SPRU601B – JULY 2002 – REVISED MAY 2003

- **Included in Code Composer Studio™ Integrated Development Environment (IDE) for the OMAP™ Platform**
- **TMS320C55x™ Digital Signal Processor (DSP) Subsystem Simulation**
- **TI925T (MPU) ARM9TDMI™ Subsystem Simulation**
- **ARM926 (MPU) ARM9EJ-S™ Subsystem Simulation**
- **Shared System Simulation**
- **Interprocessor Communication**
- **Synchronous Run and Halt**
- **Profile Clock Support for MPU and C55x™ DSP**
- **Interrupt Configurability**
- **Configurable Clock Ratios**

Description

The OMAP3.1 platform simulator, available within the Code Composer Studio IDE for the OMAP Platform, simulates the OMAP3.1devices. In addition, Code Composer Studio IDE for the OMAP Platform supports the TMS320C55x DSP stand-alone Simulator, which consists of the C55x DSP subsystem and the shared system. Another configuration supported is the ARM926EJ-S stand-alone little endian simulator. The ARM926EJ-S is part of the OMAP 3.2 platform. Table 1 lists the platforms supported, with the corresponding configuration to be selected under the Import Configuration menu of Code Composer Studio Setup.

Table 1. Processors Supported by the OMAP Platform Simulator

PROCESSOR	CODE COMPOSER STUDIO IDE IMPORT CONFIGURATION
OMAP15xx/OMAP59xx	OMAP3.1 Platform Simulator
OMAP3.1 DSP Subsystem	OMAP3.1 DSP Subsystem Simulator
ARM926EJ-S	ARM926EJ-S Simulator, Little Endian

Considerations for Simulator Selection

DSP/BIOS™ is **NOT** supported on the OMAP3.1 platform simulator. To test C55x code that uses DSP/BIOS, use the C55x DSP subsystem stand-alone simulator configuration.

Although there are many devices in the OMAP platform, only those listed in Table 1 are currently supported by the simulator. Because of similarities among the devices, users developing code for an unsupported device can select the configuration most similar in terms of the CPU and on-chip peripherals. Table 2 shows the configuration that should be selected for use with some of the unsupported devices.

Table 2. Configurations for Use With Unsupported OMAP Devices.

UNSUPPORTED HARDWARE DEVICE (PROCESSOR)	NEAREST SIMULATOR CONFIGURATION
OMAP310	OMAP3.1 Platform Simulator
OMAP710	OMAP3.1 Platform Simulator
OMAP1610 (MPU Subsystem)	ARM926EJ-S Simulator, Little Endian
OMAP1610 (DSP Subsystem)	OMAP3.1 DSP Subsystem Simulator

Only the MPU subsystem of the OMAP710 is simulated. Also, the second level interrupt handler is not modeled in the simulator; all the interrupts are directly connected to the first level interrupt handler. For both OMAP310 and OMAP710, use only the MPU side of the OMAP3.1 platform simulator. The interrupts will have to be remapped. See the section on Changing Default Interrupt Mapping, starting on page 13, for more information on how this can be achieved. The ARM926EJ-S stand-alone simulator provides simulation of the MPU side of the OMAP1610. All external memory accesses from ARM926EJ-S are handled by a 4GB flat memory space. Simulation on the DSP subsystem of the OMAP1610 is supported by the OMAP3.1 DSP subsystem simulator.

Supported Hardware Resources

OMAP3.1 Platform Simulator

The OMAP3.1 platform simulator includes the models for the DSP and MPU subsystems, as well as some of the OMAP-specific modules outside these subsystems. The simulator enables true, synchronized, dual-processor simulation with the capability to do interprocessor communication via a shared memory, API, and mailboxes. These components are described below. For more information, please see the *OMAP1510 Multimedia Processor Technical Reference Manual* (literature number SWPU030).

Cores

The OMAP3.1 platform simulator has both the MPU and C55x DSP subsystem simulators running together with shared memory access.

