Texas Instruments

Application Report
SPRA575B - April 2000

TMS320C549 to TMS320C5402 Migration

Stephen Lau — Digital Signal Processing Solutions

ABSTRACT

This application report describes issues of interest related to migration from the TMS320VC549 to the TMS320VC5402. The objective of this application report is to indicate the differences between the two devices. Functions that are identical between the two devices are not included. For detailed information on the specific functions of either device, refer to the TMS320C549 data sheet (SPRS039), the TMS320C5402 data sheet (SPRS079), and the *TMS320C54x DSP Reference Set, Volume 1: CPU and Periperhals* (SPRU131).

Unless otherwise noted, the information contained in this application report should be considered ADVANCE INFORMATION on new products in the sampling or preproduction phase of development. Information and specifications in this document are subject to change without notice.

Contents

List of Figures

List of Tables

1 Package and Pinout

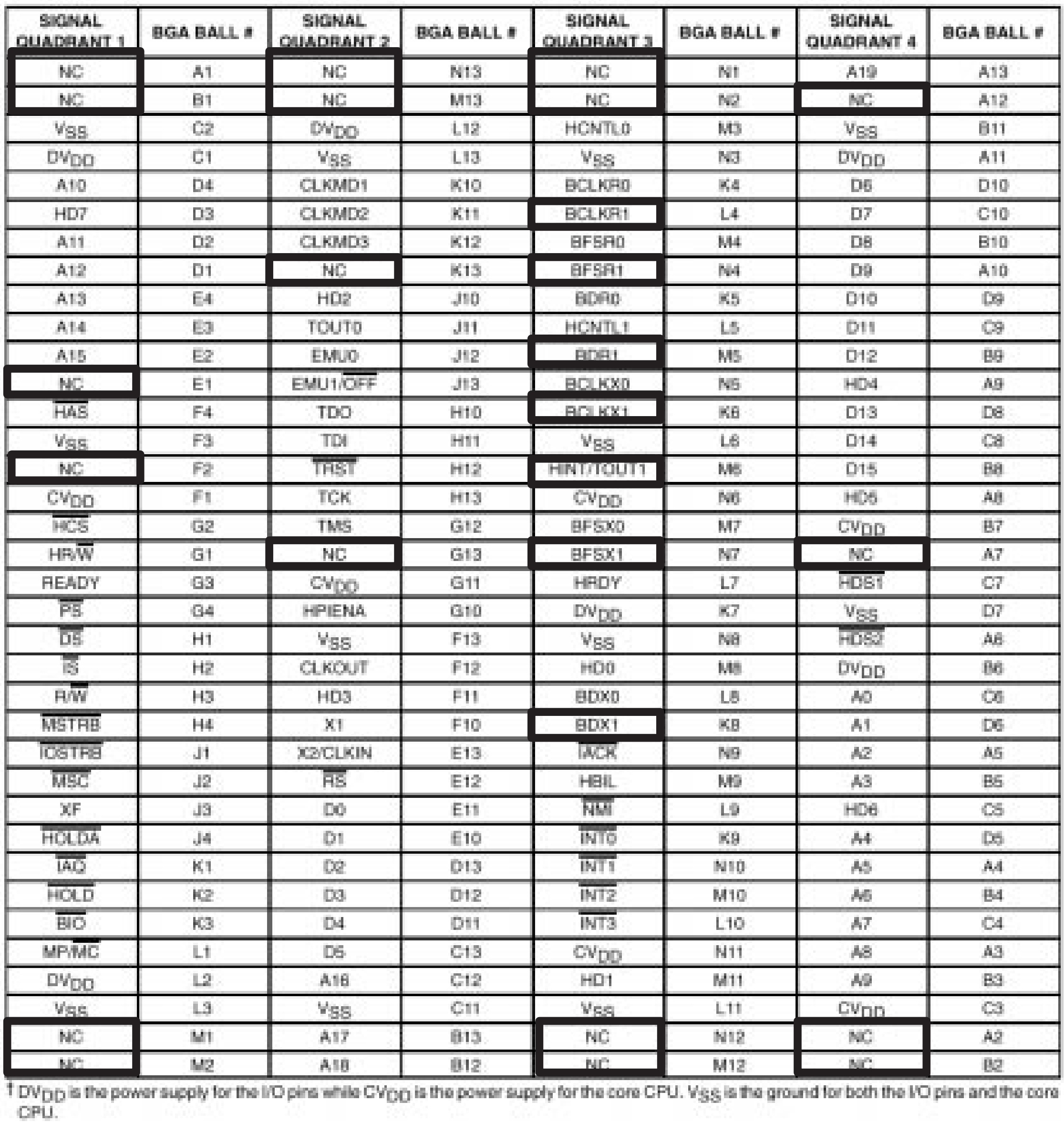
The ’5402 is footprint compatible with the ’549. The different packaging types are listed in Table 1.

Table 1. TMS320C549 and TMS320C5402 Package Types

’549	’5402	Definition
144-Pin TQFP (PGE suffix)	144-Pin TQFP (PGE suffix)	Plastic Thin Quad Flatpack
144-Pin μ*BGA (GGU Suffix	144-Pin μ*BGA (GGU Suffix)	microStar™ Ball Grid Array

The differences between the ’5402 and the ’549 are highlighted in Figure 1 (BGA package) and Figure 2 (TQFP package). The ’5402 has two McBSPs instead of the TDM and two BSPs that were available on the ’549. The pins previously associated with the TDM and BSP ports have been reassigned.

All of the pins labeled NC (no internal connection) on the ’5402 are internally isolated from the die, and the voltage placed on these pins should not have any affect on the function of the device. The pins labeled NC on the ’5402 (Figure 2) are 1, 2, 12, 15, 35, 36, 37, 38, 71, 72, 73, 74, 80, 90, 110, 126, 143, and 144. The NC pins should be treated as reserved, which means they should be left unconnected on a board design, if possible.



SIGNAL QUADRANT 1	BGA BALL #	SIGNAL QUADRANT 2	BGA BALL #	SIGNAL QUADRANT 3	BGA BALL #	SIGNAL QUADRANT 4	BGA BALL #
NC	A1	NC	N13	NC	N1	A19	A13
NC	B1	NC	M13	NC	N2	NC	A12
V_{SS}	C2	DV_{DD}	L12	HCNTL0	M3	V_{SS}	B11
DV_{DD}	C1	V_{SS}	L13	V_{SS}	N3	DV_{DD}	A11
A10	D4	CLKMD1	K10	BCLKR0	K4	D6	D10
HD7	D3	CLKMD2	K11	BCLKR1	L4	D7	C10
A11	D2	CLKMD3	K12	BFSR0	M4	D8	B10
A12	D1	NC	K13	BFSR1	N4	D9	A10
A13	E4	HD2	J10	BDR0	K5	D10	D9
A14	E3	TOUT0	J11	HCNTL1	L5	D11	C9
A15	E2	EMU0	J12	BDR1	M5	D12	B9
NC	E1	EMU1/$\overline{OFF}$	J13	BCLKX0	N5	HD4	A9
$\overline{HAS}$	F4	TDO	H10	BCLKX1	K6	D13	D8
V_{SS}	F3	TDI	H11	V_{SS}	L6	D14	C8
NC	F2	$\overline{TRST}$	H12	HINT/TOUT1	M6	D15	B8
CV_{DD}	F1	TCK	H13	CV_{DD}	N6	HD5	A8
$\overline{HCS}$	G2	TMS	G12	BFSX0	M7	CV_{DD}	B7
HR/$\overline{W}$	G1	NC	G13	BFSX1	N7	NC	A7
READY	G3	CV_{DD}	G11	HRDY	L7	$\overline{HDS1}$	C7
$\overline{PS}$	G4	HPIENA	G10	DV_{DD}	K7	V_{SS}	D7
$\overline{DS}$	H1	V_{SS}	F13	V_{SS}	N8	$\overline{HDS2}$	A6
$\overline{IS}$	H2	CLKOUT	F12	HD0	M8	DV_{DD}	B6
R/$\overline{W}$	H3	HD3	F11	BDX0	L8	A0	C6
$\overline{MSTRB}$	H4	X1	F10	BDX1	K8	A1	D6
$\overline{IOSTRB}$	J1	X2/CLKIN	E13	$\overline{IACK}$	N9	A2	A5
$\overline{MSC}$	J2	$\overline{RS}$	E12	HBIL	M9	A3	B5
XF	J3	D0	E11	$\overline{NMI}$	L9	HD6	C5
$\overline{HOLDA}$	J4	D1	E10	$\overline{INT0}$	K9	A4	D5
$\overline{IAQ}$	K1	D2	D13	$\overline{INT1}$	N10	A5	A4
$\overline{HOLD}$	K2	D3	D12	$\overline{INT2}$	M10	A6	B4
$\overline{BIO}$	K3	D4	D11	$\overline{INT3}$	L10	A7	C4
MP/$\overline{MC}$	L1	D5	C13	CV_{DD}	N11	A8	A3
DV_{DD}	L2	A16	C12	HD1	M11	A9	B3
V_{SS}	L3	V_{SS}	C11	V_{SS}	L11	CV_{DD}	C3
NC	M1	A17	B13	NC	N12	NC	A2
NC	M2	A18	B12	NC	M12	NC	B2

