ARM926EJ-S

System-on-Chip

Java™ and DSP enhanced processor

Product Overview

Target Applications

- Java enabled Mobile, Smart phones and PDAs
- Applications requiring a platform OS:
  - EPOC, Linux, PalmOS, and WindowsCE
- Wireless internet applications
- Networking applications
- Digital set top boxes
- Automotive:
  - Hands-free interfaces
  - Infotainment
- MPEG4 Video encoding and decoding
- Audio decoding:
  - Dolby AC3 digital
  - High-end MP3 players
  -AAC
- Speech codes

Benefits

- Integrated single chip microcontroller, DSP, and Java solution
- Very efficient Java byte code execution
- Reduced power consumption and chip complexity over a Java coprocessor solution
- Flexibility with tightly coupled memory system and variable size caches
- Rapid ASIC or ASSP integration with a short time-to-market
- Interface to ETM9 for real time trace support

The ARM926EJ-S™

The ARM926EJ-S macrocell is a fully synthesizable 32-bit RISC processor comprising an ARM9EJ-S™ Java enhanced processor core, instruction and data caches, *Tightly-coupled Memory* (TCM) interfaces, *Memory Management Unit* (MMU), and separate instruction and data AMBA™ (Advanced Microprocessor Bus Architecture) *Advanced High-performance Bus* (AHB) Bus Interfaces. The sizes of both caches and TCM memories can all be specified independently. The TCM interface is flexible providing the capability to attach nonzero wait state memory and to support DMA accesses, while retaining support for single cycle memory accesses using zero-wait state RAM using the same interface. This flexibility allows the Macrocell to be tailored to the specific application needs.The MMU supports virtual memory based platform operating systems such as EPOC, Linux, WindowsCE, and PalmOS. New power management features allow:

- system-level RAM power down
- architectural clock stopping
- logic-level clock gating.

The ARM926EJ-S provides a complete high-performance single processor solution, offering considerable savings in chip complexity and area, system design, power consumption, and time-to-market. It provides higher performance, lower system cost and lower power consumption than coprocessors and dual-processor solutions.

Java enhancements (Jazelle™)

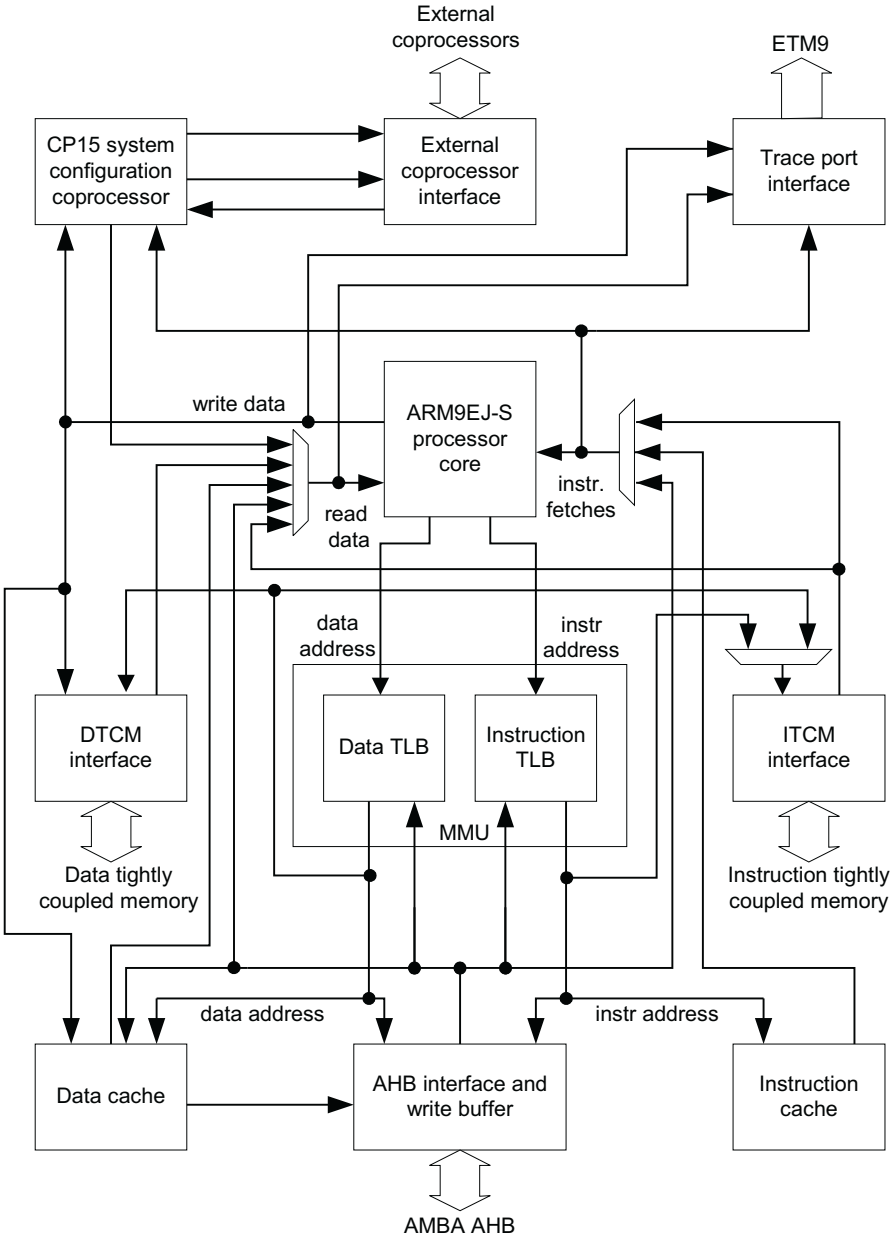
The ARM926EJ-S processor has ARM's embedded Jazelle Java acceleration hardware, within the ARM9EJ-S core. In the rapidly developing market for internet-enabled applications, Java offers advantages of rapid application development to software engineers, and the ARM926EJ-S is an ideal mechanism for implementing those applications.

