TMS470 INSTRUCTION SET SIMULATOR TECHNICAL OVERVIEW

- **Included in Code Composer Studio™ Integrated Development Environment (IDE) for TMS470™**
- **TMS470 CPU Full Instruction Set Architecture Execution**
- **Port Connect**
  - **Supports External Data Simulation**
- **Pin Connect**
  - **Supports External Event Simulation**

Description

The TMS470 simulator, available within the Code Composer Studio™ Integrated Development Environment (IDE), simulates the instruction set of the ARM™ family of processors. Table 1 lists the processors supported, with the corresponding configuration to be selected under the Import Configuration menu of Code Composer Studio Setup.

Table 1. Processors Supported by the TMS470 Simulator

PROCESSOR	CODE COMPOSER STUDIO IDE IMPORT CONFIGURATION
TMS470R1x	ARM7 Simulator, Little Endian
TMS470R1x	ARM7 Simulator, Big Endian
TMS470R2x	ARM9 Simulator, Little Endian
TMS470R2x	ARM9 Simulator, Big Endian
TMS470R2x	ARM925 Simulator, Little Endian
TMS470R2x	ARM926EJ-S Simulator, Little Endian

Considerations for Choosing a Simulator

The core only simulators, ARM7 and ARM9, can be used for algorithmic verification and for benchmarking. If cycle-count accuracy of the core is of primary concern and full device simulation is not needed, then these two configurations can be used. Any accesses outside the core will be handled by a flat memory model and, therefore, the cycles measured will not account for any memory access latencies.

If the effects of components like MMU or Cache are to be observed, then the ARM925 and ARM926 simulators can be used. Cycle-count accuracy of these simulators can not be guaranteed. Accesses outside the ARM925 or ARM926 will be handled by a flat memory model.

Supported Hardware Resources

Table 2 shows the CPU, peripherals, and memory supported by each of the simulator configurations.

Table 2. Supported Hardware Resources†

SIMULATOR CONFIGURATION	CPU	MMU	CACHE	CP15	WRITE BUFFER	BUS INTERFACE UNIT	4GB EXTERNAL MEMORY‡
ARM7 Simulator	ARM7TDMI	NA	NA	NA	NA	NA	✓
ARM9 Simulator	ARM9TDMI	NA	NA	NA	NA	NA	✓
ARM925 Simulator	ARM9TDMI	✓	✓	✓	✓	✓	✓
ARM926EJ-S Simulator	ARM9EJ-S	✓	✓	✓	✓	✓	✓

† ✓=Supported; NA= Not Applicable

‡ The 4GB external memory supported by all the simulator configurations does not account for any latency.

Supported Simulation Features

External Event and Data Simulation

The simulators model the hardware inside a particular device, whereas in real hardware DSP interacts with many other external entities. The interactions between the simulator and these external entities fall into the following two categories:

- Control Signals – These signals trigger activities to the simulator (e.g. interrupts, serial port clocks, serial port synchronization events, etc.)
- Data Values – These are part of an interaction between the simulator and an external entity (e.g. read and write to peripheral registers as a part of I/O memory, serial port data, etc.)

The simulator provides two features – namely ***Pin Connect*** and ***Port Connect*** – for the simulation of these two types of interactions, respectively.

Pin Connect

The Pin Connect tool allows the user to simulate events from the external world.

Generally, control signals from the external entities to the simulator are of most interest to the user. Pin Connect provides a generic way to simulate the control signals from the external entities to the simulator. The control signals and the time at which the signal must be triggered are important. The simulator provides the user with a list of pins corresponding to different control signals. The user must specify all the clock values at which events are to be triggered on a pin using a special format in a file. The user must then connect this file to the pin using the command window, GEL commands, or the Pin Connect plug-in.

The TMS470R1x processor configuration supports the Pin Connect feature. The other configurations do not support this feature.

The interrupt pins that are simulated are FIQ and IRQ. These are level sensitive interrupts.

For information on how to use the Pin Connect feature, please see the Code Composer Studio IDE online help.

