TMS320C3x Peripheral Control Library User's Guide

SPRU086
September 1992

IMPORTANT NOTICE

Texas Instruments Incorporated (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to current specifications in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Please be aware that TI products are not intended for use in life-support appliances, devices, or systems. Use of TI product in such applications requires the written approval of the appropriate TI officer. Certain applications using semiconductor devices may involve potential risks of personal injury, property damage, or loss of life. In order to minimize these risks, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards. Inclusion of TI products in such applications is understood to be fully at the risk of the customer using TI devices or systems.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

Read This First

About This Manual

This manual describes the TMS320C3x peripheral control library—a collection of data structures and macros for controlling the ’C3x bus control peripherals, DMA, serial ports, and timers via the C programming language.

How to Use This Manual

This document contains the following chapters:

Chapter 1: ***Overview.*** Describes the features of the library files.

Chapter 2: ***TMS320C3x Peripheral Set.*** Describes the header files, macros, and pointers for the ’C3x peripherals.

Chapter 3: ***TMS320C3x Peripheral Control Example.*** Illustrates how the ’C3x Peripheral Control Library can be used to program a ’C3x serial port.

Appendix A: ***Tables of Peripheral Registers, Structure-Member Names, and Bit-Field Names***. Provides examples and tables listing the names used to access each of the peripheral registers and bit fields through C-peripheral pointers.

Appendix B: ***Header Files***. Includes bus30.h, dma30.h, serprt30.h, and timer30.h header file listings.

Style and Symbol Conventions

This document uses the following conventions.

- Program listings, program examples, interactive displays, filenames, and symbol names are shown in a `special typeface` similar to a

typewriter's. Examples use a **bold version** of the special typeface for emphasis; interactive displays use a **bold version** of the special typeface to distinguish commands that you enter from items that the system displays (such as prompts, command output, error messages, etc.).

Here is a sample program listing:

```
0011  0005  0001          .field    1, 2
0012  0005  0003          .field    3, 4
0013  0005  0006          .field    6, 3
0014  0006                .even
```

Here is an example of a system prompt and a command that you might enter:

```
C:  csr –a /user/ti/simuboard/utilities
```

- In syntax descriptions, the instruction, command, or directive is in a **bold typeface** font and parameters are in an *italic typeface*. Portions of a syntax that are in **bold** should be entered as shown; portions of a syntax that are in *italics* describe the type of information that should be entered. Here is an example of a directive syntax:

  .asect "*section name***",** *address*

  .asect is the directive. This directive has two parameters, indicated by *section name* and *address*. When you use .asect, the first parameter must be an actual section name, enclosed in double quotes; the second parameter must be an address.

- Square brackets (**[** and **]**) identify an optional parameter. If you use an optional parameter, you specify the information within the brackets; you don't enter the brackets themselves. Here's an example of an instruction that has an optional parameter:

  LALK *16–bit constant [, shift]*

  The LALK instruction has two parameters. The first parameter, *16-bit constant*, is required. The second parameter, *shift*, is optional. As this syntax shows, if you use the optional second parameter, you must precede it with a comma.

  Square brackets are also used as part of the pathname specification for VMS pathnames; in this case, the brackets are actually part of the pathname (they are not optional).

- Braces ({ and }) indicate a list. The symbol **|** (read as *or*) separates items within the list. Here's an example of a list:

  ```
  { * | *+ | *– }
  ```

This provides three choices: *, *+, or *–.

Unless the list is enclosed in square brackets, you must choose one item from the list.

- Some directives can have a varying number of parameters. For example, the .byte directive can have up to 100 parameters. The syntax for this directive is:

  .byte $value_1$ *[, ... ,* $value_n$*]*

  This syntax shows that .byte must have at least one value parameter, but you have the option of supplying additional value parameters, separated by commas.

Related Documentation From Texas Instruments

TMS320 Floating-Point DSP Assembly Language Tools User's Guide (literature number SPRU035) 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 'C3x and 'C4x generations of devices.

TMS320 Floating-Point DSP Optimizing C Compiler User's Guide (literature number SPRU024) describes the TMS320 floating-point C compiler. This C compiler accepts ANSI standard C source code and produces TMS320 assembly language source code for the 'C3x and 'C4x generations of devices.

TMS320C3x User's Guide (literature number SPRU031) describes the 'C3x 32-bit floating-point microprocessor, developed for digital signal processing as well as general applications. Covered are its architecture, internal register structure, instruction set, pipeline, specifications, and operation of its DMA and its two serial ports. Software and hardware applications are included.

TMS320C40 Parallel Runtime Support Library User's Guide (literature number SPRU084) describes the 'C40 Parallel Runtime Support Library, a standard method of programming the 'C40 peripherals at the register and bit level via the C programming language. Library and header files, a summary of parallel runtime support functions and macros, a functions reference, and a listing of header files are included.

If You Need Assistance. . .

If you want to. . .	Do this. . .
Request more information about Texas Instruments Digital Signal Processing (DSP) products	Call the CRC†: **(214) 995–6611** Or write to: Texas Instruments Incorporated Market Communications Manager, MS 736 P.O. Box 1443 Houston, Texas 77251–1443
Order Texas Instruments documentation	Call the CRC†: **(214) 995–6611**
Ask questions about product operation or report suspected problems	Call the DSP hotline: **(713) 274–2320**
Report mistakes in this document or any other TI documentation	Fill out and return the reader response card at the end of this book, or send your comments to: Texas Instruments Incorporated Technical Publications Manager, MS 702 P.O. Box 1443 Houston, Texas 77251–1443

† Texas Instruments Customer Response Center

Contents

Tables

Examples

Chapter 1

Overview

This user's guide describes the TMS320C3x Peripheral Control Library, a standard for controlling 'C3x peripherals using the C programming language. The library, part of the TMS320 Floating-Point DSP Optimizing C Compiler, offers consistency in naming and accessing variables, thereby reducing development time for new code and facilitating code maintenance.