Memory

The OMAP3.1 Platform has three memory interfaces, namely, the EMIF Fast (SDRAM), the IMIF, and the EMIF Slow interfaces. External memory is not a part of OMAP. It sits outside the system and interacts with OMAP via the EMIF Fast and EMIF Slow interfaces. Internal memory is part of OMAP platform. All memories are modeled as flat memory. The memory interfaces (EMIF Fast/IMIF/EMIF Slow) addresses cycle count accuracy by inserting appropriate delay cycles. The simulator also provides the capability to directly access/edit all regions of the memory via debug reads and writes from the corresponding processor Code Composer Studio windows.

Modules Supported

Table 3 lists the modules that are part of the OMAP3.1 Platform and whether or not they are modeled as part of the OMAP3.1 platform simulator. Details on the degree to which each module is modeled is available in the following section, Functional Capabilities.

Table 3. Modules Supported by the OMAP3.1 Platform Simulator

MODULE	MODELED	NOT MODELED
C55x DSP Subsystem		
C55x CPU	X	
Cycle Accurate Memory Subsystem	X	
DMA Controller	X	
TIPB Bridge	X	
I-Cache	X	
MPUI	X	
EMIF	X	
Hardware Accelerators (3)	X	
OMAP DSP Timer1	X	
OMAP DSP Timer2	X	
OMAP DSP Timer3	X	
OMAP DSP Watchdog Timer	X	
DSP Interrupt I/F		X
Interrupt Controller	X	
MPU Subsystem		
ARM9TDMI Core	X	
I-Cache	X	
D-Cache	X	
MPU MMU	X	
CP15 Registers	X	
Write Buffer	X	
System Bus Interface	X	

Table 3. Modules Supported by the OMAP3.1 Platform Simulator (Continued)

MODULE	MODELED	NOT MODELED
OMAP MPU Timer1	X	
OMAP MPU Timer2	X	
OMAP MPU Timer3	X	
OMAP MPU Watchdog Timer	X	
Interrupt Controller	X	
Shared System		
DSP MMU	X	
MPUI I/F	X	
System DMA	X	
MailBoxes	X	
UART		X
GPIO		X
Clock and Reset Module	X	
LCD Controller	X	
System TIPB Bridge	X	
Traffic Controller	X	
Endianism Conversion Block	X	

Functional Capabilities

C55x DSP

OMAP simulation supports the following C55x DSP modules:

- C55x CPU revision 2.1
- Instruction Cache
- Internal Memories
- Generic Direct Memory Access (gDMA)
- TIPB Bridge
- External Memory Interface (EMIF)
- MPU Interface (MPUI)
- Hardware Accelerators

For more details, please refer to the *TMS320C55x Instruction Set Simulator Technical Overview* (SPRU599)

MPU

The full TI925T is modeled based on the ARM9T rev4 core and the Cache-MMU-CP15 Specification 1.3. The OMAP simulator includes the following MPU modules:

- ARM9TDMI Core
- I-Cache and D-Cache
- MMU
- CP15 Registers
- Write Buffers
- System Bus Interface

DSP Private Peripherals

Modeled:

- DSP-OS-Timers, Watchdog Timer and Interrupt Handler
- Reset of system via Watchdog Timer Reset is not modeled

The default OMAP peripheral interrupt connections to the C55x Interrupts (OMAP Level 1 Interrupts) are shown in Table 4. See the section titled Simulator Configuration Files, starting on page 12, to learn how to reconfigure these interrupts.

MPU Private Peripherals

Modeled:

- MPU-OS-Timers, Watchdog Timer and Interrupt Handler
- Reset of system via Watchdog Timer is not supported

DSP MMU

Modeled:

- DSP Memory Management Unit, including the WTL and the no-WTL modes
- Interrupt from the MMU to the MPU
- Abort generation

Not modeled:

- DSP access to the MMU registers
- Prefetch

System DMA

Modeled:

- IMIF port
- SDRAM port
- EMIF port
- MPUI TIPB port
- TIPB port
- Local bus
- Nine generic channels
- Interrupt generation of block interrupts, half-frame interrupts, frame interrupts, timeout and drop interrupts, and last frame interrupts for each channel.
- All four addressing modes (Constant, Post-Incremented, Single-Indexed, and Double-Indexed) for both source and destination
- Burst mode (4 and 8) for each channel for both source and destination
- Packing for both source and destination for each channel
- Channel reloading based on auto-init, repeat, and end-prog bits
- All data types (8,16, and 32) for each channel