Port Connect

The Port Connect feature allows the user to simulate data transfer between the MCU and an external entity that exists in the hardware, but is not modeled in the simulator. Port Connect could also be done on a simulated peripheral, in which case Port Connect data will take precedence over peripheral data.

The transfer of data between the MCU and the external entity (which sits at some particular address in the memory space of the MCU) can happen in the following two ways:

- From external entity to MCU
- From MCU to external entity

To simulate the data transfer from an external entity to the MCU, first, all data values which will be transferred from the external entity to the MCU are put into a file (the details of the file format are discussed in the Port Connect File Format section, beginning on page 4). Then an association is made in the simulator between this file and the addresses at which the external entity sits. This association is called read-mode port-connect. Whenever the simulator must read from the external entity through the addresses associated, it reads from the file one word at a time. To simulate the data transfer from the MCU to an external entity, a file is port-connected in write mode against the addresses of the external entity. Whenever the simulator has to write data to those addresses, it writes into the file one word at a time.

Both TMS470R1x and TMS470R2x simulator configurations support Port Connect.

For information on how to use Port Connect features from the Command window, GEL files, or the Port Connect plug-in, please see the Code Composer Studio IDE online help.

Port Connect Fields

The following fields are required for Port Connect:

- *Address* - the start address of the Port Connect range, the location to which the file needs to be connected.
- *Length* - the Port Connect range in bytes.

  For example, if 0x10000 is the address to connect to a file for read, a length of 16 bytes indicates that the range 0x10000 - 0x1000F will be used for file access.
- *File name* - the file that is used for Port Connect read or write
- *Attribute*s or *mode*s - the read(R)/write(W)/NoRewind(NR) mode.

  The NR mode represents a Read mode with NoRewind option.

Note: In the case of ARM simulators, the *Page* parameter is not used.

Port Connect File Format

A Port Connect file contains one or more lines. Each line contains less than 80 characters and represents one data value. The data is specified in hex format with or without '0x' prefix.

The following is a sample Port Connect file:

```
12345678
0000000f
0000001e
0000002d
0000003a
```

Port Connect Behavior

– The data in the file are interpreted as hexadecimal values only. Each line has a datum. Each datum is considered to be a 32-bit (one word) value.

Read

– In the case of Read (R) mode, by default, once end of file (EOF) is reached, the file is automatically rewound.

– In the case of NoRewind (NR) mode, any further accesses after reaching EOF happen directly to the memory and not to the file. Once EOF is reached, the behavior is as if there is no Port Connect.

– Every CPU read operation will read one full chunk of the data from the port connected file, and the file pointer will be advanced to the next chunk. The read will behave as if the address thrown from the CPU is the address of the least significant byte of the currently pointed chunk in the port connect file. It does not interpret the data based on target endianness.

The data read from the file is coupled to the access type. For example, if a byte access is done on the port-connected region, all bits except the least significant byte (LSB) are masked off and returned. Thus, if the access size were 4 bytes and the current chunk pointer was to 0xabcd1234 in the file, the data seen would be 0xabcd1234. If the access size was 2 bytes, the data seen would be 0c00001234.

The memory window, which includes the port connect memory ranges, will be updated for the addresses for which the values are read from the file.

Port Connect Behavior (continued)

Write

– Every CPU write operation will write one full chunk of the data to the port connected file, and the file pointer will be advanced to the next chunk. The write will behave as if the address thrown from the CPU is the address of the LSB of the current pointed chunk in the port connect file.

For example, if the access size is 4 bytes and the data in the memory is 0xabcd1234, then the data written to the file would be 0xabcd1234. If the access size is 2 bytes, the data written to the file would be 0x00001234.

One can see the memory window, which includes the port connect memory ranges, getting updated for the addresses for which the values are written to the file.

Reset

– All connected file pointers are rewound on reset. All reads beyond reset will start from the beginning of the files. Write connected files are also rewound and the data is written from the beginning of the file after reset.

User Edits

– User read(s) at the port connected memory location will not consume the contents of the port connected file. Similarly, user writes to the port connected memory location will not fill the contents of the port connected file.

