

External Programming of the TMS320C64x EDMA for Low Overhead Data Transfers

Sébastien Tomas
Wireless Infrastructure

ABSTRACT

This application report details a mechanism by which an external slave device can directly initiate direct memory access (DMA) transfers with a TMS320C64x™ digital signal processor (DSP) without processor intervention. The mechanism utilizes the RAM-based architecture of the TMS320C64x enhanced direct memory access (EDMA) controller, static DMA transfers and DMA chaining to “effectively” allow an external slave device to dynamically program the internal EDMA controller of the DSP.

This technique can be used to implement effective, low processor overhead transfers of variable-length data between an external slave device and the internal memory of the DSP. These types of transfers often occur in wireless basestations, between an external receive chip-rate processing device and a controlling DSP. In order to implement this technique, the external slave device must have enough processing capability to generate the required fields of an EDMA transfer parameter entry.

Contents

1	Overview	2
2	EDMA Controller Programming	5
	2.1 EDMA Parameter RAM (PaRAM)	5
	2.2 EDMA Registers	6
3	Implementation Example	6
	3.1 Description	6
	3.2 Information Transfer DMA	7
	3.3 Synchronization DMA	8
	3.4 Data transfer DMA	9
4	References	9

List of Figures

Figure 1.	EDMA parameter and data buffers	2
Figure 2.	Externally Programmed EDMA Control and Data Flow	4

Trademarks are the property of their respective owners.

1 Overview

Because of the high transfer bandwidth requirements and asynchronous nature of RAKE receiver operations in a wireless basestation, the data transfer and associated interrupts between an external receive chip-rate processing device and a controlling DSP can prevent efficient DSP processing if not handled properly. This application report details a mechanism by which an external slave device can directly initiate DMA transfers with a TMS320C64x DSP without processor intervention.

The technique leverages several improvements to the EDMA controller of the TMS320C64x DSP to “effectively” allow an external slave device to program the internal EDMA controller, including;

- Memory-mapped, RAM-base register architecture of the EDMA controller
- DMA chaining

Unlike previous versions of the TMS320C6000™ family of DSPs, the EDMA controller of the TMS320C64x DSP utilizes a RAM-based EDMA transfer parameter table (PaRAM). The EDMA controller PaRAM is treated like all other internal memory of the DSP, and hence can be accessed by any standard memory transaction, including DMA transactions. This allows any device which can access the internal memory space of the DSP to program the EDMA controller. This technique can also be applied to the TMS320C621x and TMS320C671x DSPs which have a similar EDMA controller architecture.

DMA chaining refers to the ability of a single DMA transfer event to trigger multiple, separate DMA transfers. The order of the transfers is controlled by the chaining order of the EDMA PaRAM entries. This allows the DSP processor to setup a DMA transfer which is used solely to read an EDMA transfer parameter entry from an external device, and then to chain to that new EDMA entry after the EDMA PaRAM has been updated. Since DMA transfers can directly access the PaRAM of the EDMA controller, this allows an external slave device to “effectively” program the internal EDMA controller.

Figure 1 details where the different EDMA parameter/data buffers involved in this mechanism reside.