- Aborts

Not Modeled:

- HSAB ports

Traffic Controller

Modeled:

- Handles access from MPU, DSP, and System DMA to IMIF, EMIFS and EMIFF
- "Round Robin" arbitration mechanism
- Burst accesses for all the interfaces mentioned
- Dynamic change in endianism

Not Modeled:

- Local bus and high speed access bus interfaces
- Dynamic priority scheduling
- Cycle count accuracy for accesses is not ensured

CAUTION: Memory belonging to the address range 0x5000100 to 0x50001ff should not be used by the program. It is reserved for internal usage by the OMAP simulator. This is a limitation of the simulator.

MPUI Interface

This module channels accesses from MPU and System DMA to the MPUI of the C55x.

Modeled:

- Arbitration in the case of conflicts (priorities are programmable)
- Byte swapping
- Accesses of all sizes (8/16/32)
- Aborts
- HOM and SAM modes

Not Modeled:

- Local interface bus

System TIPB Bridge

Modeled:

- Accesses to all modeled peripherals
- Read and write to all un-modeled peripherals supported as memory read-writes
- Read/Write to peripherals obeys the strobe-ready protocol
- Two instances of System Rhea bridge - public and private
- Posted-write from MPU
- Arbitration between MPU and DMA
- Access factor mechanism
- Timeout feature
- Aborts

Mailboxes

Modeled:

- Mailbox interrupts (enables inter-processor communication)
- ARM2DSP and DSP2ARM mailboxes

Clock and Reset Manager

This peripheral is only partially modeled.

Modeled:

- Reset from MPU to DSP

LCD Controller

This peripheral is only partially modeled.

Modeled:

- Reads and write to registers
- The mode that does **not** use Dither Logic (16 bits/pixel active mode)

For modes that use Dither Logic, the simulator generates a log file containing the index of the palette value input to the Dither Logic.

Default Interrupt Mappings

The default connections of the OMAP peripheral interrupts are described below. These default connections can be remapped. Please refer to the section on Simulator Configuration Files, beginning on page 12 of this document to understand how this can be done.

Table 4 gives the default OMAP peripheral interrupt connections to the C55x DSP interrupts.

Table 4. Default Interrupt Connections to the C55x

INTERRUPT NAME	IEN BIT NUMBER
TIMER1_INT	IEN7
TIMER2_INT	IEN15
TIMER3_INT	IEN16
WDTIMER_INT	IEN13
INTH_IRQ	IEN5
INTH_FIQ	IEN4
MGS_INTDMA0	IEN17
MGS_INTDMA1	IEN18
MGS_INTDMA2	IEN19
MGS_INTDMA3	IEN20
MGS_INTDMA4	IEN21
MGS_INTDMA5	IEN22
ARM2DSP_MAILBOX_INT1	IEN8
ARM2DSP_MAILBOX_INT2	IEN9

Table 5 below gives the mapping of IEN bit numbers and the corresponding DSP Soft Interrupt number (Level 1 Interrupt number). These interrupt numbers are to be used in the configuration file for reconfiguring the interrupts. Reset and NMI (Interrupts 1 and 0) cannot be configured.

Table 5. IEN Bit Number to Level 1 Interrupt Number Mapping

INTERRUPT NUMBER	IEN BIT NUMBER
2	IEN02
3	IEN16
4	IEN03
5	IEN11
6	IEN19
7	IEN23
8	IEN05
9	IEN17
10	IEN06
11	IEN07
12	IEN12
13	IEN13
14	IEN04
15	IEN22
16	IEN08
17	IEN18
18	IEN09
19	IEN20
20	IEN21
21	IEN14
22	IEN15
23	IEN10

CAUTION: The default interrupt mapping is not the same as OMAP1510 interrupt mapping. For OMAP1510, the user will need to remap interrupts to match those of OMAP1510.