Because all the 'C3x peripherals and their corresponding registers are memory-mapped, a device's peripheral can be controlled easily from C code. The flexibility of the C language allows structures to be defined for modifying 'C3x peripheral memory-mapped registers. Furthermore, because many of the peripheral control registers use individual bit locations to define functionality, bit fields are implemented in this standard with C bit-field structures.

By utilizing this standard for controlling 'C3x peripherals, you can develop custom support libraries that are both portable and readable.

This document describes the TMS320C3x peripheral registers and their structure member names and gives examples on how to utilize the library.

Chapter 2

TMS320C3x Peripheral Set

The TMS320C3x peripherals include external buses, direct memory access (DMA), two serial ports, and two timers. The on-chip integration of all of these peripherals increases processor performance and overall system hardware integration. Each application has different peripheral requirements and configurations that must be programmed before you use the peripheral. The peripherals are configured by writing values to the corresponding control register. For example, the following nondescriptive C statement starts timer 0 by writing a value of 0x2C1 into the timer 0 global control register at address 0x808020:

```
*(unsigned int *)0x808020 = 0x2C1;
```

The 'C3x Peripheral Control Library simplifies writing more readable C code and helps make the DSP code consistent from one application to the next.

This chapter discusses these topics:

Topic **Page**

2.1 Header Files

The ’C3x Peripheral Control Library consists of four header files:

- bus30.h
- dma30.h
- serprt30.h
- timer30.h

The header files declare structures and macros. In particular, each header file declares:

- Data structures for the peripheral’s memory mapping
- Data structures for the bit fields of the peripheral’s control registers
- Macros for declaring pointers to the peripherals
- Macros for bit field assignment

Before using a peripheral macro or data structure, you must include the appropriate header file in the program, as shown below.

```
#include <timer30.h>
    .
    .
    .
TIMER_ADDR(0)–>gcontrol=0x0;
```

2.2 Macros

Four macros facilitate the assignment of C pointers to the four peripherals:

- BUS_ADDR
- DMA_ADDR
- SERIAL_PORT_ADDR(n)
- TIMER_ADDR(n)

These macros assign C symbols to the peripheral base addresses and typecast the memory location to point to the peripheral data structure. For example, BUS_ADDR is defined as follows:

```
#define BUS_ADDR     ((volatile BUS_REG *)((char *) BUS_BASE))
```

where BUS_REG is the data structure describing the bus control register memory mapping (discussed in the next section) and BUS_BASE is 0x808060.

2.3 Pointers

Pointers to peripherals are then used to access the peripheral control registers. Peripheral pointers **point** to data structures that represent the mapping of the peripheral control registers. The four data structures are:

- BUS_REG
- DMA_REG
- SERIAL_PORT_REG
- TIMER_REG

For example, the ’C3x memory map of the timer control registers described in Figure 2–1 is defined in C with the TIMER_REG data structure that follows the figure:

Figure 2–1. Memory-Mapped Timer Locations

Register	Peripheral Address	
	Timer 0	Timer 1
Timer Global Control	808020h	808030h
Reserved	808021h	808031h
Reserved	808022h	808032h
Reserved	808023h	808033h
Timer Counter	808024h	808034h
Reserved	808025h	808035h
Reserved	808026h	808036h
Reserved	808027h	808037h
Timer Period	808028h	808038h
Reserved	808029h	808039h
Reserved	80802Ah	80803Ah
Reserved	80802Bh	80803Bh
Reserved	80802Ch	80803Ch
Reserved	80802Dh	80803Dh
Reserved	80802Eh	80803Eh
Reserved	80802Fh	80803Fh

```
typedef   struct
{
      TIMER_CONTROL    _gtrl;       /* Timer global control */
      unsigned int     reserved1[3];/* 3 reserved locations */
      unsigned int     counter;     /* Timer counter */
      unsigned int     reserved2[3] /* 3 more reserved locations */
      unsigned int     period;      /* Time period */
} TIMER_REG;
```

Because the peripheral control registers can change independently of the CPU, peripheral pointers should be declared by using the volatile-data type qualifier. Example 2–1 shows how to use the TIMER_REG data structure and the TIMER_ADDR(n) macro to assign *timer0_ptr to point to timer 0.

Example 2–1. Declaring a Peripheral Pointer

```
volatile TIMER_REG *timer0_ptr = TIMER_ADDR(0);
```

The peripheral control registers can be set one bit at a time or by assigning an integer value to the register. Example 2–2 shows how to use both integer and bit-field methods for halting timer 0 by setting the $\overline{\text{HLD}}$ bit of the timer global control register to zero. This example uses the timer pointer from Example 2–1. To aid in these operations, descriptive macros for both bit-field and integer manipulations are defined in the header files.

Example 2–2. Two Methods of Setting Control Registers

Integer Method: timer0_ptr–>gcontrol &= ~0x80

Bit-Field Assignment Method: timer0_ptr–>gcontrol_bit.hld_ = 0

Chapter 3

TMS320C3x Peripheral Control Example

This chapter illustrates how you can use the ’C3x Peripheral Control Library to program the ’C3x serial port 0 via several combined methods of setting up the control registers.

Consider the following configuration of serial port 0:

- ❑ Generate the bit clock at 8M bits per second (assuming a 32-MHz input clock)
- ❑ Generate frame sync pulses
- ❑ Transmit/receive 32-bit data
- ❑ Operate in standard/fixed mode
- ❑ Generate interrupts on data transmitted

To obtain this configuration, you must set the serial port registers accordingly:

Register	Address	Value
Global Control	0x808040	0xCBC00C4
FSX/DX/CLKX Port Control	0x808042	0x111
FSR/DR/CLKR Port Control	0x808043	0x111
Rec/Trans Timer Control	0x808044	0x3CF

Example 3–1 illustrates how this configuration is achieved with the ’C3x Peripheral Control Library. The control segment is highly readable and produces efficient assembly code when compiled.

