Bits 11:1 are the manufacturer's ID code Bit 0 must be 1 (IEEE specified).
TCK Test clock	Input	The clock for JTAG signals.
TDI Test data input	Input	JTAG serial input.
TDO Test data output	Output	JTAG serial output.
nTDOEN Not **TDO** enable	Output	When serial data is being driven out on the **TDO** output, this is HIGH. This signal is normally used as an output enable for a **TDO** pin in a packaged part.
TMS Test mode select	Input	This signal selects the next state, from the two possible options, for every state transition of the JTAG state machine used in the TAP controller.
nTRST Not test reset	Input	To correctly reset the boundary scan logic, this signal must be pulsed or driven LOW. This signal must be driven LOW whenever **HRESETn** is used to reset the processor to ensure correct processor operation.

Table 1-8 shows the debug signals.

Table 1-8 Debug signals

Name	Direction	Description
COMMRX Communications channel receive	Output	When the communications channel receive buffer contains data waiting to be read by the processor, this signal is HIGH.
COMMTX Communications channel transmit	Output	When the communications channel transmit buffer is empty, this signal is HIGH.
DBGACK Debug acknowledge	Output	When the processor is in debug state, this signal is HIGH.
DBGEN Debug enable	Input	A static configuration signal that disables the debug features when LOW.
DBGRQI Internal debug request	Output	This is a logical OR of **EDBGRQ** and bit 1 of the debug control register.
DEWPT External watchpoint	Input	This signal provides an input for external data watchpoints to be implemented. External watchpoints can be used to let hardware force the processor into debug state.
ECLK External clock output	Output	For use by external devices. This is a copy of the clock used by the processor, as modified by clock switching, or debug state.
EDBGRQ External debug request	Input	Place this HIGH to request that the processor enter debug state when execution of the current instruction has completed.
EXTERN0 External input 0	Input	This is connected to watchpoint unit 0 of the EmbeddedICE logic in the processor, and permits breakpoints or watchpoints to be dependent on an external condition.
EXTERN1 External input 1	Input	This is connected to watchpoint unit 1 of the EmbeddedICE logic in the processor, and permits breakpoints or watchpoints to be dependent on an external condition.
IEBKPT External breakpoint	Input	This signal provides an input for external instruction breakpoints to be implemented. External breakpoints can be used to let hardware force the processor into debug state.
INSTREXEC Instruction executed	Output	When the instruction in the Execute stage of the pipeline has been executed, this signal goes HIGH.

Table 1-8 Debug signals (continued)

Name	Direction	Description
RANGEOUT0 EmbeddedICE rangeout 0	Output	When the EmbeddedICE watchpoint unit 0 has matched the conditions currently present on the address, data, and control buses, this signal is HIGH. This signal is independent of the state of the watchpoint unit enable control bit.
RANGEOUT1 EmbeddedICE rangeout 1	Output	When the EmbeddedICE watchpoint unit 1 has matched the conditions currently present on the address, data, and control buses, this signal is HIGH. This signal is independent of the state of the watchpoint unit enable control bit.
TRACK Enable TrackingICE mode	Input	Place this HIGH to enable TrackingICE mode for debugging purposes.

Table 1-9 shows the INTEST/EXTEST wrapper signals.

Table 1-9 INTEST/EXTEST wrapper signals

Name	Direction	Description
WP_SI Scan data input	Input	Scan data input to the start of the INTEST/EXTEST wrapper.
WP_SO Scan data output	Output	Scan data output from the end of the INTEST/EXTEST wrapper.
WP_SI_INT[7:1] Test scan inputs	Input	Test scan inputs to each subsection of the wrapper.
WP_SO_INT[6:0] Test scan outputs	Output	Test scan outputs from each subsection of the wrapper.
SCANEN Scan enable	Input	Enables scanning of data through the INTEST/EXTEST wrapper.
INTEST	Input	This signal is HIGH during an INTEST operation.
EXTEST	Input	This signal is HIGH during an EXEST operation.

Table 1-10 shows the miscellaneous signals.

Table 1-10 Miscellaneous signals

Name	Direction	Description
BIGENDOUT Big-endian output	Output	When operating in big-endian configuration, this signal is HIGH. LOW for little-endian.
FCLKOUT Buffered **FCLK**	Output	Buffered version of **FCLK** input.
FCLK Fast clock	Input	Used when the processor is in the synchronous or asynchronous clocking mode (not during fast bus mode).
VINITHI Vectors high	Input	Determines the state of the V bit in Register 1 of CP15 coming out of reset. When HIGH, V bit is 1 coming out of reset. When LOW, V-Bit is 0 coming out of reset.
ISYNC Synchronous interrupts	Input	Set this HIGH if interrupts are synchronous to the processor clock. LOW for asynchronous interrupts.
nFIQ Not fast interrupt request	Input	Place this signal LOW to generate a fast interrupt request.
nIRQ Not interrupt request	Input	Place this LOW to generate an interrupt request.
SI Scan in	Input	Scan data in to the AHB wrapper.
SO Scan out	Output	Scan data out from the AHB wrapper.