The connections of the peripheral interrupts to the MPU interrupt handler input pins are shown in Table 6.

Table 6. MPU Connections to the Interrupt Handler Input Pins

INTERRUPT NAME	INTERRUPT LINE NO.
TIMER1_INT	IRQ_26
TIMER2_INT	IRQ_30
TIMER3_INT	IRQ_16
WDTIMER_INT	IRQ_27
SYSTEM_DMA_INT0	IRQ_19
SYSTEM_DMA_INT1	IRQ_20
SYSTEM_DMA_INT2	IRQ_21
SYSTEM_DMA_INT3	IRQ_22
SYSTEM_DMA_INT4	IRQ_23
SYSTEM_DMA_INT5	IRQ_24
DSP2ARM_INT1	IRQ_10
DSP2ARM_INT2	IRQ_11
TC_ARM_ABORT_INT	IRQ_9

The TC_ARM_ABORT_INT handles all the aborts coming to the Traffic Controller from the associated interfaces. It is connected to the MPU Interrupt Controller input pin and can be reconfigured.

Modules Not Supported in this Version of the OMAP Simulator

- GPIO
- UART
- Local Bus Controller, HSAB Controller
- External Peripheral Interface

CAUTION:

- This simulator is not capable of full board simulation, neither is it a full device simulator.
- There is no mechanism for hooking up an external system/peripheral to the OMAP3.1 simulator. Pin and port connects are not supported for this purpose.
- Cycle count accuracy for external memory accesses is not guaranteed.
- Reset of the system by means of a Watchdog Timer reset is not supported.

ARM926EJ-S Stand-alone Simulator

The ARM926EJ-S stand-alone model is a uni-processor model with a 4GB flat memory space.

Core

The ARM926EJ-S has a Harvard architecture ARMv5 ARM9EJ-S core.

Memory

The ARM926EJ-S stand-alone simulator comes with a 4GB flat memory space. There is no memory latency for any accesses. The profiler provides the total number of cycles taken to execute at the ARM926EJ-S boundary.

Modules Supported

Table 7 lists the modules that are part of the ARM926EJ-S simulator and whether or not they are modeled.

Table 7. Modules Supported by the ARM926EJ-S Simulator

MODULE	MODELED
ARM9EJ-S Core	X
I-Cache	X
D-Cache	X
MMU	X
CP15 Registers	X
Write Buffer	X
TLB	X

CAUTION:

- This simulator is not capable of full board simulation, neither is it a full device simulator.
- There is no way to connect other peripherals to the ARM926EJ-S stand-alone simulator. Pin and port connects are not supported for this purpose.
- Only a zero wait state external memory is supported.

Supported Simulation Features

OMAP3.1 Platform Simulator

Peripheral Support

This simulation module, along with the full C55x DSP and MPU subsystems, supports the three Timers, the Watchdog Timer, and the Interrupt Handler for both the DSP and MPU subsystems. Among the shared peripherals, mailboxes are modeled to facilitate interprocessor communication. The System DMA (SDMA) of the OMAP3.1 platform is supported. All the channels except the LCD controller channel are modeled. All ports except HSAB and Local Bus ports are modeled. The MPUI allows MPU and SDMA to access the internal memory of the C55x. This is essential for booting the DSP.

Interprocessor Communication

The OMAP3.1 platform simulator enables interprocessor communication via mailboxes, Shared Memory, and the MPU Port Interface (MPUI). Four mailbox interrupts (two from MPU to DSP and two from DSP to MPU) are supported. The MPUI can be used by the MPU to access DSP internal memory and shared peripherals.

Synchronous Run and Halt Capability

This release allows the user to run the system in a synchronized manner by specifying a Synchronous Run command from the Parallel Debug Manager (PDM). Once a Synchronous Run is issued, the system halts if either a break point is hit or the user issues a synchronous halt from the PDM. This ensures that the system remains synchronized. The user can also opt to run one of the subsystems by issuing a Run command from the corresponding Code Composer Studio window (Asynchronous Run). The simulator does not guarantee synchronization of the system if a combination of Synchronous and Asynchronous commands are issued.