Example 3–1. Configuring a TMS320C3x Serial Port

```
#include <serprt30.h>
          .
          .
          .
    /* define a pointer to serial port 0 */
    volatile SERIAL_PORT_REG *sp0 = SERIAL_PORT_ADDR(0);


    /* set  serial port 0's fsr/dr/clkr port control register by integer assignment */
    SERIAL_PORT_ADDR(0)->s_r_control = 0x111;

    /*  set   serial  port  0's  fsx/dx/clkx  port  control register by integer
        assignment using descriptive macros*/
    sp0->s_x_control = CLKXFUNC | DXFUNC | FSXFUNC;

    /* set serial port 0's global control register by bit field assignment*/
    SERIAL_PORT_ADDR(0)->global_bit.fsxout = 1;
    sp0->global_bit.xclksrce = 1;
    sp0->global_bit.rclksrce = 1;
    sp0->global_bit.xlen = XLEN_32;
    sp0->global_bit.rlen = RLEN_32;
    sp0->global_bit.xint = 1;
    sp0->global_bit.xreset = 1;
    sp0->global_bit.rreset = 1;

    /*  set   serial  port  0's  rec/transm  timer  control register */
    sp0->s_rxt_control = XGO | HLD_ | XCP_  |
    XCLKSRC | RGO | RHLD_ | RCP_ | RCLKSRC;
          .
          .
          .
```

Appendix A

Tables of Peripheral Registers, Structure-Member Names, and Bit-Field Names

The TMS320C3x Peripheral Control Library provides C data structures for manipulating the TMS320C3x peripherals. The following tables list the data structure member names used to access each of the peripheral registers and bit fields through C peripheral pointers. Refer to the *TMS320C3x User's Guide* for a detailed explanation of the registers and bit-field descriptions. The first entry for each register shows how to access that register as an integer. The remaining entries show how to access the register's bit fields individually. Each table is followed by an example.

This appendix includes these tables:

Topic **Page**

Table A–1. Bus-Control Registers

'C3x Register	Assignment	Bit Field	Member Name
Primary Bus Control	Integer assignment	——	–>prim_gcontrol
	Bit-field assignment	BNKCMP	–>prim_gcontrol_bit.bnkcmp
		HIZ	–>prim_gcontrol_bit.hiz
		HOLDST	–>prim_gcontrol_bit.holdst
		NOHOLD	–>prim_gcontrol_bit.nohold
		SWW	–>prim_gcontrol_bit.sww
		WTCNT	–>prim_gcontrol_bit.wtcnt
Expansion Bus Control	Integer assignment	——	–>exp_gcontrol
	Bit-field assignment	SWW	–>exp_gcontrol_bit.sww
		WTCNT	–>exp_gcontrol_bit.wtcnt

Example A–1. Bus Control

```
#include<bus30.h>
volatile BUS_REG *bus_ptr=BUS_ADDR;     /* define pointer to bus peripheral */
bus_ptr–>exp_gcontrol=0;      /* zero wait states on expansion bus */
```

Table A–2. DMA-Control Registers

'C3x Register	Assignment	Bit Field	Member Name
DMA Global Control	Integer assignment	——	–>gcontrol
	Bit-field assignment	DECSRC	–>gcontrol_bit.decsrc
		DECDST	–>gcontrol_bit.decdst
		INCDST	–>gcontrol_bit.incdst
		INCSRC	–>gcontrol_bit.incsrc
		START	–>gcontrol_bit.start
		STAT	–>gcontrol_bit.stat
		SYNC	–>gcontrol_bit.sync
		TC	–>gcontrol_bit.tc
		TCINT	–>gcontrol_bit.tcint
DMA Source Address	Integer assignment	——	–>source
DMA Destination Address	Integer assignment	——	–>destination
DMA Transfer Counter	Integer assignment	——	–>transfer_counter

Example A–2. DMA-Control Registers

```
#include <dma30.h>
volatile DMA_REG *dma=DMA_ADDR;         /* define pointer to dma */
dma–>gcontrol_bit.start=STOP; /* stop the dma by setting the start bits*/
```

Table A–3. Serial-Port-Control Registers

’C3x Register	Assignment	Bit Field	Member Name
Serial Port Global Control	Integer assignment	——	–>gcontrol
	Bit-field assignment	CLKRP	–>gcontrol_bit.clkrp
		CLKXP	–>gcontrol_bit.clkxp
		DRP	–>gcontrol_bit.drp
		DXP	–>gcontrol_bit.dxp
		FSRP	–>gcontrol_bit.fsrp
		FSXP	–>gcontrol_bit.fsxp
		FSXOUT	–>gcontrol_bit.fsxout
		HS	–>gcontrol_bit.hs
		RCLKSRCE	–>gcontrol_bit.rclksrce
		RFSM	–>gcontrol_bit.rfsm
		RINT	–>gcontrol_bit.rint
		RLEN	–>gcontrol_bit.rlen
		RRDY	–>gcontrol_bit.rrdy
		RRESET	–>gcontrol_bit.rreset
		RSRFULL	–>gcontrol_bit.rsrfull
		RTINT	–>gcontrol_bit.rtint
		RVAREN	–>gcontrol_bit.rvaren
		XCLKSRCE	–>gcontrol_bit.xclksrce
		XFSM	–>gcontrol_bit.xfsm
		XINT	–>gcontrol_bit.xint
		XLEN	–>gcontrol_bit.xlen
		XRDY	–>gcontrol_bit.xrdy
		XRESET	–>gcontrol_bit.xreset
		XSREMPTY	–>gcontrol_bit.xsrempty
		XTINT	–>gcontrol_bit.xtint
		XVAREN	–>gcontrol_bit.xvaren

Table A–3. Serial-Port-Control Registers (Continued)