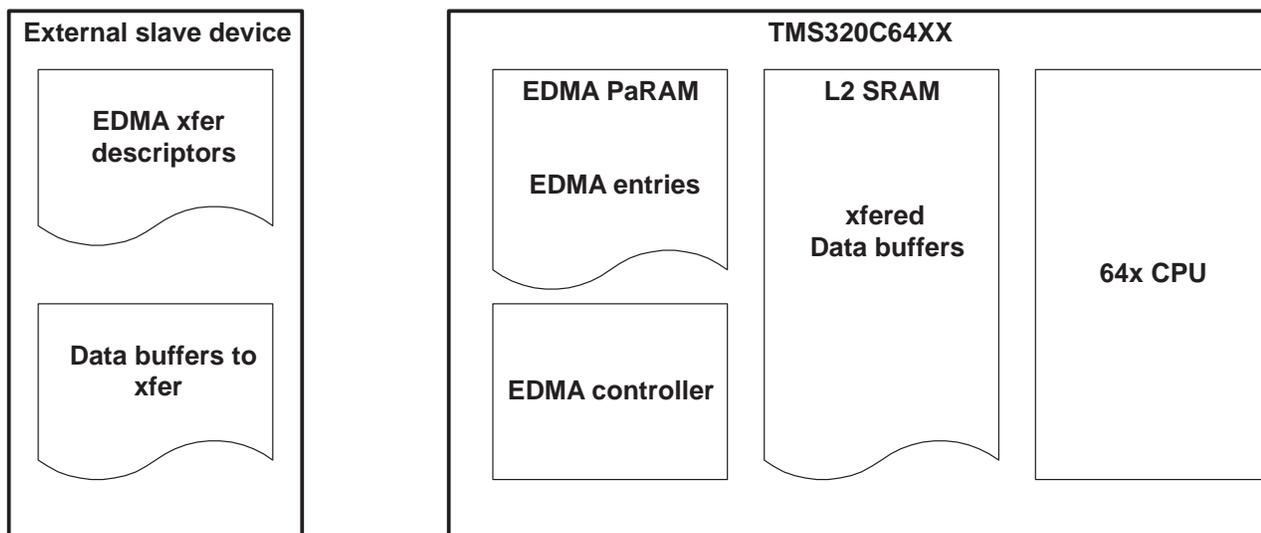


Figure 1. EDMA parameter and data buffers

The EDMA transfer descriptor contains the required information to configure an EDMA PaRAM entry for the transfer of data buffers from the external device to its location in the TMS320C64x L2 SRAM.

Because the external device is able to configure the EDMA PaRAM without processor intervention, the overhead for data transfers is significantly lowered.

The cause and effect cascade diagram below details the basic control and data flow used to implement this technique for an external device read transfer. The flow includes the following basic steps:

DSP sets up a static Information Transfer and Synchronization EDMA PaRAM entries. External device triggers the Information Transfer DMA.

- The **Information Transfer DMA** reads the Data Transfer EDMA PaRAM Entry from external device and writes it to PaRAM. The Information Transfer DMA chains (triggers) the Synchronization DMA.
- The **Synchronization DMA** sets event bit for the Data Transfer DMA in Event Set Register (ESR), which triggers the Data Transfer DMA.
- The **Data Transfer DMA** reads the data from the external device and writes it to L2 SRAM.

A similar flow could be used for external device write transfers and/or transfers to external memory buffers.