Profile Clock Support for Both MPU and C55x DSP

The cycle counts on each of the processors can be profiled. Profile clocks are enabled for both the MPU and C55x subsystems.

Interrupt Configurability

The OMAP3.1 simulator comes with default interrupt mappings. The user has the ability to reconfigure the connections of interrupts by modifying the appropriate section in the configuration file.

Configurable Clock Ratios

The clock ratio (MPU:Shared System:C55x DSP Subsystem) can be configured by the user at startup by modifying the OMAPSystem.cfg file.

ARM926EJ-S Stand-alone Simulator

Profile Clock Support

The cycle counts of the ARM926EJ-S can be profiled. The profile clock can be enabled and the cycle counts viewed for each step, or for the entire run.

Caution

- Pin and port connect features are not supported.
- Reset and restart does not alter memory contents.

Differences Between Simulator and Silicon

For details on the differences between the OMAP instruction set simulator and the silicon, please see the "Differences Between Simulator and Silicon" sections of the *TMS320C55x Instruction Set Simulator Technical Overview* (literature number SPRU599) and the *TMS470 Instruction Set Simulator Technical Overview* (literature number SPNU006).

Configuring the Simulator

OMAP3.1 Platform Simulator

The OMAP3.1 platform simulator provides the user with the flexibility to configure some features by means of configuration files. The Code Composer Studio installation comes with three OMAP platform-specific configuration files, namely, MGS3.cfg, ARM925.cfg, and OMAPSystem.cfg. This section describes how the user can make changes to these configuration files to configure the platform as per the requirements.

MGS3.cfg

This configuration file is for the C55x DSP subsystem. The user can configure the initial state of the C55x DSP by modifying the INIT_STATE parameter. If the INIT_STATE is set to IN_RESET, then the C55x does not start executing the code until the MPU sends a reset to the C55x by programming the clock and reset module. If the user is just interested in C55x behavior after the RESET, he can change INIT_STATE to RESET_DONE. When the INIT_STATE is set to RESET_DONE, the C55x starts executing immediately. For example,

```
MODULE C55x;
      PHASE                p3;
      INIT_STATE           IN_RESET;
END C55x;
```

NOTE: Only the entries in bold are to be edited.

Standalone DSP applications will not run on the OMAP3.1 platform simulator, as the initial state of the DSP side is "IN_RESET" and the DSP waits for the ARM to bring it out of reset. Therefore, for standalone DSP applications, use the OMAP3.1 DSP subsystem simulator configuration. This configuration uses the OMAP_MGS3.cfg file, in which INIT_STATE is set to RESET_DONE.

ARM925.cfg

This configuration file is for the MPU subsystem. The user should not change this file.

OMAPSystem.cfg

This configuration file can be used for the following tasks:

- change the clock ratio at startup
- change the default interrupt mapping of the peripherals
- change the MMU endianism.

Changing the Relative Clock Ratios

The clock ratios can be changed before starting the Code Composer Studio IDE. In the configuration file, only relative clock widths (not clock frequencies) can be specified. The user can also specify fractional clock widths in the form of numerator/denominator, prefixed with an "f" or an "F". For example, to specify a width ratio of 2/3, it should be specified as either f2/3 or F2/3. **Fractional clock widths cannot be specified in decimal notation**.

Within each subsystem, the CPU and device clocks can differ, so the user need not always specify the same ratio for CPU and device clocks. For example, if the frequencies of C55x, MPU, and the shared system are 200 MHz, 150 MHz, and 100 MHz, respectively, the relative frequencies, normalized with respect to the fastest clock will be 1, 3/4, and 1/2. The relative clock widths will be 1, 4/3, and 2. This is what needs to be specified in the configuration file.

In the example below, the device clock widths are assumed to be the same as the corresponding processor clock width.

```
MODULE PROCESSORS;
      MODULE ARM925;
                CPU_CLOCK F4/3;
                DEV_CLOCK F4/3;
      END ARM925;
      MODULE MEGASTAR3;
                CPU_CLOCK 1;
                DEV_CLOCK 1;
      END MEGASTAR3;
      MODULE SHARED;
                DYNAMIC_LIBRARY     shared_system.dll;
                INIT_FUNC           initSharedSystem;
                DEV_CLOCK 2;
      // Remaining entries should not be changed
      END SHARED;
END PROCESSORS;
```

NOTE: Only the entries in bold are to be edited to change clock ratios.