† DV_{DD} is the power supply for the I/O pins while CV_{DD} is the power supply for the core CPU. V_{SS} is the ground for both the I/O pins and the core CPU.

Figure 1. TMS320C5402 BGA Packaging Differences

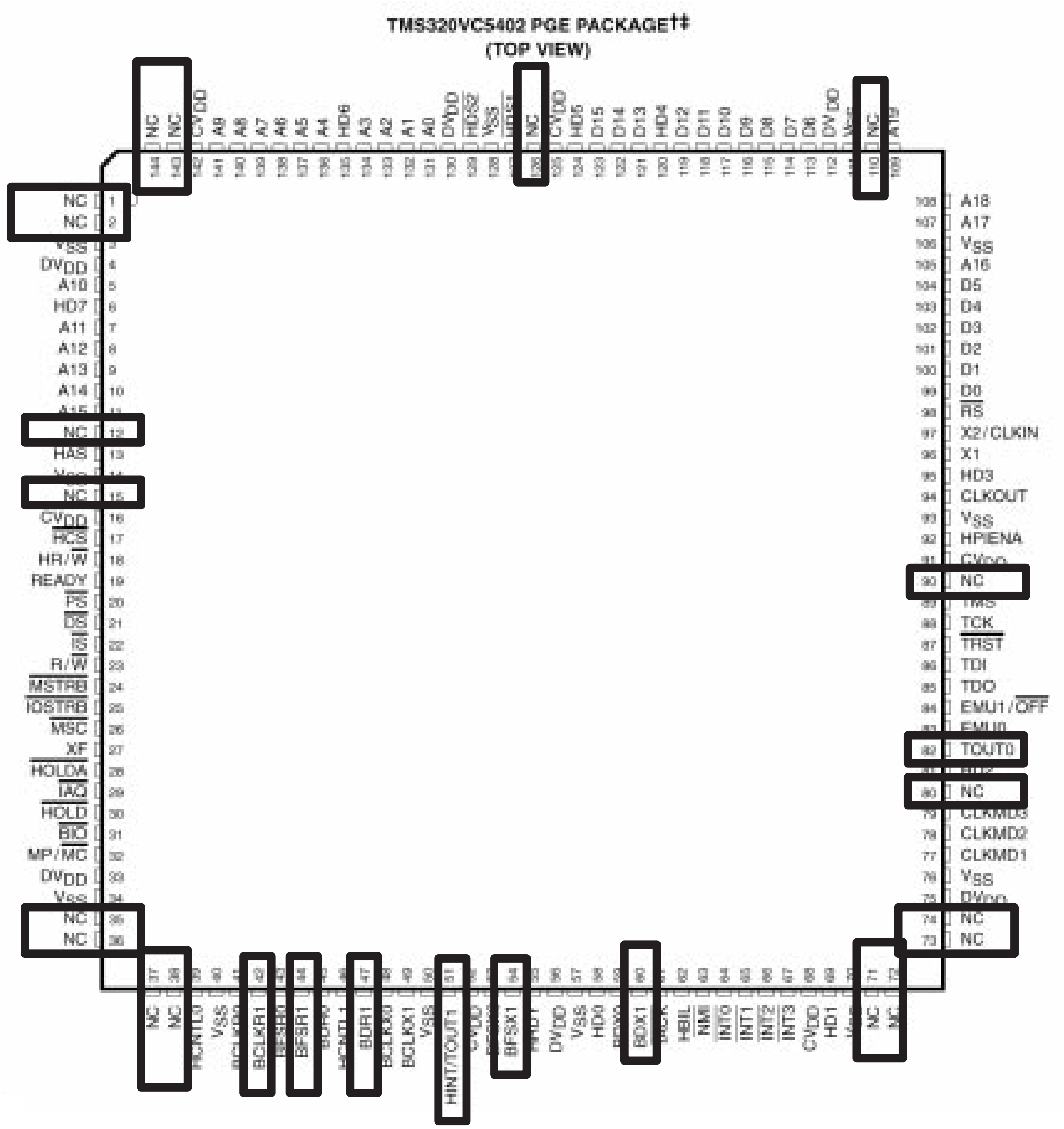


Figure 2. TMS320C5402 TQFP Packaging Differences

2 Power Supply

2.1 Voltage Levels

As shown in Table 2, the ’5402 requires a core voltage of 1.8 V, while the ’549 requires a core voltage of 2.5 V.

Table 2. TMS320C549 and TMS320C5402 Core and I/OVoltages

	’549	’5402
Core	2.5 V	1.8 V
I/O	3.3 V	3.3 V

2.2 Power-up Sequence

The ’5402 does not require specific power sequencing between the core supply and the I/O supply. However, systems should be designed to insure that neither supply is powered up for extended periods of time if the other supply is below the proper operating voltage. Excessive exposure to these conditions can adversely affect the long term reliability of the device.

System-level concerns such as bus contention may require supply sequencing to be implemented. In this case, the core supply should be powered up at the same time as, or prior to (and powered down after), the I/O buffers.

3 PLL Clocking Options

Table 3 shows the various clock ratios following reset for both the ’549 and the ’5402. For the ’549 at reset, most of the modes start the DSP in a divide-by-2 mode, which requires subsequent programming of the CLKMD register. For the ’5402 at reset, most of the modes start the DSP with a preset PLL multiplier.

Table 3. TMS320C549 and TMS320C5402 Clocking Differences

			’549		’5402	
CLKMD1	CLKMD2	CLKMD3	CLKMD at Reset	Clock Mode	CLKMD at Reset	Clock Mode
0	0	0	0000h	Divide by 2, external source	E007h	PLL X 15
0	0	1	1000h	Divide by 2, external source	9007h	PLL X 10
0	1	0	2000h	Divide by 2, external source	4007h	PLL X 5
1	0	0	4000h	Divide by 2, internal oscillator enabled	1007h	PLL X 2
1	1	0	6000h	Divide by 2, external source	F007h	PLL X 1
1	1	1	7000h	Divide by 2, internal oscillator enabled	0000h	1/2 (PLL disabled)
1	0	1	0007h	PLL X 1, external source	F000h	!/4 (PLL disabled)
0	1	1	–	Stop Mode	–	Reserved (bypass mode)

Table 4 shows some clocking examples for the ’549 and the ’5402. See the data sheets for PLL programming, and switching from DIV mode to PLL mode.