For information on how to use Port Connect features from the Command window, GEL files, or the Port Connect plug-in, please see the Code Composer Studio IDE online help.

Simulator Analysis

TMS470R1x and TMS470R2x simulators do not support the Simulator Analysis plug-in.

RTDX

TMS470R1x and TMS470R2x simulators do not support RTDX™ features.

DSP/BIOS

TMS470R1x and TMS470R2x simulators do not support the DSP/BIOS™ feature.

RTDX and DSP/BIOS are trademarks of Texas Instruments.

Detailed Capabilities of Individual Configurations

ARM7 Simulator

The ARM7 simulator models the ARM7TDMI CPU and provides a flat memory. This simulator can be used for any algorithmic verification and for benchmarking applications, as long as the functionality of the peripherals or other modules is not used by the application. All the instructions described in the *TMS470R1x 32-Bit RISC Microcontroller Family User's Guide* (literature number SPNU134) are supported. This includes support of all registers; addressing modes; processor modes (User, FIQ, IRQ, Supervisor, Abort, System, and Undefined); processor states (ARM and Thumb™); branch conditions; etc.The CPU core model is pipeline and cycle-count accurate. All the documented interrupts (Reset, IRQ and FIQ) and all exceptions (User, FIQ, IRQ, Supervisor, Abort, System, and Undefined) are supported.

Both little endian and big endian configurations are supported.

The flat memory model does not support memory maps. Pin Connect and Port Connect features are available to setup external stimuli required by the application for simulation. The list of pins supported for Pin Connect is specified in the Supported Simulation Features section, beginning on page 3.

Updates to memory and watch window happen immediately after an instruction is executed.

ARM9 Simulator

The ARM9 simulator models the ARM9TDMI CPU and provides a flat memory. Like the ARM7 simulator, this simulator is used for algorithmic verification and for benchmarking applications, as long as the functionality of the peripherals or other modules is not used by the application. All the instructions described in the *TMS470R1x 32-Bit RISC Microcontroller Family User's Guide* (literature number SPNU134) are supported. This includes support of all registers; addressing modes; processor modes (User, FIQ, IRQ, Supervisor, Abort, System, and Undefined); processor states (ARM and Thumb); branch conditions; etc. The CPU core model is cycle-count accurate. All the documented interrupts (Reset, IRQ and FIQ) and all exceptions (User, FIQ, IRQ, Supervisor, Abort, System, and Undefined) are supported.

Both little endian and big endian configurations are supported.

The memory model does not support memory maps. Port Connect feature is available, but Pin Connect is not.

Updates to memory and watch window may be delayed longer than expected in the case of the ARM9 simulator. These updates happen after the next instruction has also been executed.

ARM925 Simulator

Models the ARM925 processor that includes:

- ARM9TDMI core
- I-Cache, D-Cache
- Instruction and Data MMUs
- TLBs
- CP15 (System Control Co-processor) Registers
- write buffer

and provides a flat memory of 4 GB external to it.

For details on programming CP15 registers and other details on ARM925, please refer to the *OMAP5910 Technical Reference Manual* (literature number SPRU602).

The core (ARM9TDMI) implementation is as described in the ARM9 simulator section on page 6. The ARM9TDMI core in the ARM925 simulator is cycle-count accurate, but the accuracy of the total cycle count of ARM925 can not be guaranteed.

Thumb is a trademark of ARM Limited.

ARM925 Simulator (continued)

Little endian configuration is the only configuration available.

The memory model does not support memory maps. Port Connect feature is available, but Pin Connect is not.

Updates to memory and watch window may not happen immediately after an instruction is executed. These updates happen after the next instruction has also been executed.

Limitations

There is no IDE support to edit CP15 registers. Programming needs to be done only through application code.

ARM926 Simulator

Models the ARM926EJ-S processor that includes:

- ARM9EJ-S core
- I-Cache, D-Cache
- Instruction and Data MMUs
- TLBs
- CP15 (System Control Co-processor) Registers
- write buffer

and provides a flat memory of 4 GB external to it.