’C3x Register	Assignment	Bit Field	Member Name
FSR/DR/CLKR Port Control	Integer assignment	——	–>s_r_control
	Bit-field assignment	CLKRDATIN	–>s_r_control_bit.clkdati
		CLKRDATOUT	–>s_r_control_bit.clkdato
		CLKRFUNC	–>s_r_control_bit.clkfunc
		CLKRI/O	–>s_r_control_bit.clki_o
		DRDATIN	–>s_r_control_bit.ddatin
		DRDATOUT	–>s_r_control_bit.ddatout
		DRFUNC	–>s_r_control_bit.dfunc
		DRI/O	–>s_r_control_bit.di_o
		DRP	–>gcontrol_bit.drp
		FSRDATIN	–>s_r_control_bit.fsdatin
		FSRDATOUT	–>s_r_control_bit.fsdatout
		FSRFUNC	–>s_r_control_bit.fsfunc
		FSRI/O	–>s_r_control_bit.fsi_o
FSX/DX/CLKX Port Control	Integer assignment	——	–>s_x_control
	Bit-field assignment	CLKXDATIN	–>s_x_control_bit.clkdati
		CLKXDATOUT	–>s_x_control_bit.clkdato
		CLKXFUNC	–>s_x_control_bit.clkfunc
		CLKXI/O	–>s_x_control_bit.clki_o
		DXDATIN	–>s_x_control_bit.ddatin
		DXDATOUT	–>s_x_control_bit.ddatout
		DXFUNC	–>s_x_control_bit.dfunc
		DXI/O	–>s_x_control_bit.di_o
		FSXDATIN	–>s_x_control_bit.fsdatin
		FSXDATOUT	–>s_x_control_bit.fsdatout
		FSXFUNC	–>s_x_control_bit.fsfunc
		FSXI/O	–>s_x_control_bit.fsi_o

Table A–3. Serial-Port-Control Registers (Concluded)

’C3x Register	Assignment	Bit Field	Member Name
Receive/Transmit Timer Control	Integer assignment	——	–>s_rxt_control
	Bit-field assignment	RCLKSRC	–>s_rxt_control_bit.rclksrc
		RC/$\overline{P}$	–>s_rxt_control_bit.rcp_
		RGO	–>s_rxt_control_bit.rgo
		$\overline{RHLD}$	–>s_rxt_control_bit.rhld_
		RTSTAT	–>s_rxt_control_bit.rtstat
		XCLKSRC	–>s_rxt_control_bit.xclksrc
		XC/$\overline{P}$	–>s_rxt_control_bit.xcp_
		XGO	–>s_rxt_control_bit.xgo
		$\overline{XHLD}$	–>s_rxt_control_bit.xhld_
		XTSTAT	–>s_rxt_control_bit.xtstat
Receive/Transmit Timer Counter	Integer assignment	——	–>s_rxt_counter
	Bit-field assignment	Receive Counter	–>s_rxt_counter_bit.r_counter
		Transmit Counter	–>s_rxt_counter_bit.x_counter
Receive/Transmit Timer Period	Integer assignment	——	–>s_rxt_period
	Bit-field assignment	Receive Period	–>s_rxt_period_bit.r_period
		Transmit Period	–>s_rxt_period_bit.x_period
Data Receive	Integer assignment	——	r_data
Data Transmit		——	x_data

Example A–3. Serial Port Control

```
#include <serprt30.h>
volatile SERIAL_PORT_REG *sp1=SERIAL_PORT_ADDR(1);  /* define pointer to serial port 1 */
sp1–> x_data= uncompress(x);                /* write out result */
```

Table A–4. Timer-Control Registers

’C3x Register	Assignment	Bit Field	Member Name
Timer Global Control	Integer assignment	——	–>gcontrol
	Bit-field assignment	CLKSRC	–>gcontrol_bit.clksrc
		C/$\overline{P}$	–>gcontrol_bit.cp_
		DATIN	–>gcontrol_bit.datin
		DATOUT	–>gcontrol_bit.datout
		FUNC	–>gcontrol_bit.func
		GO	–>gcontrol_bit.go
		$\overline{HLD}$	–>gcontrol_bit.hld_
		INV	–>gcontrol_bit.inv
		$\overline{I/O}$	–>gcontrol_bit.i_o
		TSTAT	–>gcontrol_bit.tstat
Timer Counter	Integer assignment	——	–>counter
Timer Period		——	–>period

Example A–4. Timer Control

```
#include<timer30.h>
volatile TIMER_ADDR(0)–>gcontrol=0;     /* stop timer */
                                        /* determine remaining time */
int time_remain=TIMER_ADDR(0)–>period–TIMER_ADDR(0)–>counter
```

Appendix B

Header Files

This appendix lists the bus30.h, dma30.h, serprt30.h, and timer30.h header files:

Topic **Page**

B.1 bus30.h Header File

```
/*************************************************************************/
/* bus30.h v4.5                                                          */
/* Copyright (c) 1991 Texas Instruments Incorporated                     */
/*************************************************************************/
#ifndef _BUS30
#define _BUS30

#define exp_gcontrol      _egctrl._intval
#define exp_gcontrol_bit  _egctrl._bitval
#define prim_gcontrol     _pgctrl._intval
#define prim_gcontrol_bit _pgctrl._bitval

/*************************************************************************/
/* MACRO DEFINITION FOR BUS BASE ADDRESS                                 */
/*************************************************************************/
#define BUS_BASE        0x808060
#define BUS_ADDR        ((volatile BUS_REG *) ((char *) BUS_BASE))

/*************************************************************************/
/* STRUCTURE DEFINITION FOR PRIM AND EXP BUS GLOBAL CONTROL REGISTERS    */
/*************************************************************************/
typedef union
{
    unsigned int _intval;
    struct
    {
         unsigned int holdst   :1;                  /* Hold status bit */
         unsigned int nohold   :1;                 /* Port hold signal */
         unsigned int hiz      :1;                    /* Internal hold */
         unsigned int sww      :2;                    /* S/W wait mode */
         unsigned int wtcnt    :3;                  /* S/W wait count */
         unsigned int bnkcmp   :5;                    /* Bank compare */
         unsigned int r_rest   :19;                       /* reserved */
    } _bitval;
} PRIMARY_BUS_CONTROL;

typedef union
{
    unsigned int _intval;
    struct
    {
         unsigned int r_012    :3;                        /* reserved */
         unsigned int sww      :2;                   /* S/W wait mode */
         unsigned int wtcnt    :3;                  /* S/W wait count */
         unsigned int r_rest   :24;                       /* reserved */
    } _bitval;
} EXPANSION_BUS_CONTROL;
```

```
/*************************************************************************/
/* STRUCTURE DEFINITION FOR BUS REGISTERS                                */
/*************************************************************************/
typedef struct
{
    EXPANSION_BUS_CONTROL _egctrl;     /* Exapansion bus control register */
    unsigned int          reserved1[3];                    /* reserved */
    PRIMARY_BUS_CONTROL   _pgctrl;        /* Primary bus control register */
} BUS_REG;

/*************************************************************************/
/* BUS MACROS FOR BIT FIELD DEFINITIONS                                  */
/*************************************************************************/
/* Bus macros for global control registers integer assignments           */
/* e.g. BUS_ADDR->prim_gcontrol = EXTERNAL_RDY | WS_1;   sets the        */
/* corresponding bits of the primary bus control register.               */
/*************************************************************************/
#define HOLDST          0x1
#define NOHOLD          0x2
#define HIZ             0x4
#define EXTERNAL_RDY    0x00
#define INTERNAL_RDY    0x08
#define OR_EXT_INT      0x10
#define AND_EXT_INT     0x18
#define WS_0            0x00
#define WS_1            0x20
#define WS_2            0x40
#define WS_3            0x60
#define WS_4            0x80
#define WS_5            0xA0
#define WS_6            0xC0
#define WS_7            0xE0
#define BANK_16M        0x00
#define BANK_8M         0x100
#define BANK_4M         0x200
#define BANK_2M         0x300
#define BANK_1M         0x400
#define BANK_512K       0x500
#define BANK_256K       0x600
#define BANK_128K       0x700
#define BANK_64K        0x800
#define BANK_32K        0x900
#define BANK_16K        0xA00
#define BANK_8K         0xB00
#define BANK_4K         0xC00
#define BANK_2K         0xD00
#define BANK_1K         0xE00
#define BANK_512        0xF00
#define BANK_256        0x1000
```

```
/*************************************************************************/
/* Bus macros for global control registers bit field assignments          */
/* e.g. BUS_ADDR->prim_gcontrol_bit.sww = AND_RDY;                        */
/*      BUS_ADDR->exp_gcontrol_bit.wtcnt = WAIT_2;                        */
/* sets the appropriate bits of bus control registers.                    */
/*************************************************************************/
#define HOLD            1
#define INT_RDY         0
#define EXT_RDY         1
#define OR_RDY          2
#define AND_RDY         3
#define BANK_SIZE_16M   0
#define BANK_SIZE_8M    1
#define BANK_SIZE_4M    2
#define BANK_SIZE_2M    3
#define BANK_SIZE_1M    4
#define BANK_SIZE_512K  5
#define BANK_SIZE_256K  6
#define BANK_SIZE_128K  7
#define BANK_SIZE_64K   8
#define BANK_SIZE_32K   9
#define BANK_SIZE_16K   10
#define BANK_SIZE_8K    11
#define BANK_SIZE_4K    12
#define BANK_SIZE_2K    13
#define BANK_SIZE_1K    14
#define BANK_SIZE_512   15
#define BANK_SIZE_256   16
#define WAIT_0          0
#define WAIT_1          1
#define WAIT_2          2
#define WAIT_3          3
#define WAIT_4          4
#define WAIT_5          5
#define WAIT_6          6
#define WAIT_7          7

#endif /* #ifndef _BUS30 */
```

B.2 dma30.h Header Files

```
/*****************************************************************************/
/* dma30.h v4.5                                                              */
/* Copyright (c) 1991 Texas Instruments Incorporated                         */
/*****************************************************************************/
#ifndef _DMA30
#define _DMA30

#ifndef gcontrol
#define gcontrol     _gctrl._intval
#endif

#ifndef gcontrol_bit
#define gcontrol_bit _gctrl._bitval
#endif

/*****************************************************************************/
/* MACRO DEFINITIONS FOR DMA BASE ADDRESS                                    */
/*****************************************************************************/
#define DMA_BASE        0x808000
#define DMA_SIZE        16
#define DMA_ADDR        ((volatile DMA_REG *) ((char *) DMA_BASE))

/*****************************************************************************/
/* STRUCTURE DEFINITION FOR DMA GLOBAL CONTROL REGISTER                      */
/*****************************************************************************/
typedef union
{
    unsigned int _intval;
    struct
    {
         unsigned int start    :2;                 /* Start/Stop control */
         unsigned int stat     :2;                       /* DMA status */
         unsigned int incsrc   :1;        /* Increment source addr ON/OFF */
         unsigned int decsrc   :1;        /* Decrement source addr ON/OFF */
         unsigned int incdst   :1;          /* Increment dest addr ON/OFF */
         unsigned int decdst   :1;          /* Increment dest addr ON/OFF */
         unsigned int sync     :2;         /* Source/dest synchronization */
         unsigned int tc       :1;          /* Transfer terminate control */
         unsigned int tcint    :1;        /* DMA interrupt to CPU control */
         unsigned int r_rest   :20;                          /* reserved */
    } _bitval;
} DMA_CONTROL;

/*****************************************************************************/
/* STRUCTURE DEFINITION FOR DMA REGISTERS                                    */
/*****************************************************************************/
typedef struct
{
    DMA_CONTROL       _gctrl;                 /* Global control register */
    unsigned int      reserved1[3];                       /* reserved */
    unsigned int      source;                 /* Source address register */
    unsigned int      reserved2[1];                       /* reserved */
    unsigned int      destination;        /* Destination address register */
    unsigned int      reserved3[1];                       /* reserved */
    unsigned int      transfer_counter;      /* Transfer counter register */
} DMA_REG;
```

```
/*************************************************************************/
/* DMA MACROS FOR BIT FIELD DEFINITIONS                                  */
/*************************************************************************/
/* DMA macros for global control register integer assignments            */
/* e.g. DMA_ADDR->gcontrol = START0 | SYNC1 | TCINT sets the corresponding*/
/* bits of the DMA global control register                               */
/*************************************************************************/
#define START0     0x0
#define START1     0x1
#define START2     0x2
#define START3     0x3
#define STAT0      0x0
#define STAT1      0x4
#define STAT2      0x8
#define STAT3      0xC
#define INCSRC     0x10
#define DECSRC     0x20
#define INCDST     0x40
#define DECDST     0x80
#define SYNC0      0x00
#define SYNC1      0x100
#define SYNC2      0x200
#define SYNC3      0x300
#define TC         0x400
#define TCINT      0x800
/*************************************************************************/
/* DMA macros for global control register bit field assignments          */
/* e.g. DMA_ADDR->gcontrol.incsrc = INCREMENT;                           */
/* e.g. DMA_ADDR->gcontrol.incdst = INCREMENT;                           */
/* sets the appropriate bits of the DMA global control register         */
/*************************************************************************/
#define INCREMENT  1
#define DECREMENT  1
#define TERMINATE  1

#endif /* #ifndef _DMA30 */
```