Table 4. Example CLKMD and PLL Programming

	’5402		’549	
Pin Name	25 MHz Oscillator	20 MHz XTAL	25 MHz Oscillator	20 MHz XTAL
CLKMD1	Hi	Lo	Hi	Hi
CLKMD2	Hi	Hi	Hi	Lo
CLKMD3	Lo	Lo	Lo	Lo
Initial CLKOUT frequency	25 MHz	100 MHz	12.5 MHz	10 MHz
Clock mode at reset	PLL X 1	PLL X 5	Divide by 2, with external source	Divide by 2, internal oscillator enabled
PLL multiplier for 100 MHz	4	5	4	5

3.1 CLKIN/X2

The ’549 CLKIN/X2 can be operated with an external clock or crystal with voltage levels of 3.3V. All revisions of the ’5402 can be operated with an external clock source, provided that the proper voltage levels be driven on the X2/CLKIN pin. On the ’5402, the X2/CLKIN pin is referenced to the device 1.8V power supply (CVdd), rather than the 3.3V I/O supply (DVdd). Thus, if an external clock is used, the input voltage must not exceed 1.8V. Please refer to the recommended operating conditions section of the datasheet for the allowable voltage levels of the X2/CLKIN pin.

4 Bank-Switching Control Register (BSCR)

Bit 2 of the ’5402 BSCR (Figure 3) is used to enable/disable the HPI-8 bus holders. This bit is reserved in the BSCR of the ’549.

	15–12	11	10–2	1	0
’549	BNKCMP	PS-DS	Reserved	BH	EXIO
	R/W	R/W		R/W	R/W

	15–12	11	10–3	2	1	0
’5402	BNKCMP	PS-DS	Reserved	HBH	BH	EXIO
	R/W-1111	R/W-1	R-0	R/W-0	R/W-0	R/W-0

Figure 3. Bank-Switching Control Register (BSCR)

5 External Parallel Interface

5.1 Software Wait-State Generator

There are no functional differences between the ’549 and ’5402 wait-state generators.

The ’5402 is capable of generating up to 14 software wait states. This is achieved with the software wait-state multiplier (SWSM) bit which, when set to 1, multiplies the programmed number of software wait states by 2.

The SWSM bit is located in the software wait-state control register (SWCR) at address 0x2B in data space. The structure of this register is shown in Figure 4. SWSM multiplies the number of software wait states in the SWWSR by 2. Therefore, the ’5402 can be programmed to generate 0–8, 10, 12, or 14 wait states. The SWSM bit is cleared at reset, making the wait-state generation scheme fully compatible with the ’549.

	15–1	0
SWCR (0x2B)	Reserved	SWSM
	R/W	R/W

Figure 4. Software Wait-State Control Register (SWCR)

There is an XSWR on the ’549 that performs the same function as the SWCR.

5.2 Software Wait-State Register (SWWSR)

The location and field definitions of the software wait-state register (SWWSR) are identical on the ’549 and ’5402.

5.3 Data Bus

The data bus buffer on the ’5402 is different from the data bus buffer on the ’549. This results in signals that are not as precise.

6 Memory Security

The ’5402 includes the standard C54x ROM/RAM security options that are defined in *TMS320C54x DSP Reference Set, Volume 1: CPU and Peripherals* (SPRU131). These security options are only available for devices with custom ROM, and no security is enabled on the standard ’5402 device. Also, the ’5402 security options do not affect the HPI-8 accesses to on-chip RAM (for example, the on-chip RAM cannot be secured from HPI-8 accesses.)

7 Timer

The ’5402 has two timers and the ’549 has only one timer. The two timers on the ’5402 are identical to the one on the ’549. The timer out (TOUT) signals are:

’549	’5402
HINT	HINT/TOUT1 (multiplexed)
TOUT	TOUT0

8 Direct Memory Access (DMA) Controller

The ’549 does not have a direct memory access (DMA) controller; however, the ’5402 does have a DMA controller. The DMA controller transfers data between regions in the memory map without intervention by the CPU. The DMA controller allows movement to and from internal memory, internal peripherals, or external devices to occur in the background of CPU operation.

The ’549 buffered serial port (BSP) has the capability to transfer data directly from the serial port to memory (autobuffering mode). This capability is extended in the ’5402 through the use of the DMA controller in conjunction with the multichannel BSP (McBSP). Refer to *TMS320C54x DSP Reference Set, Volume 5: Enhanced Peripherals* (SPRU302), for details.

8.1 Idle

The DMA controller will work properly in IDLE3 mode. For it to work properly, CLKIN must be maintained active during IDLE3. If it is turned on, a transfer is completed and the clocks are turned off again.

8.2 Priority

If the CPU has an access to memory pending, the DMA controller waits for the pending CPU access to be completed. The DMA controller does have higher priority than the CPU, but this priority is evaluated at the time a request is made to use a memory space.

8.3 Interrupts

With the addition of the McBSP and DMA controller, some of the new interrupts are multiplexed. See the ’5402 datasheet (SPRS079) for more information.

9 Interrupt Mask Register (IMR) and Interrupt Flag Register (IFR)

The interrupt mask register (IMR) and interrupt flag register (IFR) are slightly different between the ’549 and the ’5402. This is because of the addition of the DMA and McBSP. Table 5 shows the differences between the ’549 and ’5402.

Table 5. Interrupt Flag Register (IFR) and Interrupt Mask Register (IMR) for TMS320C549 and TMS320C5402

Register Bit Number	’549	’5402
15-14	reserved	reserved
13	bmint1	dmac5
12	bmint0	dmac4
11	bxint1	bxint1/dmac3
10	brint1	brint1/dmac2
9	hint	hpint
8	int3	int3
7	txnt	tint1/dmac1
6	trnt	dmac0
5	bxint0	bxint0
4	trint0	brint0
3	tint	tint0
2	int2	int2
1	int1	int1
0	int0	int0

10 Memory

The ’549 can address 8M-word of program memory, whereas the ’5402 can only address 1M-word of program memory. The pins for A20–A22 are not connected (NC) on the ’5402 as shown in Table 6. The granularity of the DARAM blocks is also different. The ’5402 has 8K blocks.

Table 6. TMS320C549 and TMS320C5402 Pinout Differences for Address Lines

’549	’5402
A20	NC
A21	NC
A22	NC

11 Memory-Mapped Registers

Table 7 lists the differences between the ’549 and the ’5402 memory-mapped registers. The primary difference is the use of subaddressing registers for the McBSP and the DMA.

Highlighted sections indicate differences between ’549 and ’5402.

Table 7. Differences Between TMS320C549 and TMS320C5402 Memory-Mapped Registers

HEX	’549	Description	’5402	Subaddress	Subaddress Description
0	IMR	Interrupt mask register	IMR		Interrupt mask register
1	IFR	Interrupt flag register	IFR		Interrupt flag register
2–5	–	Reserved for testing	–		Reserved for testing
6	ST0	Status register 0	ST0		Status register 0
7	ST1	Status register 1	ST1		Status register 1
8	AL	Accumulator A low word (15–0)	AL		Accumulator A low word (15–0)
9	AH	Accumulator A high word (31–16)	AH		Accumulator A high word (31–16)
A	AG	Accumulator A guard bits (39–32)	AG		Accumulator A guard bits (39–32)
B	BL	Accumulator B low word (15–0)	BL		Accumulator B low word (15–0)
C	BH	Accumulator B high word (31–16)	BH		Accumulator B high word (31–16)
D	BG	Accumulator B guard bits (39–32)	BG		Accumulator B guard bits (39–32)
E	TREG	Temporary register	TREG		Temporary register
F	TRN	Transition register	TRN		Transition register
10	AR0	Auxiliary register 0	AR0		Auxiliary register 0
11	AR1	Auxiliary register 1	AR1		Auxiliary register 1
12	AR2	Auxiliary register 2	AR2		Auxiliary register 2
13	AR3	Auxiliary register 3	AR3		Auxiliary register 3
14	AR4	Auxiliary register 4	AR4		Auxiliary register 4
15	AR5	Auxiliary register 5	AR5		Auxiliary register 5
16	AR6	Auxiliary register 6	AR6		Auxiliary register 6
17	AR7	Auxiliary register 7	AR7		Auxiliary register 7
18	SP	Stack pointer register	SP		Stack pointer register
19	BK	Circular buffer size register	BK		Circular buffer size register

Table 7. Differences Between TMS320C549 and TMS320C5402 Memory-Mapped Registers (Continued)