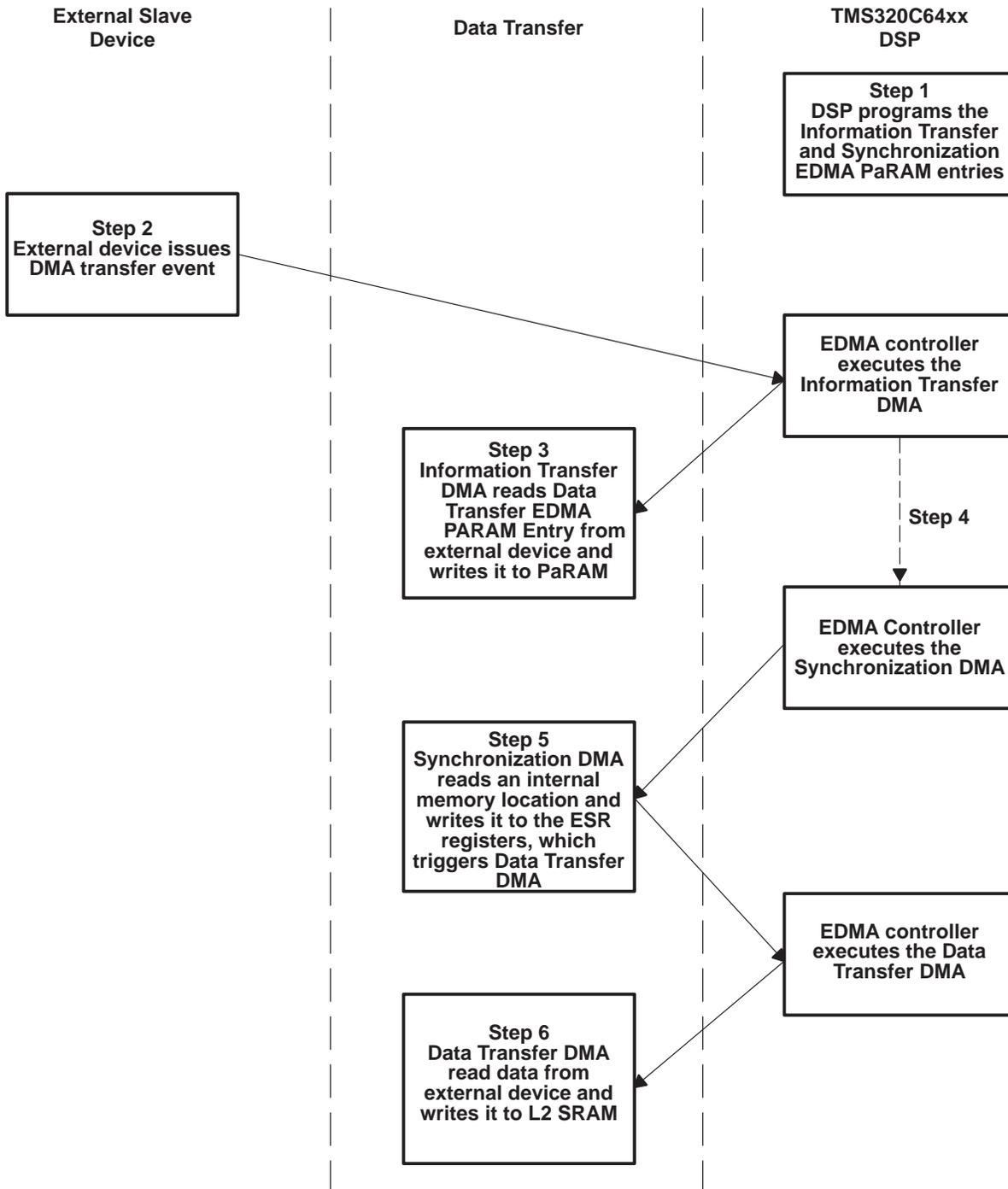


Figure 2. Externally Programmed EDMA Control and Data Flow

The **Synchronization DMA** is required to ensure that the EDMA controller PaRAM is completely updated with the Data Transfer EDMA PaRAM Entry before it is triggered. Using the event bit to manually trigger the Data Transfer DMA, instead of automatically trigger the DMA via chaining, eliminates possible race conditions caused by the internal pipeline buffers in the DSP.

If the **Data Transfer DMA** is used to transfer variable-length data, then the data must contain a data descriptor of some sort, or another chained DMA transfer could be used to transfer the data descriptor separately. Either way, the DSP processor and the external device must agree on the destination address prior to initiating the data transfer.

2 EDMA Controller Programming

2.1 EDMA Parameter RAM (PaRAM)

In order to use this technique of low overhead data transfers, the data transfer EDMA PaRAM entry must be programmed in a specific fashion. Depending on the processing capabilities and DSP interface of the external device, some programming aspects of the data transfer EDMA PaRAM entry can be simplified.

Each PaRAM entry consists of nine separate fields, including:

- Options (OPT)
- Source address (SRC)
- Array/frame count (FRMCNT)
- Element count (ELECNT)
- Destination address (DST)
- Array/frame index (FRMIDX)
- Element index (ELEIDX)
- Element count reload (ELERLD)
- Link address (LINK)

Depending on the complexity of the transfer, all or only a subset of these parameters needs to be dynamically calculated by the external device. Also, it is possible that the DSP processor could pre-calculate some of the dynamic parameters and download them to the external device prior to the data transfer event. This allows for reduced processing capabilities by the external device, while still supporting flexible transfer configurations.

At an absolute minimum, the source address, destination address and transfer count must be set dynamically in the data transfer EDMA PaRAM entry. If the external device includes distributed internal memory and/or registers, the source address can be dynamically set to gather the data from internal memory. If the external device implements a sequential access host interface, like a First In First Out (FIFO), then the source address can be fixed. Both the DSP and the external device must agree on the destination address prior to initiating the data transfer. This address could be static or dynamic, and could reference an internal or external memory buffer.