The ARM9EJ-S processor core executes an extended ARMv5TE instruction set, which includes support for Java byte code execution (ARMv5TEJ). An ARM optimized *Java Virtual Machine* (JVM) software layer has been written to work with the Jazelle hardware. The Java byte code acceleration is achieved by the hardware directly executing 80% of simple Java byte codes directly and the remaining 20% of less frequently byte codes by software emulation within the ARM optimized JVM.

DSP enhancements

The ARM926EJ-S processor core offers the full advantage of the enhanced DSP capability of the ARM9EJ-S processor core. Multiply instructions are processed using a single-cycle 32x16 implementation. There are 32x32, 32x16, and 16x16 multiply instructions or *Multiply Accumulate* (MAC), and the pipeline allows one multiply to start each cycle. Saturating arithmetic improves efficiency by automatically selecting saturating behavior during execution and is used to set limits on signal processing calculations to minimize the effect of noise or signal errors. All of these instructions are beneficial for algorithms that implement:

- GSM protocols
- FFT
- state space servo control.



Memory Management Unit

The MMU supports a demand page virtual memory system required by platform operating systems such as Linux, EPOC, or WindowsCE. The MMU provides the access protection mechanism for all memory accesses. The address translation, protection and region type information is stored in a series of page tables in main memory. The MMU contains hardware for automatically reading and processing these tables, and storing the resultant translation and protection information in a *translation lookaside buffer* (TLB).

- Page sizes of 1MB, 64KB, 4KB & 1KB
- Hardware page table walks
- Separate TLB for instruction and data
- Domains allow protection attributes to be changed without TLB flushes
- Fast context switch enables virtual address remapping in a 0-32MB region
- TLB entries for critical code and data can be locked down
- TLB entries can be removed from the TLB selectivity
- Access permissions for large pages and small pages can be specified separately for each quarter of the page (these quarters are called subpages).

Caches

The ARM926EJ-S uses separate caches for instructions and data, allowing both an instruction fetch, and a load/store access to take place simultaneously. The cache is constructed using standard compiled SRAM blocks, allowing a wide range of processes and libraries to be targeted The size of the each cache can be selected from the following range:

- 4KB
- 8KB
- 16KB
- 32KB
- 64KB
- 128KB

The caches are virtually addressed, with each cache line containing eight words. Cache lines are allocated on a read-miss basis. Victim replacement can be configured to be either round-robin or pseudo random.
A cacheable memory region can be defined as either being writeback (copy-back) or write-through. This is controlled by the memory region attributes contained in the page table entry for the region.