1A	BRC	Block repeat counter	BRC		Block repeat counter
1B	RSA	Block repeat start address	RSA		Block repeat start address
1C	REA	Block repeat end address	REA		Block repeat end address
1D	PMST	Processor mode status (PMST) register	PMST		Processor mode status (PMST) register
1E	XPC	Extended program page register	XPC		Extended program page register
1F	–	Reserved	–		Reserved
20	BDRR0	BSP 0 data-receive register	DRR20	–	McBSP0 data receive register
21	BDXR0	BSP 0 data-transmit register	DRR10	–	McBSP0 data receive register
22	BSPC0	BSP 0 control register	DXR20	–	McBSP0 data transmit register
23	BSPCE0	BSP 0 control extension register	DXR10	–	McBSP0 data transmit register
24	TIM	Timer count register	TIM	–	Timer0 register
25	PRD	Timer period register	PRD	–	Timer0 period
26	TCR	Timer control register	TCR	–	Timer0 control
27	–	Reserved	–	–	Reserved
28	SWWSR	External interface software wait-state register	SWWSR	–	Software wait-state register
29	BSCR	External interface bank-switching control register	BSCR	–	Bank-switching control register
2A	–	Reserved	–	–	Reserved
2B	–	Reserved	SWCR	–	Software wait-state control
2C	HPIC	HPI control register	HPIC	–	HPI control register
2D–2F	–	Reserved	–	–	Reserved
30	TRCV	TDM port data-receive register	TIM1	–	Timer1 register
31	TDXR	TDM port data-transmit register	PRD1	–	Timer1 period counter
32	TSPC	TDM serial port control register	TCR1	–	Timer1 control register
33	TCSR	TDM channel-select register	–	–	Reserved
34	TRTA	TDM receive/ transmit register	–	–	Reserved
35	TRAD	TDM receive/ address register	–	–	Reserved
36	–	Reserved	–	–	Reserved
37	–	Reserved	–	–	Reserved
38	AXR0	ABU 0 transmit-address register	SPSA0	–	McBSP0 sub-address register

39	BKX0	ABU 0 transmit-buffer-size register	SPCR10	00h	McBSP0 serial port control register 1
			SPCR20	01h	McBSP0 serial port control register 2
			RCR10	02h	McBSP0 receive control register 1
			RCR20	03h	McBSP0 receive control register 2
			XCR10	04h	McBSP0 transmit control register 1
			XCR20	05h	McBSP0 transmit control register 2
			SRGR10	06h	McBSP0 sample rate generator register 1
			SRGR20	07h	McBSP0 sample rate generator register 2
			MCR10	08h	McBSP0 multichannel register 1
			MCR20	09h	McBSP0 multichannel register 2
			RCERA0	0Ah	McBSP0 receive channel enable register partition A
			RCERB0	0Bh	McBSP0 receive channel enable register partition B
			XCERA0	0Ch	McBSP0 transmit channel enable register partition A
			XCERB0	0Dh	McBSP0 transmit channel enable register partition B
			PCR0	0Eh	McBSP0 pin control register
3A	ARR0	ABU 0 receive-address register	–	–	Reserved
3B	BKR0	ABU 0 receive-buffer-size register	–	–	Reserved
3C	AXR1	ABU 1 transmit-address register	GPIOCR	–	General purpose I/O pins control register
3D	BKX1	ABU 1 transmit-buffer-size register	GPIOSR	–	General purpose I/O pins status register
3E	ARR1	ABU 1 receive-address register	–	–	Reserved
3F	BKR1	ABU 1 receive-buffer-size register	–	–	Reserved
40	BDRR1	BSP 1 data-receive register	DRR21	–	McBSP1 Data receive register 1
41	BDXR1	BSP 1 data-transmit register	DRR11	–	McBSP1 Data receive register 2
42	BSPC1	BSP 1 control register	DXR21	–	McBSP1 Data transmit register 1

43	BSPCE1	BSP 1 control extension register	DXR11	–	McBSP1 Data transmit register 2
44h–47h	–	Reserved	–	–	Reserved
48h	–	Reserved	SPSA1	–	McBSP1 sub-address register
49h	–	Reserved	SPCR11	00h	McBSP1 control register 1
			SPCR21	01h	McBSP1 control register 2
			RCR11	02h	McBSP1 receive control register 1
			RCR21	03h	McBSP1 receive control register 2
			XCR11	04h	McBSP1 transmit control register 1
			XCR21	05h	McBSP1 transmit control register 2
			SRGR11	06h	McBSP1 sample rate generator register 1
			SRGR21	07h	McBSP1 sample rate generator register 2
			MCR11	08h	McBSP1 multichannel register 1
			MCR21	09h	McBSP1 multichannel register 2
			RCERA1	0Ah	McBSP1 receive channel enable register partition A
			RCERB1	0Bh	McBSP1 receive channel enable register partition B
			XCERA1	0Ch	McBSP1 transmit channel enable register partition A
			XCERB1	0Dh	McBSP1 transmit channel enable register partition B
			PCR1	0Eh	McBSP1 pin control register
4Ah–53h	–	Reserved	–	–	Reserved
54h	–	Reserved	DMPRE C	–	DMA channel priority and enable control register
55h	–	Reserved	DMSBA	–	DMA channel sub-address register
56h–57h	–	Reserved	DMSRC0	00h	DMA channel 0 source address register
			DMDST0	01h	DMA channel 0 destination address register
			DMCTR0	02h	DMA channel 0 element count register

	DMSFC0	03h	DMA channel 0 sync select and frame count register
	DMMCR 0	04h	DMA channel 0 transfer mode control register
	DMSRC1	05h	DMA channel 1 source address register
	DMDST1	06h	DMA channel 1 destination address register
	DMCTR1	07h	DMA channel 1 element count register
	DMSFC1	08h	DMA channel 1 sync select and frame count register
	DMMCR1	09h	DMA channel 1 transfer mode control register
	DMSRC2	0Ah	DMA channel 2 source address register
	DMDST2	0Bh	DMA channel 2 destination address register
	DMCTR2	0Ch	DMA channel 2 element count register
	DMSFC2	0Dh	DMA channel 2 sync select and frame count register
	DMMCR2	0Eh	DMA channel 2 transfer mode control register
	DMSRC3	0Fh	DMA channel 3 source address register
	DMDST3	10h	DMA channel 3 destination address register
	DMCTR3	11h	DMA channel 3 element count register
	DMSFC3	12h	DMA channel 3 sync select and frame count register
	DMMCR3	13h	DMA channel 3 transfer mode control register
	DMSRC4	14h	DMA channel 4 source address register
	DMDST4	15h	DMA channel 4 destination address register
	DMCTR4	16h	DMA channel 4 element count register
	DMSFC4	17h	DMA channel 4 sync select and frame count register

			DMMCR4	18h	DMA channel 4 transfer mode control register
			DMSRC5	19h	DMA channel 5 source address register
			DMDST5	1Ah	DMA channel 5 destination address register
			DMCTR5	1Bh	DMA channel 5 element count register
			DMSFC5	1Ch	DMA channel 5 sync select and frame count register
			DMMCR5	1Dh	DMA channel 5 transfer mode control register
			DMSRCP	1Eh	DMA source program page address(common channel)
			DMDSTP	1Fh	DMA destination program page address(common channel)
			DMIDX0	20h	DMA element index address register 0
			DMIDX1	21h	DMA element index address register 1
			DMFRI0	22h	DMA frame index register 0
			DMFRI1	23h	DMA frame index register 1
			DMGSA	24h	DMA global source address reload register
			DMGDA	25h	DMA global destination address reload register
			DMGCR	26h	DMA global count reload register
			DMGFR	27h	DMA global frame count reload register
58	CLKMD	Clock mode register	CLKMD	–	Clock mode register
59–5F	–	Reserved	–	–	Reserved

12 Host Port Interface

12.1 Memory Map

The ’5402 HPI-8 can access all of the internal memory, except for the memory-mapped registers (0–5Fh). The ’5402 has this ability because the HPI now accesses memory through the DMA controller. The ’549 is limited to a 2K-word DARAM block. This is illustrated in Figure 5.