Changing Default Interrupt Mapping

The default interrupt mapping can be changed by the user by editing the relevant sections of the OMAPSystem.cfg file.

For the C55x DSP subsystem, the following interrupts can be reconfigured:

TIMER1_INT, TIMER2_INT, TIMER3_INT, WDTIMER_INT, INTH_IRQ, INTH_FIQ, ARM2DSP_MAILBOX_INT1, ARM2DSP_MAILBOX_INT2, MGS_INTDMA0, MGS_INTDMA1, MGS_INTDMA2, MGS_INTDMA3, MGS_INTDMA4, and MGS_INTDMA5.

The destination of these interrupts can be either some interrupt pin of the C55x interrupt controller (0-15) or any of the C55x soft interrupts (2-23). Refer to Table 5 for mapping of OMAP Level 1 interrupt numbers and IEN Bit numbers. In the example below, TIMER1_INT is being remapped to interrupt handler interrupt 7, and ARM2DSP_MAILBOX_INT2 to C55x soft interrupt 5.

```
MODULE DSP_CROSSBAR;
      MODULE MGS_RHEA_INTERRUPTS;
                ARM2DSP_MAILBOX_INT2 5; //Connect to C55x soft interrupt 5
      END MGS_RHEA_INTERRUPTS;

      MODULE DSP_INTERRUPT_HANDLER;
                TIMER1_INT1 7; //Connect to INT Handler Input pin 7
      END DSP_INTERRUPT_HANDLER;

END DSP_CROSSBAR;
```

NOTE: Only the entries in bold are to be edited. For interrupt names, refer to Table 4

For the MPU subsystem, the following interrupts can be reconfigured:

TIMER1_INT, TIMER2_INT, TIMER3_INT, WDTIMER_INT, INTH_IRQ, INTH_FIQ, ARM2DSP_MAILBOX_INT1, ARM2DSP_MAILBOX_INT2, SYSTEM_DMA_INT0, SYSTEM_DMA_INT1, SYSTEM_DMA_INT2, SYSTEM_DMA_INT3, SYSTEM_DMA_INT4, SYSTEM_DMA_INT5, TC_ARM_ABORT_INT.

The destination of these interrupts can be any of the interrupt handler pins (0-31). In the following example, WDTIMER_INT is being remapped to interrupt handler interrupt 25.

```
MODULE ARM_CROSSBAR;
      MODULE ARM_INTERRUPT_HANDLER;
                WDTIMER_INT 25; //Connect to INT Handler Input pin 25
      END ARM_INTERRUPT_HANDLER;

END ARM_CROSSBAR;
```

NOTE: Only the entries in bold are to be edited. For interrupt names, refer to Table 6.

Changing MMU Endianism

The DSP MMU endianism and byte/word swap can be controlled via the OMAPSystem.cfg file. For example, to enable DSP MMU endianism block and to set the byte/word swap to word swap, set the fields as shown.

```
MODULE DSP_MMU_ENDIANISM;
      DSP_MMU_ENDIANISM_ENABLE 1;
      WORD_NBYTE_SWAP 1;
END DSP_MMU_ENDIANISM;
```

For DSP MMU endianism, ’1’ indicates enabled and ’0’ indicates disabled.

For byte/word swap, ’0’ indicates byte swap enabled and ’1’ indicates word swap enabled.

NOTE: Only the entries in bold are to be edited.

ARM926EJ-S Stand-alone Simulator

Armulate.cfg

This file, available in the ’drivers’ directory of the Code Composer Studio installation, has configuration options set for the ARM926EJ-S stand-alone simulator to initialize. Other configuration files are found in the ’ARM_config’ sub-directory of the ’drivers’ directory.

NOTE: Users should NOT edit these configuration files.