B.3 serprt30.h Header Files

```
/*****************************************************************************/
/* serprt30.h v4.5                                                           */
/* Copyright (c) 1991 Texas Instruments Incorporated                         */
/*****************************************************************************/
#ifndef _SERPRT30
#define _SERPRT30

#ifndef gcontrol
#define gcontrol            _gctrl._intval
#endif

#ifndef gcontrol_bit
#define gcontrol_bit        _gctrl._bitval
#endif

#define s_x_control       _xctrl._intval
#define s_x_control_bit   _xctrl._bitval
#define s_r_control       _rctrl._intval
#define s_r_control_bit   _rctrl._bitval
#define s_rxt_control     _rxtctrl._intval
#define s_rxt_control_bit _rxtctrl._bitval
#define s_rxt_counter     _rxtcounter._intval
#define s_rxt_counter_bit _rxtcounter._bitval
#define s_rxt_period      _rxtperiod._intval
#define s_rxt_period_bit  _rxtperiod._bitval

/*****************************************************************************/
/* MACRO DEFINITIONS FOR SERIAL PORT BASE ADDRESS                            */
/*****************************************************************************/
#define SERIAL_PORT_ZERO    0
#define SERIAL_PORT_ONE     1
#define SERIAL_PORT_BASE    0x808040
#define SERIAL_PORT_SIZE    16
#define SERIAL_PORT_ADDR(n) ((volatile SERIAL_PORT_REG *) ((char *) \
                            SERIAL_PORT_BASE + (n)*SERIAL_PORT_SIZE))
```

```
/*************************************************************************/
/* STRUCTURE DEFINTION FOR SERIAL PORT GLOBAL CONTROL REGISTER           */
/*************************************************************************/
typedef union
{
    unsigned int _intval;
    struct
    {
         unsigned int rrdy     :1;                 /* Receive ready flag */
         unsigned int xrdy     :1;               /* Transmitt ready flag */
         unsigned int fsxout   :1;                  /* FSX configuration */
         unsigned int xsrempty :1;             /* Transm shift reg empty */
         unsigned int rsrfull  :1;             /* Receive registers full */
         unsigned int hs       :1;            /* Handshaking mode enable */
         unsigned int xclksrce :1;               /* Transm clock source */
         unsigned int rclksrce :1;              /* Receive clock source */
         unsigned int xvaren   :1;         /* Transm data rate signaling */
         unsigned int rvaren   :1;         /* Receiv data rate signaling */
         unsigned int xfsm     :1;             /* Transm frame sync mode */
         unsigned int rfsm     :1;             /* Receiv frame sync mode */
         unsigned int clkxp    :1;              /* Transm clock polarity */
         unsigned int clkrp    :1;              /* Receiv clock polarity */
         unsigned int dxp      :1;               /* Transm data polarity */
         unsigned int drp      :1;               /* Receiv data priority */
         unsigned int fsxp     :1;         /* Transm frame sync polarity */
         unsigned int fsrp     :1;         /* Receiv frame sync polarity */
         unsigned int xlen     :2;            /* Transm data word length */
         unsigned int rlen     :2;            /* Receiv data word length */
         unsigned int xtint    :1;       /* Transm timer interrupt enable */
         unsigned int xint     :1;            /* Transm interrupt enable */
         unsigned int rtint    :1;       /* Receiv timer interrupt enable */
         unsigned int rint     :1;            /* Receiv interrupt enable */
         unsigned int xreset   :1;                      /* Transm reset */
         unsigned int rreset   :1;                      /* Receiv reset */
         unsigned int r_rest   :4;                          /* reserved */
    } _bitval;
} SERIAL_PORT_CONTROL;
```

```
/*************************************************************************/
/* STRUCTURE DEFINTION FOR SERIAL PORT RECEIVE/TRANSMIT PORT CONTROL     */
/* REGISTER                                                              */
/*************************************************************************/
typedef union
{
    unsigned int _intval;
    struct
    {
         unsigned int clkfunc  :1;             /* Clock function control */
         unsigned int clki_o   :1;                  /* Clock i/o control */
         unsigned int clkdato  :1;               /* Data output on clock */
         unsigned int clkdati  :1;                /* Data input on clock */
         unsigned int dfunc    :1;              /* Data function control */
         unsigned int di_o     :1;                   /* Data i/o control */
         unsigned int ddatout  :1;               /* Data output on data */
         unsigned int ddatin   :1;                /* Data input on data */
         unsigned int fsfunc   :1;        /* Frame sync function control */
         unsigned int fsi_o    :1;             /* Frame sync i/o control */
         unsigned int fsdatout :1;          /* Data output on frame sync */
         unsigned int fsdatin  :1;           /* Data input on frame sync */
         unsigned int r_rest   :20;                        /* reserved */
    } _bitval;
} RX_PORT_CONTROL;
/*************************************************************************/
/* STRUCTURE DEFINTION FOR SERIAL PORT RECEIVE/TRANSMIT TIMER CONTROL    */
/* REGISTER                                                              */
/*************************************************************************/
typedef union
{
    unsigned int _intval;
    struct
    {
         unsigned int xgo      :1;     /* Reset and start transmit timer */
         unsigned int xhld_    :1;                /* Transm timer hold */
         unsigned int xcp_     :1;                /* Transm clock mode */
         unsigned int xclksrc  :1;              /* Transm clock source */
         unsigned int r_4      :1;                        /* reserved */
         unsigned int xtstat   :1;              /* Transmit timer stat */
         unsigned int rgo      :1;      /* Reset and start receive timer */
         unsigned int rhld_    :1;               /* Receive timer hold */
         unsigned int rcp_     :1;               /* Receive clock mode */
         unsigned int rclksrc  :1;             /* Receive clock source */
         unsigned int r_10     :1;                        /* reserved */
         unsigned int rtstat   :1;               /* Receive timer stat */
         unsigned int r_rest   :20;                        /* reserved */
    } _bitval;
} RX_TIMER_CONTROL;
```

```
/*************************************************************************/