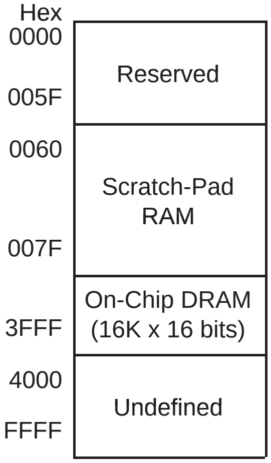
’549	’5402
2K-word DARAM block at 1000h to 17FFh in data memory.	 Hex 0000 005F Reserved 0060 Scratch-Pad RAM 007F On-Chip DRAM (16K x 16 bits) 3FFF 4000 Undefined FFFF

Figure 5. TMS320C549 and TMS320C5402 Host Port Interface (HPI) Memory Maps

The address autoincrement function on the ’549 HPI is available on the ’5402 HPI-8. It behaves identically to the autoincrement feature of the ’549 HPI.

If the HPI-8 attempts to access areas of the HPI memory map marked as reserved, the data read from these locations will be unpredictable. The HPI-8 can only access on-chip memory and will not go to external memory.

The ’5402 does not have the XHPIA and HPIENA bits in the HPIC (HPI control register).

12.2 Host-Only Mode (HOM)

HPI boot occurs when the device comes out of reset. The ’5402 HPI-8 does not have host-only mode (HOM), whereas the ’549 does have HOM. HOM gives the host exclusive access to the on-chip RAM.

Without HOM, you cannot load code during reset on the ’5402. This is because the ’5402 HPI-8 does not function during reset. The bootloader supports HPI-8 accesses after reset.

13 Bootloader/ROM Contents

13.1 Entry Point for HPI Boot Mode

The ’5402 bootloader supports a programmable entry point for the HPI boot mode, which differs from the ’549. After completing the bootload process, the host must perform an additional HPI-8 write to load the entry point to location 07Fh of on-chip RAM. In contrast, the ’549 bootloader always starts executing code at 2000h in HPI boot mode.

13.2 Serial EEPROM Bootloading (TMX/TMS difference)

The ’5402 bootloader will support a SPI EEPROM boot mode. This feature will be available in the ROM of TMS devices (romcode# D16780). The initial version of the bootloader available on TMX samples does not include the SPI EEPROM boot mode.

13.3 Standard Serial Port Bootloading

The 5402 features a serial port boot mode which can be used to load code from a serial master device. In this mode, the multi-channel buffered serial port (McBSP) is configured to emulate the TI standard serial port in burst mode. Like the ’549 The ’5402 is capable of both 8-bit and 16-bit standard serial boot modes.

The ’5402 expects the same boot table format as the ’548/549 in both 16-bit and 8-bit standard serial boot modes. The SPC, SPCE, ARR and BKR values in the boot table will be ignored by the ’5402 since these registers are not applicable.

On the ’5402, the McBSP0 is configured for dedicated 16-bit data width operation, while McBSP1 is configured for 8-bit data width operation. The configuration of the McBSP cannot be changed during the boot load as on the ’548/549. After reset, the XF pin is driven low to indicate the ’5402 is ready to receive serial data.

14 Multichannel Buffered Serial Port (McBSP)

The ’5402 features two multichannel buffered serial ports (McBSPs); whereas, the ’549 has three serial ports, two buffered serial ports (BSPs) and one time-division multiplexed (TDM) serial port. These differences have affected the pinout, resulting in the reassignment of the TDM pins. This is highlighted in Table 8. The second BSP on the ’549 is disconnected on the ’5402.

Table 8. TMS320C549 and TMS320C5402 Serial Port Pinout Mappings

’549	’5402
BCLKR0	BCLKR0
TCLKR	BCLKR1
BCLKX0	BCLKX0
TDR	BDR1
BDR0	BDR0
BFSR0	BFSR0
TFSR/TADD	BFSR1
BDX0	BDX0
TCLKX	BCLKX1
TDX	BDX1
BFSX0	BFSX0
TFSX/TFRM	BFSX1
BCLKR1	NC
BCLKX1	NC
BDR1	NC
BDX1	NC
BFSR1	NC
BFSX1	NC

The capabilities of the ABU mode available on the ’549 can be emulated with the McBSP and the DMA controller. There are no flags on the ’5402 DMA controller similar to the ’549 buffered serial port RH and XH flags. The position of the DMA transfers in the buffer can be determined by reading the source or destination address (whichever one is autobuffering). Refer to *TMS320C54x DSP Reference Set, Volume 5: Enhanced Peripherals* (SPRU302) for details.

14.1 CLKS Pins

The CLKS pins are not implemented on the ’5402. These pins are omitted to maintain the 144-pin footprint of previous devices. The McBSP can be set to use the internal CPU clock as a reference instead of the CLKS. Without this pin, the ’5402 does not comply with the AC’97 Audio Interface Standard.

15 Acknowledgement

A special acknowledgement goes to Nortel who helped drive the production of this application report and provided some very useful information.