TCM interfaces

The ARM926EJ-S provides a separate TCM interface for instructions and data. The interface is designed to allow single cycle access for data and instructions. The interface allows TCM accesses to be extended by virtue of a *wait* signal. Extending memory accesses allows nonzero wait state memory to be used, and for DMA type access to occur. To optimize use of memories which have a different access latency for non-sequential and sequential accesses, the type of access to the TCM address space is indicated. There is no restriction on the type of memory or memory system attached to the TCM interface other than that it must not contain read-sensitive locations.

The TCM uses physical addresses. Write accesses are controlled using the protection information stored in the MMU.

The TCMs are typically used to store critical code and data for which deterministic access times are required, which is not always the case for a code or data which resides in a cacheable region of memory.

Bus Interface Unit (BIU)

The ARM926EJ-S contains a separate AMBA BIU for instructions and data. Each interface is a full AHB bus master, allowing all the advantages of the multilayer AHB system to be realized.

Unless a memory region is specified as being nonbuffered all writes are made using the 16-entry write-buffer, which avoids stalling the processor core when writes to main memory are performed.

Coprocessor interface

The ARM926EJ-S supports the connection of on-chip coprocessors through an external coprocessor interface.

The ARM v5TEJ Architecture

Registers

The ARM9EJ-S processor core consists of a 32-bit datapath and associated control logic. The datapath contains 31 general-purpose registers, coupled to a full shifter, Arithmetic Logic Unit, and multiplier. At any one time 16 registers are visible to the user. The remainder are banked registers used to speed up exception processing. Register 15 is the *Program Counter* (PC) and can be used by the majority of 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.

Modes and exception handling

All privileged modes and exception handling modes 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 in some cases used 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 eight to 12 so that interrupt processing can begin without the need to save or restore these registers. A seventh processing mode, System mode, does not have any banked registers. It uses the User mode registers. System mode runs tasks that require a privileged processor mode and allows them to invoke all classes of exceptions.

Status registers

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

- Q flag (used in saturating arithmetic instructions)
- four ALU flags (Negative, Zero, Carry, and Overflow)
- two interrupt disable bits (one for each type of interrupt)
- state bits to indicate ARM, Thumb, or Java state
- five bits to encode the current processor mode.

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

Exception types

ARM9EJ-S 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
- undefined instruction
- software interrupt (SWI).

Conditional execution

The majority of ARM instructions are conditionally executed. Instructions optionally update the four condition code flags (Negative, Zero, Carry, and Overflow) according to their result. Subsequent instructions are conditionally executed according to the status of flags. Fifteen conditions are implemented.

Four classes of instructions

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.

Data processing

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 amount might have 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. Multiply instructions come in two classes:

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

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

Load and store

The second class of instruction is load and store instructions. These instructions come in two main types:

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

Load and store single register instructions can transfer a 32-bit word, a 16-bit halfword and an eight-bit byte between memory and a register. Byte and halfword loads can be automatically zero extended or sign extended as they are loaded. Swap instructions perform an atomic load and store as a synchronization primitive.

Addressing modes

Load and store instructions have three primary addressing modes:

- offset
- pre-indexed
- post-indexed.

They are formed by adding or subtracting an immediate or register-based offset to or from 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.

Block transfers

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 (which 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.

Branch

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

Branch with link

There is a Branch with Link (BL) instruction which allows efficient subroutine calls by automatically preserving the return address.

Branch with exchange

There are three types of branches:

- BX switches between ARM state and Thumb state
- BLX switches between ARM state and Thumb state preserving the return address
- BXJ switches between ARM state and Java state.

Coprocessor

There are three types of coprocessor instructions:

- coprocessor data processing instructions are used to invoke a coprocessor specific internal operation
- coprocessor register transfer instructions allow a coprocessor value to be transferred to or from an ARM register
- coprocessor data transfer instructions transfer coprocessor data to or from memory, where the ARM calculates the address of the transfer.