If the data transfer DMA is used to transfer variable-length data, then the element and/or frame count must also be set dynamically. Also, if a separate data descriptor transfer is used, then the data transfer EDMA PaRAM entry must set the appropriate chaining option fields. Finally, if the DSP processor needs to know that the data transfer is complete, the interrupt option fields can be set.

Finally, if this transfer mechanism is to be used repeatedly, both the Information Transfer and Synchronization EDMA PaRAM entries must include link addresses for reloading the original parameters.

2.2 EDMA Registers

To enable an external device to issue DMA requests for instance with the external interrupt 4, the Event Enable Registers (EER) must be set as follows:

```
EERH = 0x0000 0000
EERL = 0x0000 0010 (GPINT4/EXT_INT4 enabled)
```

To enable the appropriate DMA chaining for instance of channel 5, the Channel Chain Enable Registers (CCER) must be set as follows:

```
CCERH = 0x0000 0000
CCERL = 0x0000 0020 (PaRAM entry 5 enabled)
```

To issue DMA requests as the Synchronization DMA does for the Data Transfer DMA (Channel 6 for instance), the Event Set Registers (ESR) must be set as follows:

```
ESRH = 0x0000 0000
ESRL = 0x0000 0040 (Issues Channel 6)
```

It should be noted that only ESRL needs to be written by the Synchronization DMA (or ESRH if Channel 32 – 63 is used).

To enable the generation of a transfer complete interrupt to the DSP processor on channel 6 for instance, the Channel Interrupt Enable Register (CIER) must be set as follows:

```
CIERH = 0x0000 0000
CIERL = 0x0000 0040 (PaRAM entry 6 enabled)
```

3 Implementation Example

3.1 Description

The example EDMA controller PaRAM entries shown below implement the complete set of DMA transfers needed to repeatedly move a one-dimensional, variable-length block of data (containing a data descriptor header) from an external device to L2 memory (configured as SRAM, not cache). The example uses PaRAM entries 4, 5 and 6, plus several of the parameter reload PaRAM entries. The external device is assumed to have a FIFO-type host interface, which means that it must coordinate the transfer of parameters and data from its internal memory and/or registers to the FIFO. The external device issues its DMA requests via the EXT_INT4 signal. The DSP processor is interrupted at the end of each transfer. The dynamic fields of the Data Transfer EDMA PaRAM Entry that must be programmed by the external device are highlighted. The source address for the Synchronization DMA is in the to L2 SRAM. Unused locations at the bottom of PaRAM could also be used for the source address, with a slightly worse performance.

In the following section, the implementation examples for each DMA transfer based on the Chip Support Library (CSL) are provided. These examples can be used to build more complex schemes as required for each application and host-DSP interface.