Appendix A TMS320C549 and TMS320C5402 ASM/C Header File

```
/******************************************************************
 * 54x MMR
 * DEFINE MEMORY MAPPED REGISTERS
 * $Header: 3.1 1999/06/29 16:10:07 STEPHEN Exp $
 * Stephen Lau
 *
 * Possible Defines:
 *  C549             Causes this file to be ’549 specific (BSP/TDM/Timer)
 *  C5402            Causes this file to be ’5402 specific (McBSP/DMA/2 Timers/GPIO)
 *  ASMDEFS          Generates an Assembly .set file when cl500 –k –dASMDEFS
 *                   54xmmr.h is invoked. This file can then be used as an include
 *                   file for assembly functions
 *
 * Usage:
 * ex:
 *
 * Define:
 * #define XXX_BASE  0x01
 * #define XXX_ADDR  ((volatile XXX_REG *)  ((char *) XXX_BASE))
 *
 *
 *  typedef union {
 *                   struct {
 *                      unsigned int int2 :1;
 *                      unsigned int int1 :1;
 *                      unsigned int int0 :1;
 *                      } bitval;
 *                   unsigned int value;
 *  } XXX_REG;
 *
 * Declare:
 *
 * volatile XXX_REG *Something = XXX_ADDR;
 *
* Then:
 *
 * Something–>value = 12;
 * Something–>bitval.int0 = 0;
 *
 * Notes:
 *
 * With NO declaration, access to the locations seems to be possible!
 *
 * XXX_ADDR–>value = 12;
 *
 *
 * You can generate an .include ASM file by using the
 * CL500 –dASMDEFS –dC5402 –k 54xmmr.h
 *
/******************************************************************

/******************************************************************/
/* Check to see if 54xmmr.h has been previously included by       */
/* another header, if so, skip this and go on                     */
/******************************************************************/
```

```
#if !defined(__MMREGS)

#if ASMDEFS /* If defined, create .set directives for ASM usage */
#define ASM_SET(sym) asm(#sym”\t.set\t”VAL(sym))
#define VAL(sym) #sym
#endif /* ASMDEFS */

/******************************************************************/
/* Define Interrupt Flag and Interrupt Mask Registers             */
/******************************************************************/
#define IMR_BASE     0x00
#define IMR_ADDR     ((volatile IMR_REG *) ((char *) IMR_BASE))

#define IFR_BASE     0x01
#define IFR_ADDR     ((volatile IFR_REG *) ((char *) IFR_BASE))

#if ASMDEFS
ASM_SET(IMR_BASE);
ASM_SET(IFR_BASE);
#endif /* ASMDEFS */

#if C5402
typedef union {
                     struct {
                        unsigned int res     :2;
                        unsigned int dmac5   :1;
                        unsigned int dmac4   :1;
                        unsigned int bxint1  :1;
                        unsigned int brint1  :1;
                        unsigned int hint    :1;
                        unsigned int int3    :1;
                        unsigned int tint1   :1;
                        unsigned int dmac0   :1;
                        unsigned int bxint0  :1;
                        unsigned int brint0  :1;
                        unsigned int tint0   :1;
                        unsigned int int2    :1;
                        unsigned int int1    :1;
                        unsigned int int0    :1;
                        } bitval;
                     unsigned int value;
} IFR_REG;

typedef union {
                     struct {
                        unsigned int res     :2;
                        unsigned int dmac5   :1;
                        unsigned int dmac4   :1;
                        unsigned int bxint1  :1;
                        unsigned int brint1  :1;
                        unsigned int hint    :1;
                        unsigned int int3    :1;
                        unsigned int tint1   :1;
                        unsigned int dmac0   :1;
                        unsigned int bxint0  :1;
                        unsigned int brint0  :1;
                        unsigned int tint0   :1;
                        unsigned int int2    :1;
                        unsigned int int1    :1;
                        unsigned int int0    :1;
                        } bitval;
                     unsigned int value;
} IMR_REG;

#elif C549

typedef union {
                     struct {
                        unsigned int res     :2;
                        unsigned int bmint1  :1;
                        unsigned int bmint0  :1;
                        unsigned int bxint1  :1;
                        unsigned int brint1  :1;
                        unsigned int hint    :1;
                        unsigned int int3    :1;
                        unsigned int txnt    :1;
                        unsigned int trnt    :1;
                        unsigned int bxint0  :1;
                        unsigned int trint0  :1;
                        unsigned int tint    :1;
                        unsigned int int2    :1;
                        unsigned int int1    :1;
                        unsigned int int0    :1;
                        } bitval;
                     unsigned int value;
} IFR_REG;

typedef union {
                     struct {
                        unsigned int res     :2;
                        unsigned int bmint1  :1;
                        unsigned int bmint0  :1;
                        unsigned int bxint1  :1;
                        unsigned int brint1  :1;
                        unsigned int hint    :1;
                        unsigned int int3    :1;
                        unsigned int txnt    :1;
                        unsigned int trnt    :1;
                        unsigned int bxint0  :1;
                        unsigned int trint0  :1;
                        unsigned int tint    :1;
                        unsigned int int2    :1;
                        unsigned int int1    :1;
                        unsigned int int0    :1;
                        } bitval;
                     unsigned int value;
```

```
} IMR_REG;

#endif /* ’549 endif */

/*******************************************************************/
/*Status Registers                                                 */
/*******************************************************************/
#define ST0_BASE 0x06
#define ST1_BASE 0x07
#define ST0_ADDR ((volatile ST0_REG *) ((char *) ST0_BASE))
#define ST1_ADDR ((volatile ST1_REG *) ((char *) ST1_BASE))

#if ASMDEFS
ASM_SET(ST0_BASE);
ASM_SET(ST1_BASE);
#endif /* ASMDEFS */

typedef union {
                     struct {
                        unsigned int arp  :3;
                        unsigned int tc   :1;
                        unsigned int c    :1;
                        unsigned int ova  :1;
                        unsigned int ovb  :1;
                        unsigned int dp   :9;
                        } bitval;
                     unsigned int value;
} ST0_REG;

typedef union {
                     struct {
                        unsigned int braf :1;
                        unsigned int cpl  :1;
                        unsigned int xf   :1;
                        unsigned int hm   :1;
                        unsigned int intm :1;
                        unsigned int zero :1;
                        unsigned int ovm  :1;
                        unsigned int sxm  :1;
                        unsigned int c16  :1;
                        unsigned int frct :1;
                        unsigned int cmpt :1;
                        unsigned int asmm :5;
                        } bitval;
                        unsigned int value;
} ST1_REG;
```

```
/******************************************************************/
/*Accumulators                                                    */
/******************************************************************/
#define AL_BASE 0x08
#define AH_BASE 0x09
#define AG_BASE 0x0A
#define BL_BASE 0x0B
#define BH_BASE 0x0C
#define BG_BASE 0x0D

#define AL_REG (*(volatile unsigned int *)AL_BASE) /* ACCUMULATOR A Low Word      */
#define AH_REG (*(volatile unsigned int *)AH_BASE) /* ACCUMULATOR A High Word     */
#define AG_REG (*(volatile unsigned int *)AG_BASE) /* ACCUMULATOR A Guard Bits    */
#define BL_REG (*(volatile unsigned int *)BL_BASE) /* ACCUMULATOR B Low Word      */
#define BH_REG (*(volatile unsigned int *)BH_BASE) /* ACCUMULATOR B High Word     */
#define BG_REG (*(volatile unsigned int *)BG_BASE) /* ACCUMULATOR B Guard Bits    */

#if ASMDEFS
ASM_SET(AL_BASE);
ASM_SET(AH_BASE);
ASM_SET(AG_BASE);
ASM_SET(BL_BASE);