Debugging Using the OMAP3.1 Platform Simulator

Once setup of Code Composer Studio for OMAP3.1 platform simulator is completed, starting Code Composer Studio IDE brings up the Parallel Debug Manager (PDM), as shown in the figure below. From the PDM, the user can open individual processor windows and issue synchronous commands.

Figure 1. Parallel Debug Manager Window

The PDM lets the user open two Code Composer Studio windows at once – one for the MPU and the other for C55x. It also allows the user to issue synchronous commands such as Synchronous Run, Synchronous Halt, Synchronous Step and Reset. The semantics of the various commands are as follows.

Synchronous Halt:

Halts the system synchronously.

Synchronous Run:

Runs the system synchronously until some breakpoint is hit on any of the processors. Both the processors are clocked, and the clock synchronization of the system is maintained during this run.

CAUTION: When a user starts running the system synchronously (Synchronous Run), a Synchronous Halt from the PDM must be used in order to stop the system.

Locked Step:

Clocks the system until the MPU completes one instruction. The system then halts synchronously. Synchronization of the system is ensured.

CAUTION: Synchronous step (SYNC-STEP) behavior on the OMAP simulator:
(a) If both the MPU and C55x DSP are in **assembly mode/mixed mode**, then SYNC-STEP is equivalent to Instruction Step on the MPU.
(b) If one or both of the processors are in **source mode**, the user is advised against using Locked Step. Interrupts may be lost.

Step Over:

The semantics of Step Over are the same as that of Locked Step (see above).

Reset:

Resets the entire system.

CAUTION: If the MPU is in Thumb™ mode, reset of the MPU (from PDM or from the MPU Code Composer Studio window) does not work, and the PC becomes stuck at the same value. This is an MPU simulator bug. In this case, the Code Composer Studio IDE needs to be restarted.

Thumb is a registered trademark of ARM Limited.

Note: The following commands from the PDM are inapplicable and not supported: Load Program, Load Symbol, Reload Program, Run Free, Animate and Step Out. The user is advised against using these commands.

Asynchronous commands:

In addition to the synchronous commands above, commands can be issued from individual processor Code Composer Studio windows. These are the asynchronous commands and relate to the processor from whose window they have been issued. For example, a Run command from the MPU Code Composer Studio window is an Asynchronous Run and will run only the MPU and shared subsystems.

CAUTION: If the user issues a Synchronous Run and, after halting the system, issues an asynchronous call (step/run) from either of the processors, from that point onward synchronization is lost. There is no guarantee about the accuracy of the simulation. Interrupts can be lost. *For accurate simulation, the user is advised not to use a combination of synchronous and asynchronous commands.*

Running and Debugging Programs on the OMAP3.1 Platform Simulator:

Setup Code Composer Studio IDE as specified for the OMAP3.1 platform simulator. If necessary, change the configuration files (see Simulator Configuration Files section starting on page 12). Start the Code Composer Studio IDE.

To load a program, the user must open the corresponding processor Code Composer Studio window and load the coff executable as in a single processor scenario. For example, to load a test case on the MPU, the MPU Code Composer Studio window has to be opened from the PDM and the test case has to be loaded from this window.

Set breakpoints at appropriate places in the code. Issue a synchronous run to the system from the PDM. The system will run synchronously and halt whenever a break point is hit

CAUTION: All breakpoints are Global breakpoints. When a breakpoint is hit, the entire system halts. If the breakpoint was hit during a synchronous run, synchronization is maintained. **Local breakpoints are not supported**.

For breakpoints set from the C55x Code Composer Studio window, there can be a skid when the system halts, and the blue bar may move further than the point where the breakpoint was hit. This is because the MPU is an instruction-callable model, as opposed to the C55x, which is cycle-callable. To ensure synchronization, the system is clocked until the next immediate MPU instruction boundary is reached. Any breakpoints that fall within the skid region of the C55x code are ignored.

To test the correctness of the code, load the test cases and set a breakpoint at the termination of the MPU-side executable. Issue a synchronous run to the system from the PDM. When the system hits the breakpoint and halts, check for correctness of the test case.