The V5TEJ ARM Instruction Set including DSP extensions

Mnemonic	Operation	Mnemonic	Operation
MOV	Move	MVN	Move Not
QADD	Saturated add	QDADD	Saturated add with double
QSUB	Saturated subtract	QDSUB	Saturated subtract with double
RSB	Reverse subtract	RSC	Reverse Subtract with Carry
CMP	Compare	CMN	Compare Negated
TST	Test	TEQ	Test Equivalence
ADD	Add	ADC	Add with Carry
SUB	Subtract	SBC	Subtract with Carry
AND	Logical AND	BIC	Bit Clear
EOR	Logical exclusive OR	ORR	Logical (inclusive) OR
MUL	Multiply	MLA	Multiply Accumulate
SMUL	Signed Multiply	SMULW	Signed Word Multiply-Accumulate
SMULL	Sign long multiply	SMLA	Signed Multiply-Accumulate
SMLAW	Signed Word Multiply-Accumulate	SMLAL	Signed Long Multiply-Accumulate
UMULL	Unsigned long multiply	UMLAL	Unsigned Long Multiply Accumulate
CLZ	Count Leading Zeroes	BKPT	Breakpoint
MRS	Move From Status Register	MSR	Move to Status Register
B	Branch	BXJ	Branch to Java
BL	Branch and Link	BLX	Branch and Link and Exchange
BX	Branch and Exchange	SWI	Software Interrupt
LDR	Load Word	STR	Store Word
LDRH	Load Halfword	STRH	Store Halfword
LDRB	Load Byte	STRB	Store Byte
LDRD	Load double	STRD	Store double
LDM	Load Multiple	STM	Store Multiple
LDRSH	Load Signed Halfword	LDRSB	Load Signed Byte
LDRT	Load Register with Translation	STRT	Store Register with Translation
LDRBT	Load Register Byte with Translation	STRBT	Store Register Byte with Translation
SWP	Swap Word	SWPB	Swap Byte
PLD	No operation	CDP	Coprocessor Data Processing
MRC	Move From Coprocessor	MCR	Move to Coprocessor
MRCC	Move double from coprocessor	MCRR	Move double to coprocessor
LDC	Load To Coprocessor	STC	Store From Coprocessor

The Thumb instruction set

Mnemonic	Operation	Mnemonic	Operation
MOV	Move	MVN	Move Not
ADD	Add	ADC	Add with Carry
SUB	Subtract	SBC	Subtract with Carry
CMP	Compare	CMN	Compare Negated
TST	Test	NEG	Negate
AND	Logical AND	BIC	Bit Clear
EOR	Logical Exclusive OR	ORR	Logical (inclusive) OR
LSL	Logical Shift Left	LSR	Logical Shift Right
ASR	Arithmetic Shift Right	ROR	Rotate Right
MUL	Multiply	BKPT	Breakpoint
B	Unconditional Branch	Bcc	Conditional Branch
BL	Branch and Link	BLX	Branch and Link and Exchange
BX	Branch and Exchange	SWI	Software Interrupt
LDR	Load Word	STR	Store Word
LDRH	Load Halfword	STRH	Store Halfword
LDRB	Load Byte	STRB	Store Byte
LDRSH	Load Signed Halfword	LDRSB	Load Signed Byte
LDMIA	Load Multiple	STMIA	Store Multiple
PUSH	Push Registers to stack	POP	Pop Registers from stack

Modes and registers

User and System mode	Supervisor mode	Abort mode	Undefined mode	Interrupt mode	Fast Interrupt mode
R0	R0	R0	R0	R0	R0
R1	R1	R1	R1	R1	R1
R2	R2	R2	R2	R2	R2
R3	R3	R3	R3	R3	R3
R4	R4	R4	R4	R4	R4
R5	R5	R5	R5	R5	R5
R6	R6	R6	R6	R6	R6
R7	R7	R7	R7	R7	R7
R8	R8	R8	R8	R8	R8_FIQ
R9	R9	R9	R9	R9	R9_FIQ
R10	R10	R10	R10	R10	R10_FIQ
R11	R11	R11	R11	R11	R11_FIQ
R12	R12	R12	R12	R12	R12_FIQ
R13	R13_SVC	R13_ABORT	R13_UNDEF	R13_IRQ	R13_FIQ
R14	R14_SVC	R14_ABORT	R14_UNDEF	R14_IRQ	R14_FIQ
PC	PC	PC	PC	PC	PC

CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
	SPSR_SVC	SPSR_ABORT	SPSR_UNDEF	SPSR_IRQ	SPSR_FIQ

Mode-specific banked registers

ARM9EJ-S processor core

The ARM9EJ-S processor core is an implementation of the ARM *Instruction Set Architecture* (ISA) ARMv5TEJ, extended with enhanced Java and DSP performance. ARMv5TEJ is a superset of the ARMv4 ISA implemented by the StrongARM™ processors and the ARMv4T ISA implemented by the ARM7™ Thumb and ARM9™ Thumb Family processors.