ASM_SET(BH_BASE);
ASM_SET(BG_BASE);
#endif /* ASMDEFS */

/******************************************************************/
/* Registers                                                      */
/******************************************************************/
#define TREG_BASE    0x0E
#define TRN_BASE     0x0F
#define AR0_BASE     0x10
#define AR1_BASE     0x11
#define AR2_BASE     0x12
#define AR3_BASE     0x13
#define AR4_BASE     0x14
#define AR5_BASE     0x15
#define AR6_BASE     0x16
#define AR7_BASE     0x17
#define SP_BASE      0x18
#define BK_BASE      0x19
#define BRC_BASE     0x1A
#define RSA_BASE     0x1B
#define REA_BASE     0x1C

#define TREG_REG (*(volatile unsigned int *)TREG_BASE) /* Temporary Register      */
#define TRN_REG  (*(volatile unsigned int *)TRN_BASE)  /* TRN_ransition Register  */
#define AR0_REG  (*(volatile unsigned int *)AR0_BASE)  /* AR0_uxiliary Register 0 */
#define AR1_REG  (*(volatile unsigned int *)AR1_BASE)  /* Auxiliary Register 1    */
#define AR2_REG  (*(volatile unsigned int *)AR2_BASE)  /* Auxiliary Register 2    */
#define AR3_REG  (*(volatile unsigned int *)AR3_BASE)  /* Auxiliary Register 3    */
#define AR4_REG  (*(volatile unsigned int *)AR4_BASE)  /* Auxiliary Register 4    */
#define AR5_REG  (*(volatile unsigned int *)AR5_BASE)  /* Auxiliary Register 5    */
```

```
#define AR6_REG  (*(volatile unsigned int *)AR6_BASE)  /* Auxiliary Register 6          */
#define AR7_REG  (*(volatile unsigned int *)AR7_BASE)  /* Auxiliary Register 7          */
#define SP_REG   (*(volatile unsigned int *)SP_BASE)   /* Stack Pointer                 */
#define BK_REG   (*(volatile unsigned int *)BK_BASE)   /* Circular Buffer size register */
#define BRC_REG  (*(volatile unsigned int *)BRC_BASE)  /* Block Repeat Counter          */
#define RSA_REG  (*(volatile unsigned int *)RSA_BASE)  /* Block Repeat Start Address    */
#define REA_REG  (*(volatile unsigned int *)REA_BASE)  /* Block Repeat End Address      */

#if ASMDEFS
ASM_SET(TREG_BASE);
ASM_SET(TRN_BASE);
ASM_SET(AR0_BASE);
ASM_SET(AR1_BASE);
ASM_SET(AR2_BASE);
ASM_SET(AR3_BASE);
ASM_SET(AR4_BASE);
ASM_SET(AR5_BASE);
ASM_SET(AR6_BASE);
ASM_SET(AR7_BASE);
ASM_SET(SP_BASE);
ASM_SET(BK_BASE);
ASM_SET(BRC_BASE);
ASM_SET(RSA_BASE);
ASM_SET(REA_BASE);
#endif /* ASMDEFS */

/*****************************************************************/
/*PMST                                                          */
/*****************************************************************/
#define PMST_BASE 0x1d
#define PMST_ADDR ((volatile PMST_REG *) ((char *) PMST_BASE))

#if ASMDEFS
ASM_SET(PMST_BASE);
#endif /* ASMDEFS */

typedef union {
                   struct {
                      unsigned int iptr     :9;
                      unsigned int mpmc     :1;
                      unsigned int ovly     :1;
                      unsigned int avis     :1;
                      unsigned int drom     :1;
                      unsigned int clkoff   :1;
                      unsigned int smul     :1;
                      unsigned int sst      :1;
                      } bitval;
                   unsigned int value;
} PMST_REG;
```

```
/******************************************************************/
/* Extended Program Counter –XPC register                         */
/******************************************************************/
#define XPC_BASE     0x1e
#define XPC_ADDR     *(volatile unsigned int *)XPC_BASE

#if ASMDEFS
ASM_SET(XPC_BASE);
#endif /* ASMDEFS */

/******************************************************************/
/* EXTERNAL BUS CONTROL REGISTERS                                 */
/******************************************************************/
#define SWWSR_BASE   0x28
#define SWWSR_ADDR   ((volatile SWWSR_REG *) ((char *) SWWSR_BASE))

#define BSCR_BASE    0x29
#define BSCR_ADDR    ((volatile BSCR_REG *)  ((char *) BSCR_BASE))

#if ASMDEFS
ASM_SET(SWWSR_BASE);
ASM_SET(BSCR_BASE);
#endif /* ASMDEFS */

#if C5402
#define SWCR_BASE    0x2B
#define SWCR_ADDR    ((volatile SWCR_REG *)  ((char *) SWCR_BASE))

#if ASMDEFS
ASM_SET(SWCR_BASE);
#endif /* ASMDEFS */

#elif C549
#define XWCR_BASE    0x2B
#define XWCR_ADDR    ((volatile XWCR_REG *)  ((char *) XWCR_BASE))

#if ASMDEFS
ASM_SET(XWCR_BASE);
#endif /* ASMDEFS */
#endif /* ’5402/549  endif */

typedef union {
                     struct {
                        unsigned int xpa     :1;
                        unsigned int io      :3;
                        unsigned int data_hi :3;
                        unsigned int data_low:3;
                        unsigned int prog_hi :3;
                        unsigned int prog_low:3;
                        } bitval;
                     unsigned int value;
} SWWSR_REG;
```

```
typedef union {
                     struct {
                        unsigned int bnkcmp  :4;
                        unsigned int psds    :1;
                        unsigned int resvd   :8;
                        unsigned int hbh     :1;
                        unsigned int bh      :1;
                        unsigned int exio    :1;
                        } bitval;
                     unsigned int value;
} BSCR_REG;

#if C5402
typedef union {
                     struct {
                        unsigned int reserved:15;
                        unsigned int swsm    :1;
                        } bitval;
                     unsigned int value;
} SWCR_REG;

#elif C549
typedef union {
                     struct {
                        unsigned int reserved:15;
                        unsigned int swsm    :1;
                        } bitval;
                     unsigned int value;
} XWCR_REG;
#endif /* ’5402/549 endif */

/**********************************************************************/
/* HOST PORT INTERFACE REGISTER ADDRESS                               */
/**********************************************************************/
#define HPIC_BASE    0x2c
#define HPIC_ADDR    ((volatile HPIC_REG *) ((char *) HPIC_BASE))

#if ASMDEFS
ASM_SET(HPIC_BASE);
#endif /* ASMDEFS */

typedef union {
                     struct {
                        unsigned int zero3   :3;
                        unsigned int xhpia2  :1;
                        unsigned int hint2   :1;
                        unsigned int dspint2 :1;
                        unsigned int zero2   :1;
                        unsigned int bob2    :1;
                        unsigned int hpiena  :1;
                        unsigned int rsvd    :2;
                        unsigned int xhpia   :1;
                        unsigned int hint    :1;
                        unsigned int dspint  :1;
                        unsigned int zero    :1;
                        unsigned int bob     :1;
                        } bitval;
                     unsigned int value;
} HPIC_REG;

#if C5402
/********************************************************************/
/* Structure for McBSP                                              */
/********************************************************************/

/*--------------------------------------------------------------------*/
/* McBSP 0 */
/*--------------------------------------------------------------------*/
#define DRR20_BASE   0x20
#define DRR10_BASE   0x21
#define DXR20_BASE   0x22
#define DXR10_BASE   0x23
#define SPSA0_BASE   0x38
#define SPCR10_BASE  0x39
#define SPCR20_BASE  0x39
#define RCR10_BASE   0x39
#define RCR20_BASE   0x39
#define XCR10_BASE   0x39
#define XCR20_BASE   0x39
#define SRGR10_BASE  0x39
#define SRGR20_BASE  0x39
#define MCR10_BASE   0x39
#define MCR20_BASE   0x39
#define RCERA0_BASE  0x39
#define RCERB0_BASE  0x39
#define XCERA0_BASE  0x39
#define XCERB0_BASE  0x39
#define PCR0_BASE    0x39

#define SPCR10_SUB   0x00
#define SPCR20_SUB   0x01
#define RCR10_SUB    0x02
#define RCR20_SUB    0x03
#define XCR10_SUB    0x04
#define XCR20_SUB    0x05
#define SRGR10_SUB   0x06
#define SRGR20_SUB   0x07
#define MCR10_SUB    0x08
#define MCR20_SUB    0x09
#define RCERA0_SUB   0x0A
#define RCERB0_SUB   0x0B
#define XCERA0_SUB   0x0C
#define XCERB0_SUB   0x0D
#define PCR0_SUB     0x0E
```

```
#define DRR20_ADDR   (*(volatile unsigned int *)DRR20_BASE)
#define DRR10_ADDR   (*(volatile unsigned int *)DRR10_BASE)
#define DXR20_ADDR   (*(volatile unsigned int *)DXR20_BASE)
#define DXR10_ADDR   (*(volatile unsigned int *)DXR10_BASE)