3.2 Information Transfer DMA

```

#define INFORMATION_XFER_DMA EDMA_CHA_EXTINT4    // Channel 4

// EDMA_OPT_PRI_HIGH: Arbitrary, could be from 1 to 3
// EDMA_OPT_ESIZE_32BIT: 32 bit elements
// EDMA_OPT_2DS_NO: One-dimensional transfer
// EDMA_OPT_SUM_NONE: FIFO-type host interface with fixed address
// EDMA_OPT_2DD_NO: One-dimensional transfer
// EDMA_OPT_DUM_INC: Linear transfer from external device to L2 memory
// EDMA_OPT_TCINT_YES: Required for chaining
// EDMA_OPT_TCC_OF(SYNCHRONIZATION_XFER_DMA): PaRAM entry 5, required for chaining
// EDMA_OPT_TCCM_OF(SYNCHRONIZATION_XFER_DMA>>4): PaRAM entry 5, required for chain-
ing
// EDMA_OPT_ATCINT_NO: not used
// EDMA_OPT_ATCC_DEFAULT: not used
// EDMA_OPT_PDTS_DISABLE: not used
// EDMA_OPT_PDTD_DISABLE: not used
// EDMA_OPT_LINK_YES: Reload of parameters for repeated operation
// EDMA_OPT_FS_YES: Frame synchronized

#define EDMA_OPT_ENTRY_4          EDMA_OPT_RMK(EDMA_OPT_PRI_HIGH, \
                                             EDMA_OPT_ESIZE_32BIT, \
                                             EDMA_OPT_2DS_NO, \
                                             EDMA_OPT_SUM_NONE, \
                                             EDMA_OPT_2DD_NO, \
                                             EDMA_OPT_DUM_INC, \
                                             EDMA_OPT_TCINT_YES, \
                                             EDMA_OPT_TCC_OF(SYNCHRONIZATION_XFER_DMA), \
                                             EDMA_OPT_TCCM_OF(SYNCHRONIZATION_XFER_DMA>>4), \
                                             EDMA_OPT_ATCINT_NO, \
                                             EDMA_OPT_ATCC_DEFAULT, \
                                             EDMA_OPT_PDTS_DISABLE, \
                                             EDMA_OPT_PDTD_DISABLE, \
                                             EDMA_OPT_LINK_YES, \
                                             EDMA_OPT_FS_YES)

{
    EDMA_Config conf_ch;
// Step1: Configure information xfer
    conf_ch.opt = EDMA_OPT_ENTRY_4;           // Xfer options
    conf_ch.src = (Uint32)&xferDescriptor;    // Xfer src: descriptor header in host
    conf_ch.cnt = 4;                          // 4*32 bit words in PaRAM (OPT->DST)
    conf_ch.dst = EDMA_ADDRH(hEdma_6, OPT);   // Xfer dest.: DMA set of param. for chan.
6
    conf_ch.idx = 0;                          // index initialized to 0
    conf_ch.rld = 0;                          // reload initialized to 0
    EDMA_config(hEdma_4, &conf_ch);          // Load configuration in PaRAM
    EDMA_link(hEdma_4, hEdma_4_rld);         // Link to reload set of parameters
    EDMA_config(hEdma_4_rld, &conf_ch);     // Load config. in the reload set of param.
    EDMA_link(hEdma_4_rld, hEdma_4_rld);    // Link forever

    EDMA_enableChannel(hEdma_4);             // Enable INT4 event in EER
}
    
```

3.3 Synchronization DMA

```

#define SYNCHRONIZATION_XFER_DMA 5 // Arbitrary, could be any entry from 0 to 63

// EDMA_OPT_PRI_HIGH: MUST match the Information Transfer DMA priority
// EDMA_OPT_ESIZE_32BIT: 32 bit elements
// EDMA_OPT_2DS_NO: One-dimensional transfer
// EDMA_OPT_SUM_NONE: Fixed address
// EDMA_OPT_2DD_NO: One-dimensional transfer
// EDMA_OPT_DUM_NONE: Fixed address
// EDMA_OPT_TCINT_NO: No event generation
// EDMA_OPT_TCC_DEFAULT: not used
// EDMA_OPT_TCCM_DEFAULT: not used
// EDMA_OPT_ATCINT_NO: not used
// EDMA_OPT_ATCC_DEFAULT: not used
// EDMA_OPT_PDTS_DISABLE: not used
// EDMA_OPT_PDTD_DISABLE: not used
// EDMA_OPT_LINK_YES: Reload of parameters for repeated operation
// EDMA_OPT_FS_YES: Frame synchronized

#define EDMA_OPT_ENTRY_5          EDMA_OPT_RMK(EDMA_OPT_PRI_HIGH, \
                                              EDMA_OPT_ESIZE_32BIT, \
                                              EDMA_OPT_2DS_NO, \
                                              EDMA_OPT_SUM_NONE, \
                                              EDMA_OPT_2DD_NO, \
                                              EDMA_OPT_DUM_NONE, \
                                              EDMA_OPT_TCINT_NO, \
                                              EDMA_OPT_TCC_DEFAULT, \
                                              EDMA_OPT_TCCM_DEFAULT, \
                                              EDMA_OPT_ATCINT_NO, \
                                              EDMA_OPT_ATCC_DEFAULT, \
                                              EDMA_OPT_PDTS_DISABLE, \
                                              EDMA_OPT_PDTD_DISABLE, \
                                              EDMA_OPT_LINK_YES, \
                                              EDMA_OPT_FS_YES)

{
    EDMA_Config conf_ch;
// Step1: Configure synchronization DMA
    ESRegLowImage = (1<<(DATA_XFER_DMA)); // Configure ESR image to be xfered

    conf_ch.opt = EDMA_OPT_ENTRY_5; // Xfer options
    conf_ch.src = (Uint32)&ESRegLowImage; // Xfer source: ESR image
    conf_ch.cnt = 1; // 1*32 bit word
    conf_ch.dst = (Uint32)0x01A0FFFC; // Xfer destination: Event Set Low register
    EDMA_config(hEdma_5, &conf_ch); // Load configuration in PaRAM
    EDMA_link(hEdma_5, hEdma_5_rld); // Link to reload set of parameters
    EDMA_config(hEdma_5_rld, &conf_ch); // Load config. in the reload set of parame-
ters
    EDMA_link(hEdma_5_rld, hEdma_5_rld); // Link forever

    EDMA_enableChaining(hEdma_5); // Enable chaining: CCER (CCERL/CCERH) set-
ting
}