Performance and code density

The ARM9EJ-S executes three instruction sets:

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

The ARM instruction set allows a program to achieve maximum performance with the minimum number of instructions.

The majority of ARM9EJ-S instructions are executed in a single cycle. These are shown in the ARM9EJ-S instruction execution timing table on page 10. The table shows for each instruction class:

- Issue Cycles:
  Number of cycles in execute stage.
- Result Delay:
  Number of cycles after execute until data available.

The simpler Thumb instruction set offers much increased code density for code that does not require maximum performance. Code can switch between the ARM and Thumb instruction sets on any procedure call.

Java state

In Java state the ARM9EJ-S processor core executes a majority of Java Bytecodes natively. Bytecodes are decoded in two stages (compared to a single decode stage when in ARM/Thumb state).

ARM9EJ-S integer pipeline stages

The integer pipeline consists of five stages to maximize instruction throughput on ARM9EJ-S:

F: Instruction Fetch

D: Instruction Decode and Register Read

E: Execute Shift and ALU, or Address Calculate, or Multiply

M: Memory Access and Multiply

W: Register Write

Pipelining

By overlapping the various stages of execution, ARM9EJ-S maximizes the clock rate achievable to execute each instruction. It delivers a throughput approaching one instruction per cycle.

32-bit data buses

ARM9EJ-S provides 32-bit data buses between the processor core and the instruction and data caches, and between coprocessors and the processor core. This allows ARM9EJ-S to achieve very high performance on many code sequences, especially those that require data movement in parallel with data processing.

Debug features

The ARM9EJ-S processor core also incorporates a sophisticated debug unit to allow both software tasks and external debug hardware to perform hardware and software breakpoint, single stepping, register and memory access. This functionality is made available from hardware using 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.

Table 1: ARM9EJ-S instruction execution timing

Instruction class	Issue Cycles	Result delay
Branch	3 (4 for BXJ)	NA
Condition failed	1	NA
ALU instruction	1	0
ALU instruction with register specified shift	2	0
MOV PC, Rx	3	NA
ALU instruction dest = PC	3(4 for shift instructions)	0
MUL (flags modified)	4 to 5	0
MUL (flags unmodified)	1 to 3	1
MSR (flags only)	1	0
MSR (mode change)	3	0
MRS	2	0
LDR (loaded value)	1(2 for shift instructions)	1
STR	1(2 for shift instructions)	NA
LDM	Number of words	0 (Not last register in list)
LDM	Number of words	1 (Last register in list)
STM	Number of words	NA
SWP	2	1
CDP	1	NA
MRC	1	1
MCR	1	NA
LDC	Number of words	NA
LDC dest = PC	5	NA
LDC dest = PC, scaled offset	6	NA
STC	Number of words	NA
LDRD	2	0 (Not last register in list)
LDRD	2	1 (Last register in list)
STRD	2	NA
PLD	1	NA
MRRC	2	0 (Not last register in list)
MRRC	2	1 (Last register in list)
MCRR	2	NA

System Design and Third Party Support

Embedded Trace Macrocell

The *Embedded Trace Macrocell* (ETM) monitors the ARM926EJ-S processor core address, data, and status signals and generates a compressed trace data stream of processor activity. The ETM provides non-intrusive real-time instruction trace by broadcasting compressed instruction addresses at branches. Data tracing is supported by broadcasting load and store operations. The unique triggering and data capabilities of the ETM means that complex events can be easily traced.

AMBA bus architecture

The ARM926EJ-S processor is designed for use with the AMBA multi-master on-chip bus architecture. AMBA includes an AHB connecting processors and high-bandwidth peripherals and memory interfaces, and a low-power peripheral bus allowing a large number of low-bandwidth peripherals.