/* STRUCTURE DEFINTION FOR SERIAL PORT RECEIVE/TRANSMIT TIMER COUNTER    */
/* REGISTER                                                              */
/*************************************************************************/
typedef union
{
    unsigned int _intval;
    struct
    {
         unsigned int x_counter:16;            /* Transmit timer counter */
         unsigned int r_counter:16;             /* Receive timer counter */
    } _bitval;
} RX_TIMER_COUNTER;

/*************************************************************************/
/* STRUCTURE DEFINTION FOR SERIAL PORT RECEIVE/TRANSMIT TIMER PERIOD     */
/* REGISTER                                                              */
/*************************************************************************/
typedef union
{
    unsigned int _intval;
    struct
    {
         unsigned int x_period :16;             /* Transmit timer period */
         unsigned int r_period :16;              /* Receive timer period */
    } _bitval;
} RX_TIMER_PERIOD;

/*************************************************************************/
/* STRUCTURE DEFINITION FOR SERIAL PORT REGISTERS                        */
/*************************************************************************/
typedef struct
{
    SERIAL_PORT_CONTROL _gctrl;            /* Serial port global control */
    unsigned int        reserved1;                          /* reserved */
    RX_PORT_CONTROL     _xctrl;                 /* Transm port control */
    RX_PORT_CONTROL     _rctrl;                /* Receive port control */
    RX_TIMER_CONTROL    _rxtctrl;       /* Receive/transmit timer control */
    RX_TIMER_COUNTER    _rxtcounter;    /* Receive/transmit timer counter */
    RX_TIMER_PERIOD     _rxtperiod;      /* Receive/transmit timer period */
    unsigned int        reserved2;                          /* reserved */
    unsigned int        x_data;             /* Serial port transmit data */
    unsigned int        reserved3[3];                       /* reserved */
    unsigned int        r_data;              /* Serial port receive data */
    unsigned int        reserved4[3];                       /* reserved */
} SERIAL_PORT_REG;
```

```
/*************************************************************************/
/* SERIAL PORT MACROS FOR BIT FIELD DEFINITIONS                          */
/*************************************************************************/
/* Serial port macros for global control register integer assignments    */
/* e.g. SERIAL_PORT_ADDR(0)->gcontrol = XVAREN | DXP | XLEN_16 set the   */
/* corresponding bits of serial port 0's global control register         */
/*************************************************************************/
#define RRDY            0x1
#define XRDY            0x2
#define FSXOUT          0x4
#define XSREMPTY        0x8
#define RSRFULL         0x10
#define HS              0x20
#define XCLKSRCE        0x40
#define RCLKSRCE        0x80
#define XVAREN          0x100
#define RVAREN          0x200
#define XFSM            0x400
#define RFSM            0x800
#define CLKXP           0x1000
#define CLKRP           0x2000
#define DXP             0x4000
#define DRP             0x8000
#define FSXP            0x10000
#define FSRP            0x20000
#define XLEN_8          0x00000
#define XLEN_16         0x40000
#define XLEN_24         0x80000
#define XLEN_32         0xC0000
#define RLEN_8          0x000000
#define RLEN_16         0x100000
#define RLEN_24         0x200000
#define RLEN_32         0x300000
#define XTINT           0x400000
#define XINT            0x800000
#define RTINT           0x1000000
#define RINT            0x2000000
#define XRESET          0x4000000
#define RRESET          0x8000000
```

```
#define INPUT_PIN       0
#define OUTPUT_PIN      1
#define DISABLED        0
#define ENABLED         1
#define EXTERNAL        0
#define INTERNAL        1
#define FIXED           0
#define VARIABLE        1
#define CONTINUOUS      1
#define STANDARD        0
#define ACTIVE_HIGH     0
#define ACTIVE_LOW      1
#define EIGHT_BITS      0
#define SIXTEEN_BITS    1
#define TWENTY_FOUR_BITS 2
#define THIRTY_TWO_BITS 3
#define RESET           0
#define UN_RESET        1

/*************************************************************************/
/* Serial port macros for fsx/dx/clkx control register integer           */
/* assignments, e.g. SERIAL_PORT_ADDR(0)->s_x_control = CLKXFUNC | DXFUNC */
/* sets the corresponding bits of serial port 0's fsx/dx/clkx control    */
/* register                                                              */
/*************************************************************************/
#define CLKXFUNC        0x1
#define CLKXI_O         0x2
#define CLKXDATOUT      0x4
#define CLKXDATIN       0x8
#define DXFUNC          0x10
#define DXI_O           0x20
#define DXDATOUT        0x40
#define DXDATIN         0x80
#define FSXFUNC         0x100
#define FSXI_O          0x200
#define FSXDATOUT       0x400
#define FSXDATIN        0x800

/*************************************************************************/
/* Serial port macros for fsr/dr/clkr control register integer           */
/* assignments, e.g. SERIAL_PORT_ADDR(0)->s_r_control = CLKRFUNC | DRFUNC */
/* sets the corresponding bits of serial port 0's fsr/dr/clkr control    */
/* register                                                              */
/*************************************************************************/
#define CLKRFUNC        0x1
#define CLKRI_O         0x2
#define CLKRDATOUT      0x4
#define CLKRDATIN       0x8
#define DRFUNC          0x10
#define DRI_O           0x20
#define DRDATOUT        0x40
#define DRDATIN         0x80
#define FSRFUNC         0x100
#define FSRI_O          0x200
#define FSRDATOUT       0x400
#define FSRDATIN        0x800
```

```
#define GENERAL_PURPOSE_IO 0
#define SERIAL_PORT_PIN 1

/*************************************************************************/
/* Serial port macros for receive/transmit timer control register integer */
/* assignments, e.g. SERIAL_PORT_ADDR(0)–>s_rxt_control = XGO | RGO       */
/* sets the corresponding bits of serial port 0's receive/transmit control*/
/* register                                                              */
/*************************************************************************/
#define XGO             0x1
#define XHLD_           0x2
#define XCP_            0x4
#define XCLKSRC         0x8
#define XTSTAT          0x20
#define RGO             0x40
#define RHLD_           0x80
#define RCP_            0x100
#define RCLKSRC         0x200
#define RSTAT           0x800

#endif /* #ifndef _SERPRT30 */
```