For details on programming CP15 registers and other details on ARM926EJ-S, refer to the *ARM926EJ-S (Rev0) Technical Reference Manual* (literature number ARM DDI 0198B) (downloadable from ARM Limited's web site). The ARM9ES-J core in the ARM926 simulator is cycle-count accurate, but the total cycle count of ARM926 can not be guaranteed.

Little endian configuration is the only configuration available.

The memory model does not support memory maps. Port Connect feature is available, but Pin Connect is not.

Updates to memory and watch window happen immediately after an instruction is executed.

Limitations

There is no IDE support to edit CP15 registers. Programming must be done only through application code.

All Simulators

Table 3 contains a summary of the detailed capabilities of individual configurations.

Table 3. Detailed Capabilities of Individual Configurations†

SIMULATOR CONFIGURATION	PORT CONNECT	PIN CONNECT	LITTLE ENDIAN CONFIGURATION	BIG ENDIAN CONFIGURATION	MEMORY MAPS	MEMORY UPDATES
ARM7 Simulator	✓	✓	✓	✓	–	Immediate
ARM9 Simulator	✓	–	✓	✓	–	Delayed
ARM925 Simulator	✓	–	✓	–	–	Delayed
ARM926EJ-S Simulator	✓	–	✓	–	–	Immediate

† ✓=Supported; – = Not Supported

Note: When using C programs built for 16-bit (Thumb) code, any further execution after "Restart" or "Reload" from Thumb state leads to unpredictable behavior of the simulator. To avoid this, users are advised to do a "Reset" before giving "Reload" or "Restart" from Thumb state. There is no similar problem with ARM state.

Configuring the Simulator

The simulators are not configurable.

Users are advised not to edit/modify the supplied configuration files, arm9tdmi_little.cfg and arm9tdmi_big.cfg.

Performance Numbers

Table 4 shows the performance numbers of the simulator for the two configurations. These numbers were gathered on a 1GHz Intel™ Pentium™ IV PC with 256MB of RAM. The application used for measurement in both cases is the Reed-Solomon encoding and decoding application from a standard benchmarking suite.

Table 4. Performance Numbers of the TMS470 Simulator

SIMULATOR CONFIGURATION	SIMULATOR SPEED (IN KILOCYCLES/SECOND)
TMS470R1x	1062
TMS470R2x	952

Cycle Accuracy

The CPU model is cycle-count accurate. Cycle accuracy cannot be guaranteed in the case of cached cores, that is, the ARM925 and ARM926 simulators. Flat memory accesses are single-cycle accesses for all the ARM simulators.

Related Documentation

TMS470R1x User's Guide (literature number SPNU134) describes the TMS470R1x RISC micrcontroller, its architecture (including registers), ICEBreaker module, interfaces (memory, coprocessor, and debugger), 16-bit and 32-bit instruction sets, and electrical specifications.

TMS470R1x Assembly Language Tools User's Guide (literature number SPNU118) describes the assembly language tools (assembler, linker, and other tools used to develop assembly language code), assembler directives, macros, common object file format, and symbolic debugging directives for the TMS470R1x devices.

TMS470R1x Optimizing C Compiler User's Guide (literature number SPNU119) describes the TMS470R1x C compiler. This C compiler accepts ANSI standard C source code and produces assembly language source code for the TMS470R1x devices.

TMS470R1x C Source Debugger User's Guide (literature number SPNU124) describes the TMS470R1x emulator and simulator versions of the C source debugger interface. This book discusses various aspects of the debugger interface, including window management, command entry, code execution, data management, and breakpoints. It also includes a tutorial that introduces basic debugger functionality.

TMS470 Code Composer Studio Online Help is accessible through the Code Composer Studio Integrated Development Environment (IDE).

OMAP5910 Technical Reference Manual (literature number SPRU602) describes the features, architecture, memory maps, and software compatibility of the OMAP5910 dual-core processor.

ARM926EJ-S (Rev0) Technical Reference Manual (literature number ARM DDI 0198B) (downloadable from ARM Limited's web site)

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and application s using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different form or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments
Post Office Box 655303 Dallas, Texas 75265