If the results are not as expected, set breakpoints at potential failure points of the code and again issue a synchronous run. The system state can be analyzed at each of the breakpoints and the problem can be localized. To maintain synchronization, use synchronous runs to continue after the system has halted (on hitting a breakpoint).

If the program uses only one of the subsystems (MPU or C55x DSP) or if there is no inter processor communication, then the user can use asynchronous runs to test and debug the code. For example, if the user has a program that runs on the MPU side alone and does not use the C55x DSP subsystem, the program can be loaded on the MPU and MPU can be run asynchronously.

CAUTION: Memory belonging to the address range 0x5000100 to 0x50001ff should not be used by the program. It is reserved for internal usage by the OMAP simulator. This is a limitation of the simulator.

Performance Numbers

OMAP 3.1 Platform Simulator

The OMAP3.1 platform simulator runs at approximately 25KCS (kilo MPU cycles per host CPU second). Benchmarking was done on a 1GHz Intel™ Pentium™ CPU with 128MB RAM, using sample programs that involved IPC and shared memory access. The clock ratio used was 1:1:1 (MPU:Shared System:DSP).

ARM926EJ-S Stand-alone Simulator

The ARM926EJ-S stand-alone simulator runs at approximately 920 KCS (kilocycles per host CPU second). Benchmarking was done on a 1GHz Intel™ Pentium™ CPU with 128MB RAM, using Reed-Solomon and EEMBC® Reed-Solomon applications.

Troubleshooting

ARM926EJ-S Stand-alone Simulator

When the ARM926EJ-S simulator is unable to initialize correctly, it displays an error message to highlight the problem. The following error messages are generated:

1. "Armulate.cfg if faulty"

   Meaning: Armulate.cfg is absent or non-readable.

   Action: Check that the Armulate.cfg file is present in the ’drivers’ directory and whether it is readable.

2. "Armulate.sdi not found"

   Meaning: armulate.sdi, one of the dlls required by the ARM926EJ-S is absent.

   Action: Check if armulate.sdi file is present in the ’drivers’ directory.

3. "ARM unable to initialize"

   Meaning: There is an initialization error.

   Action: Check to see that all of the following conditions are satisfied:
   - The ARM_dll directory is present in the ’drivers’ directory.
   - The ARM_config directory is present in the ’drivers’ directory.
   - No file in the ARM_config directory has been modified.
   - The Armulate.cfg file in the ’drivers’ directory has not been modified.

Please replace any configuration files that are missing or have been changed with the original installation files.

Related Documentation

OMAP Multimedia Processor Technical Reference Manual (literature number SWPU030) describes the setup, components, and features of the OMAP1510 multimedia processor.

OMAP310 Multimedia Processor Technical Reference Manual (literature number SWPU031) describes the setup, components and features of the OMAP310 multimedia processor.

OMAP710 Multimedia Processor Technical Reference Manual (literature number SWPU012) describes the setup, components and features of the OMAP710 multimedia processor.

OMAP710 Multimedia Processor Datasheet (literature number SWPS004)

TMS320C55x Instruction Set Simulator Technical Overview (literature number SPRU599) provides an overview of the capabilities of the TMS320C55x Simulator.

TMS320C55x DSP CPU Reference Guide (literature number SPRU371) describes the CPU of the TMS320C55x fixed-point DSPs, including architecture, registers and operation.

TMS320C55x DSP Peripherals Reference Guide (literature number SPRU317) describes the peripherals, interfaces, and related hardware that are available on TMS320C55x DSPs.

TMS320C55x DSP Algebraic Instruction Set Reference Guide (literature number SPRU375) contains information about the instructions used for all types of operations (arithmetical, bit manipulation, logical, move, and program control).

TMS320C55x DSP Mnemonic Instruction Set Reference Guide (literature number SPRU374) contains information about the instructions used for all types of operations (arithmetical, bit manipulation, logical, move, and program control).

TMS320C55x Instruction Set Simulator Technical Overview (literature number SPRU599)

TMS470 Instruction Set Simulator Technical Overview (literature number SPNU006)

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 applications 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 that 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 from 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.

Mailing Address:

Texas Instruments
Post Office Box 655303
Dallas, Texas 75265