

March 30, 2001

## *Running C6x Applications from External Memory*

**Innovative Integration Technical Staff**

*<http://www.innovative-dsp.com>*

**This note is intended for help application programmers using the Zuma Toolset to modify target application programs to execute directly from SDRAM or SBSRAM.**

---

### *The Problem*

#### **ONCHIP PROGRAM AND DATA SPACE IS LIMITED**

The C6x01 processor family provides 128KB of onchip memory. The generic command file provided in the Zuma Toolset used to link the example programs (`generic.cmd`) locates all application code and data into this onchip memory to illustrate optimal execution performance.

However, 64KB of code space and 64 KB of data space is sometimes insufficient for larger, user-written target applications. A means of efficiently utilizing SDRAM or SBSRAM for program and data storage is needed.

---

### *The Solution*

#### **CACHE CONTROLLER EFFICIENTLY MANAGES ONCHIP MEMORY**

The C6x01 processor family features an onchip cache controller. When enabled, all 128 KB of onchip memory is efficiently managed by the cache circuitry. Application programs can may be modified to run directly from external memory (SBSRAM or SDRAM). When executing from external memory with the cache controller enabled, large applications can achieve nearly the same performance as applications running entirely onchip.

In order for application programs to run from external memory, they must be linked using an appropriate command (`.cmd`) file. The `cached.cmd` file, included in the Zuma Toolset, directs all program and data into SDRAM. Re-linking an application using this

---

---

## The Solution

alternative command file is all that is necessary to redirect all application code and data to run from external SDRAM memory.

However, in order to exploit the cache controller, two minor modifications to application initialization code are required. Specifically, the interrupt vector table must be relocated into external memory and the cache controller must be enabled.

To relocate the interrupt vector table, use the Zuma `RelocateVectorTable` function. This function consumes a pointer to the new base address of the system vector table. The new vector table must reside in external memory. The `cached.cmd` command file reserves 0x200 bytes at the beginning of SDRAM for this purpose. Typically, the base address of SDRAM (0x2000000 on an M6x) is typically passed to the `RelocateVectorTable` function in application programs.

Interrupt vectors are dynamically installed and removed within Zuma application programs using the `InstallIntVector` and `DeinstallIntVector` functions. After relocation of the vector table, calls to these functions will install or remove interrupt vectors into the relocated table.

Following vector table relocation, the processor's cache circuitry may be enabled. The Zuma `EnableCache` function is used to perform this function. Following this call, the cache controller will efficiently regulate use of all onchip memory. Both program memory fetches and application data fetches are managed.

Once the onchip memory is under the control of the caching circuitry, explicit accesses to onchip memory are forbidden. It is illegal to attempt to place strategic code and data fragments into onchip memory when using the cache controller.