```

3.4 Data transfer DMA

```

#define DATA_XFER_DMA 6 // Arbitrary, could be any entry from 0 to 63

// EDMA_OPT_PRI_LOW: Arbitrary
// EDMA_OPT_ESIZE_32BIT: Arbitrary, could be from 0 to 2
// EDMA_OPT_2DS_NO: One-dimensional transfer
// EDMA_OPT_SUM_INC: FIFO-type host interface with fixed address - INC FOR SIM TEST
// EDMA_OPT_2DD_NO: One-dimensional transfer
// EDMA_OPT_DUM_INC: Linear transfer from external device to L2 memory
// EDMA_OPT_TCINT_YES: Interrupt DSP processor
// EDMA_OPT_TCC_OF(DATA_XFER_DMA): Interrupt DSP processor
// EDMA_OPT_TCCM_OF(DATA_XFER_DMA>>4): Interrupt DSP processor
// EDMA_OPT_ATCINT_NO: not used
// EDMA_OPT_ATCC_DEFAULT: not used
// EDMA_OPT_PDTS_DISABLE: not used
// EDMA_OPT_PDTD_DISABLE: not used
// EDMA_OPT_LINK_YES: // Link to NULL Entry for termination
// EDMA_OPT_FS_YES: Frame synchronized

#define EDMA_OPT_ENTRY_6          EDMA_OPT_RMK(EDMA_OPT_PRI_LOW, \
                                                EDMA_OPT_ESIZE_32BIT, \
                                                EDMA_OPT_2DS_NO, \
                                                EDMA_OPT_SUM_INC, \
                                                EDMA_OPT_2DD_NO, \
                                                EDMA_OPT_DUM_INC, \
                                                EDMA_OPT_TCINT_YES, \
                                                EDMA_OPT_TCC_OF(DATA_XFER_DMA), \
                                                EDMA_OPT_TCCM_OF(DATA_XFER_DMA>>4), \
                                                EDMA_OPT_ATCINT_NO, \
                                                EDMA_OPT_ATCC_DEFAULT, \
                                                EDMA_OPT_PDTS_DISABLE, \
                                                EDMA_OPT_PDTD_DISABLE, \
                                                EDMA_OPT_LINK_YES, \
                                                EDMA_OPT_FS_YES)

{
    EDMA_Config conf_ch;
    // Step1: Configure data transfer DMA and NULL transfer
    conf_ch.opt = 0;
    conf_ch.src = 0;
    conf_ch.cnt = 0;
    conf_ch.dst = 0;
    conf_ch.idx = 0;
    conf_ch.rld = 0;
    EDMA_config(hEdma_6, &conf_ch); // Load configuration in PaRAM
    EDMA_link(hEdma_6, hEdma_NULL); // Link to NULL set of parameters
    EDMA_config(hEdma_NULL, &conf_ch); // Load configuration in PaRAM

    EDMA_intEnable(DATA_XFER_DMA); // Enable the DATA_XFER_DMA to gen. a CPU interrupt
}
    
```

4 References

1. *TMS320C6000 DSP Peripherals Overview Reference Guide (SPRU190)*
2. *TMS320C6000 Chip Support Library API Reference Guide (SPRU401)*

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 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 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.

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

Copyright © 2004, Texas Instruments Incorporated