Table 1-11 on page 1-24 shows the ETM interface signals.

Table 1-11 ETM interface signals

Name	Direction	Description
ETMBIGEND **ETMCHSD[1:0]** **ETMCHSE[1:0]** **ETMCLOCK** **ETMDA[31:0]** **ETMDABORT** **ETMDBGACK** **ETMDD[31:0]** **ETMDMAS[1:0]** **ETMDMORE** **ETMDnMREQ** **ETMDnRW** **ETMDSEQ** **ETMHIVECS** **ETMIA[31:1]** **ETMIABORT** **ETMID15To8[15:8]** **ETMID31To24[31:24]** **ETMInMREQ** **ETMINSTREXEC** **ETMISEQ** **ETMITBIT** **ETMLATECANCEL** **ETMPASS** **ETMRNGOUT[1:0]** **ETMnWAIT**	Output	These signals connect directly to the equivalent signals on the ETM9.
ETMPWRDOWN	Input	This signal connects directly to the equivalent signal on the ETM9. It must be tied LOW if no ETM is connected.

1.6 System Issues and Third Party Support

This section describes system issues and third party support for the ARM922T with AHB SoC.

1.6.1 JTAG debug

The internal state of the ARM922T macrocell is examined through a JTAG-style serial interface. This permits instructions to be inserted serially into the pipeline of the core without using the external data bus. Therefore, when in debug state, a store-multiple (STM) can be inserted into the instruction pipeline. This exports the contents of the ARM9TDMI registers. This data can be serially shifted out without affecting the rest of the system.

1.6.2 AMBA bus architecture

The ARM9 Thumb Family processors are designed for use with the AMBA multi-master on-chip bus architecture. AMBA includes an *Advanced High-performance Bus* (AHB) connecting processors and high-bandwidth peripherals and memory interfaces, and a low power peripheral bus permitting a large number of low-bandwidth peripherals.

The ARM922T macrocell implements a full AHB interface, either as an AHB bus master, or as a slave for production test.

1.6.3 Everything you need

ARM Limited provides a wide range of products and services to support its processor families, including software development tools, development boards, models, applications software, training, and consulting services.

1.6.4 Compatible with ARM7 and StrongARM processor cores

The ARM9 Thumb Family processors are strongly software compatible with existing ARM families. This ensures that users benefit immediately from existing support. ARM is working with its software, EDA, and semiconductor partners to extend this support to use new ARM9 Family features.

The ARM922T processor is 100% user code binary compatible with the ARM7TDMI core, and backwards compatible with the ARM7 Thumb family and the StrongARM processor families. This gives designers software-compatible processors with a range of price versus performance points from 60MIPS to more than 1000MIPS.

1.6.5 Current support

The ARM Architecture today enjoys broad third-party support. Support for the ARM Architecture today includes:

- *ARM Developer Suite* (ADS):
  - integrated development environment
  - C, C++, assembler, instruction set simulators, and windowing source-level debugger
  - available on Windows 95, 98, Windows NT, Windows 2000, and Unix.
- ARM Multi-ICE™ JTAG interface:
  - permits EmbeddedICE software debug of ARM processor systems
  - integrates with ADS.
- ARM Multi Trace
- ARMulator instruction-accurate software simulator
- development boards
- Design Signoff Models provide sign off quality ASIC simulation
- software toolkits available from ARM, Cygnus/GNU, Greenhills, JavaSoft, MetaWare, Mentor, and WindRiver permitting software development in C, C++, Java, FORTRAN, Pascal, Ada, and assembler.
- more than 40 Real Time Operating Systems including:
  - WindRiver VxWorks
  - Sun Microsystems Chorus
  - JavaOS
  - Mentor VRTX
  - JMI
  - Embedded System Products RTXC
  - Integrated Systems pSOS.
- major OS including:
  - Microsoft WindowsCE
  - Symbian EPOC
  - Linux.

- application software components including:
  - speech and image compression
  - software modems
  - Chinese character input network protocols
  - digital AC3 decode
  - MPEG3 encode and decode
  - MPEG4 decode and encode.
- hardware and software cosimulation tools from leading EDA Vendors.

For more information, see `www.arm.com`.