The ARM926EJ-S supports multilayer AHB, providing 32-bit address and 32-bit data buses for instructions and data for high-bandwidth data transfers made possible by on-chip memory and modern SDRAM and RAMBUS memories.

Compatibility

The ARM926EJ-S is backwards compatible from the ARM7, ARM9, ARM9E Thumb and StrongARM processor families, giving designers software-compatible processors with a wide range of price and performance points.

Everything you need

ARM 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.

The ARM Architecture today enjoys broad third party support. The ARM9 Thumb family of processors has strong software compatibility with existing ARM families ensuring that its 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.

Current support

Support for the ARM architecture today includes:

- Software toolkits available from ARM: *ARM Developer Suite* (ADS), Cygnus/GNU, Microtec, Greenhills, JavaSoft, MetaWare, and Windriver allowing software development in C, C++, Java, FORTRAN, Pascal, Ada, and assembler. *ARM Developer Suite* (ADS) includes:
  - Integrated development environment
  - C, C++, assembler, simulators and windowing source-level debugger
  - Available on Windows95, WindowsNT, and Unix.
  - ARM Multi-ICE™ JTAG interface
  - Allows EmbeddedICE software debug of ARM processor systems
  - ARMulator instruction-accurate software simulator.
- ARM and third party Development boards.
- Application software components: DSP, speech and image compression, software modems, Chinese character input, network protocols, for example.
- Design Sign-off Models provide quality ASIC simulation.
- Major OS including Microsoft WindowsCE, EPOC, and LINUX.
- 40+ Real-time Operating Systems including Windriver VxWorks, pSOS, Sun Microsystems, Microtec VRTX, JMI, Embedded SystemProducts RTXC.
- Hardware and software cosimulation tools from leading EDA Vendors.

For more information, see www.arm.com

Contacting ARM

America
Los Gatos
ARM INC.
750 University Avenue
Suite 150,
Los Gatos, CA 95032
USA

Tel: +1-408-579-2209
Fax:+1-408-399-8854
Email: info@arm.com

Austin
ARM INC.
1250 Capital of Texas Highway
Building3, Suite 560
Austin
Texas 78746
USA

Tel: +1-512-327-9249
Fax:+1-512-314-1078
Email: info@arm.com

Seattle
ARM INC.
18300 NE Union Hill Road
Suite 257
Redmond, WA 98052
USA

Tel: +1-425-882-9781
Email: info@arm.com

Boston
ARM INC.
300 West Main St., Suite 215
Northborough
MA 01532
USA

Tel: +1-508-351-1670
Fax:+1-508-351-1668
Email: info@arm.com

San Diego
ARM INC.
4275 Executive Square
Suite 800
La Jolla, CA 92037
USA

Tel: +1-858-546-2935
Fax:+1-508-351-2936
Email: info@arm.com

England
ARM Ltd.
110 Fulbourn Road
Cherry Hinton
Cambridgeshire
CB1 9NJ
England

Tel: +44 1223 400400
Fax:+44 1223 400410
Email: info@arm.com

France
ARM
12 Avenue des Pres
BL 204 Montigny le Bretonneux
78059 Saint Quentin en Yvelines
Cedex
FRANCE

Tel: +33 1-30-79-05-10
Fax: +33 1-30-79-05-11
Email: info@arm.com

Germany
ARM Germany
Rennweg 33
85435 Erding
Germany

Tel: +49-8122-89209-0
Fax:+49-8122-89209-49
Email: info@arm.com

Japan
ARM K.K.
Plustaria Building 4F
3-1-4 Shin-Yokohama
Kohoku-ku,
Yokohama-shi
Kanagawa 222-0033

Tel: +81 45 477 5260
Fax: +81 45 477 5261
Email: info-armkk@arm.com

Korea
ARM
Room #1115, Hyundai Building
9-4, Soonae-Dong, Boondang-Ku
Sungnam, Kyunggi-Do
Korea 463-020

Tel: +82-342 712 8234
Fax: +82 432 713 8225
Email: info@arm.com

Taiwan
ARM Taiwan
8F, No. 50
Lane 10
Kee Hu Road
Taipee (114)
TAIWAN

Tel. +886-2-26271681
Fax. +886-2-26271682