B.4 timer30.h Header File

```
/***************************************************************************/
/* timer30.h v4.5                                                           */
/* Copyright (c) 1991 Texas Instruments Incorporated                       */
/***************************************************************************/
#ifndef _TIMER30
#define _TIMER30

#ifndef gcontrol
#define gcontrol     _gctrl._intval
#endif

#ifndef gcontrol_bit
#define gcontrol_bit _gctrl._bitval
#endif

/***************************************************************************/
/* MACRO DEFINITIONS FOR TIMER BASE ADDRESS                                */
/***************************************************************************/
#define TIMER_ZERO   0
#define TIMER_ONE    1
#define TIMER_BASE   0x808020
#define TIMER_SIZE   16
#define TIMER_ADDR(n) ((volatile TIMER_REG *) ((char *) TIMER_BASE   \
                                               + (n)*TIMER_SIZE))

#define TMR_HLD_0    TIMER_ADDR(0)->gcontrol  &= ~HLD_
#define TMR_HLD_1    TIMER_ADDR(1)->gcontrol  &= ~HLD_

/***************************************************************************/
/* STRUCTURE DEFINITION FOR TIMER GLOBAL CONTROL REGISTER                  */
/***************************************************************************/
typedef union
{
    unsigned int _intval;
    struct
    {
         unsigned int func     :1;                 /* Timer pin function */
         unsigned int i_o      :1;                  /* Timer i/o control */
         unsigned int datout   :1;               /* Data output on timer */
         unsigned int datin    :1;                /* Data input on timer */
         unsigned int r_45     :2;                          /* reserved */
         unsigned int go       :1;              /* Reset and start timer */
         unsigned int hld_     :1;                        /* Counter hold */
         unsigned int cp_      :1;           /* Clock/pulse mode control */
         unsigned int clksrc   :1;                        /* Clock source */
         unsigned int inv      :1;                      /* Invert control */
         unsigned int tstat    :1;                        /* Timer status */
         unsigned int r_rest   :20;                           /* reserved */
    } _bitval;
} TIMER_CONTROL;
```

```
/*************************************************************************/
/* STRUCTURE DEFINITION FOR TIMER REGISTERS                              */
/*************************************************************************/
typedef struct
{
    TIMER_CONTROL     _gctrl;                    /* Timer global control */
    unsigned int      reserved1[3];                    /* reserved */
    unsigned int      counter;                      /* Timer counter */
    unsigned int      reserved2[3];                    /* reserved */
    unsigned int      period;                       /* Timer period */
} TIMER_REG;

/*************************************************************************/
/* TIMER MACROS FOR BIT FIELD DEFINITIONS                                */
/*************************************************************************/
/* Timer macros for global control register integer assignments          */
/* e.g. TIMER_ADDR(0)->gcontrol = FUNC | GO | HLD_ | CLKSRC sets the     */
/* corresponding bits of timer 0's global control register               */
/*************************************************************************/
#define FUNC       0x1
#define I_O        0x2
#define DATOUT     0x4
#define DATIN      0x8
#define GO         0x40
#define HLD_       0x80
#define CP_        0x100
#define CLKSRC     0x200
#define INV        0x400
#define TSTAT      0x800

/*************************************************************************/
/* Timer macros for global control register bit field assignments        */
/* e.g. TIMER_ADDR(0)->gcontrol_bit.func = TIMER_PIN;                    */
/*      TIMER_ADDR(0)->gcontrol_bit.go   = SET;                          */
/*      TIMER_ADDR(0)->gcontrol_bit.hld_ = SET;                          */
/*      TIMER_ADDR(0)->gcontrol_bit.clksrc = INTERNAL;                   */
/* sets the appropriate bits of timer 0's global control register.       */
/*************************************************************************/
#define TIMER_PIN  1
#define INPUT      0
#define OUTPUT     1
#define CLOCK_MODE 1
#define PULSE_MODE 0
#define INVERT     1
#define NON_INVERT 0
#define EXTERNAL   0
#define INTERNAL   1

#endif /* #ifndef _TIMER30 */
```