#define SPSA0_ADDR   (*(volatile unsigned int *)SPSA0_BASE)
#define SPCR10_ADDR  ((volatile SPCR1_REG *)  ((char *) SPCR10_BASE))
#define SPCR20_ADDR  (volatile SPCR2_REG *)   ((char *) SPCR20_BASE))
#define RCR10_ADDR   ((volatile RCR1_REG *)   ((char *) RCR10_BASE))
#define RCR20_ADDR   ((volatile RCR2_REG *)   ((char *) RCR20_BASE))
#define XCR10_ADDR   ((volatile XCR1_REG *)   ((char *) XCR10_BASE))
#define XCR20_ADDR   ((volatile XCR2_REG *)   ((char *) XCR20_BASE))
#define SRGR10_ADDR  ((volatile SRGR1_REG *)  ((char *) SRGR10_BASE))
#define SRGR20_ADDR  ((volatile SRGR2_REG *)  ((char *) SRGR20_BASE))
#define MCR10_ADDR   ((volatile MCR1_REG *)   ((char *) MCR10_BASE))
#define MCR20_ADDR   ((volatile MCR2_REG *)   ((char *) MCR20_BASE))
#define RCERA0_ADDR  ((volatile RCERA_REG *)  ((char *) RCERA0_BASE))
#define RCERB0_ADDR  ((volatile RCERB_REG *)  ((char *) RCERB0_BASE))
#define XCERA0_ADDR  ((volatile XCERA_REG *)  ((char *) XCERA0_BASE))
#define XCERB0_ADDR  ((volatile XCERB_REG *)  ((char *) XCERB0_BASE))
#define PCR0_ADDR    ((volatile PCR_REG *)    ((char *) PCR0_BASE))

#if ASMDEFS
ASM_SET(DRR20_BASE);
ASM_SET(DRR10_BASE);
ASM_SET(DXR20_BASE);
ASM_SET(DXR10_BASE);
ASM_SET(SPSA0_BASE);
ASM_SET(SPCR10_BASE);
ASM_SET(SPCR20_BASE);
ASM_SET(RCR10_BASE);
ASM_SET(RCR20_BASE);
ASM_SET(XCR10_BASE);
ASM_SET(XCR20_BASE);
ASM_SET(SRGR10_BASE);
ASM_SET(SRGR20_BASE);
ASM_SET(MCR10_BASE);
ASM_SET(MCR20_BASE);
ASM_SET(RCERA0_BASE);
ASM_SET(RCERB0_BASE);
ASM_SET(XCERA0_BASE);
ASM_SET(XCERB0_BASE);
ASM_SET(PCR0_BASE);
#endif /* ASMDEFS */

#if ASMDEFS
ASM_SET(SPCR10_SUB);
ASM_SET(SPCR20_SUB);
ASM_SET(RCR10_SUB);
ASM_SET(RCR20_SUB);
ASM_SET(XCR10_SUB);
ASM_SET(XCR20_SUB);
ASM_SET(SRGR10_SUB);
ASM_SET(SRGR20_SUB);
ASM_SET(MCR10_SUB);
ASM_SET(MCR20_SUB);
ASM_SET(RCERA0_SUB);
ASM_SET(RCERB0_SUB);
ASM_SET(XCERA0_SUB);
ASM_SET(XCERB0_SUB);
ASM_SET(PCR0_SUB);
#endif /* ASMDEFS */

/*-------------------------------------------------------------------*/
/* McBSP 1                                                           */
/*-------------------------------------------------------------------*/
#define DRR21_BASE   0x40
#define DRR11_BASE   0x41
#define DXR21_BASE   0x42
#define DXR11_BASE   0x43
#define SPSA1_BASE   0x48
#define SPCR11_BASE  0x49
#define SPCR21_BASE  0x49
#define RCR11_BASE   0x49
#define RCR21_BASE   0x49
#define XCR11_BASE   0x49
#define XCR21_BASE   0x49
#define SRGR11_BASE  0x49
#define SRGR21_BASE  0x49
#define MCR11_BASE   0x49
#define MCR21_BASE   0x49
#define RCERA1_BASE  0x49
#define RCERB1_BASE  0x49
#define XCERA1_BASE  0x49
#define XCERB1_BASE  0x49
#define PCR1_BASE    0x49

#define SPCR11_SUB   0x00
#define SPCR21_SUB   0x01
#define RCR11_SUB    0x02
#define RCR21_SUB    0x03
#define XCR11_SUB    0x04
#define XCR21_SUB    0x05
#define SRGR11_SUB   0x06
#define SRGR21_SUB   0x07
#define MCR11_SUB    0x08
#define MCR21_SUB    0x09
#define RCERA1_SUB   0x0A
#define RCERB1_SUB   0x0B
#define XCERA1_SUB   0x0C
#define XCERB1_SUB   0x0D
#define PCR1_SUB     0x0E
```

```
#define DRR21_ADDR   (*(volatile unsigned int *)DRR21_BASE)
#define DRR11_ADDR   (*(volatile unsigned int *)DRR11_BASE)
#define DXR21_ADDR   (*(volatile unsigned int *)DXR21_BASE)
#define DXR11_ADDR   (*(volatile unsigned int *)DXR11_BASE)
#define SPSA1_ADDR   (*(volatile unsigned int *)SPSA1_BASE)
#define SPCR11_ADDR  ((volatile SPCR1_REG *)  ((char *) SPCR11_BASE))
#define SPCR21_ADDR  ((volatile SPCR2_REG *)  ((char *) SPCR21_BASE))
#define RCR11_ADDR   ((volatile RCR1_REG *)   ((char *) RCR11_BASE))
#define RCR21_ADDR   ((volatile RCR2_REG *)   ((char *) RCR21_BASE))
#define XCR11_ADDR   ((volatile XCR1_REG *)   ((char *) XCR11_BASE))
#define XCR21_ADDR   ((volatile XCR2_REG *)   ((char *) XCR21_BASE))
#define SRGR11_ADDR  ((volatile SRGR1_REG *)  ((char *) SRGR11_BASE))
#define SRGR21_ADDR  ((volatile SRGR2_REG *)  ((char *) SRGR21_BASE))
#define MCR11_ADDR   ((volatile MCR1_REG *)   ((char *) MCR11_BASE))
#define MCR21_ADDR   ((volatile MCR2_REG *)   ((char *) MCR21_BASE))
#define RCERA1_ADDR  ((volatile RCERA_REG *)  ((char *) RCERA1_BASE))
#define RCERB1_ADDR  ((volatile RCERB_REG *)  ((char *) RCERB1_BASE))
#define XCERA1_ADDR  ((volatile XCERA_REG *)  ((char *) XCERA1_BASE))
#define XCERB1_ADDR  ((volatile XCERB_REG *)  ((char *) XCERB1_BASE))
#define PCR1_ADDR    ((volatile PCR_REG *)    ((char *) PCR1_BASE))

#if ASMDEFS
ASM_SET(DRR21_BASE);
ASM_SET(DRR11_BASE);
ASM_SET(DXR21_BASE);
ASM_SET(DXR11_BASE);
ASM_SET(SPSA1_BASE);
ASM_SET(SPCR11_BASE);
ASM_SET(SPCR21_BASE);
ASM_SET(RCR11_BASE);
ASM_SET(RCR21_BASE);
ASM_SET(XCR11_BASE);
ASM_SET(XCR21_BASE);
ASM_SET(SRGR11_BASE);
ASM_SET(SRGR21_BASE);
ASM_SET(MCR11_BASE);
ASM_SET(MCR21_BASE);
ASM_SET(RCERA1_BASE);
ASM_SET(RCERB1_BASE);
ASM_SET(XCERA1_BASE);
ASM_SET(XCERB1_BASE);
ASM_SET(PCR1_BASE);
#endif /* ASMDEFS */

#if ASMDEFS
ASM_SET(SPCR11_SUB);
ASM_SET(SPCR21_SUB);
ASM_SET(RCR11_SUB);
ASM_SET(RCR21_SUB);
ASM_SET(XCR11_SUB);
ASM_SET(XCR21_SUB);
ASM_SET(SRGR11_SUB);
ASM_SET(SRGR21_SUB);
ASM_SET(MCR11_SUB);
ASM_SET(MCR21_SUB);
ASM_SET(RCERA1_SUB);
ASM_SET(RCERB1_SUB);
ASM_SET(XCERA1_SUB);
ASM_SET(XCERB1_SUB);
ASM_SET(PCR1_SUB);
#endif /* ASMDEFS */

/*------------------------------------------------------------------*/
/* SPCR1                                                            */
/*------------------------------------------------------------------*/
typedef union {
                     struct {
                        unsigned int dlb     :1;
                        unsigned int rjust   :2;
                        unsigned int clkstp  :2;
                        unsigned int rsrvd   :3;
                        unsigned int dxena   :1;
                        unsigned int abis    :1;
                        unsigned int rintm   :2;
                        unsigned int rsyncerr:1;
                        unsigned int rfull   :1;
                        unsigned int rrdy    :1;
                        unsigned int rrst    :1;
                        } bitval;
                     unsigned int value;
} SPCR1_REG;

/*------------------------------------------------------------------*/
/* SPCR2                                                            */
/*------------------------------------------------------------------*/
typedef union {
                     struct {
                        unsigned int rsrvd   :6;
                        unsigned int free    :1;
                        unsigned int soft    :1;
                        unsigned int frst    :1;
                        unsigned int grst    :1;
                        unsigned int xintm   :2;
                        unsigned int xsyncerr:1;
                        unsigned int xempty  :1;
                        unsigned int xrdy    :1;
                        unsigned int xrst    :1;
                        } bitval;
                     unsigned int value;
} SPCR2_REG;
```

```
/*-------------------------------------------------------------------*/
/* PCR                                                               */
/*-------------------------------------------------------------------*/
typedef union {
                     struct {
                        unsigned int rsrvd1     :2;
                        unsigned int xioen      :1;
                        unsigned int rioen      :1;
                        unsigned int fsxm       :1;
                        unsigned int fsrm       :1;
                        unsigned int clkxm      :1;
                        unsigned int clkrm      :1;
                        unsigned int rsrvd2     :1;
                        unsigned int clks_stat  :1;
                        unsigned int dx_stat    :1;
                        unsigned int dr_stat    :1;
                        unsigned int fsxp       :1;
                        unsigned int fsrp       :1;
                        unsigned int clkxp      :1;
                        unsigned int clkrp      :1;
                        } bitval;
                     unsigned int value;
} PCR_REG;

/*-------------------------------------------------------------------*/
/* RCR1                                                              */
/*-------------------------------------------------------------------*/
typedef union {
                     struct {
                        unsigned int rsrvd1  :1;
                        unsigned int rfrlen1 :7;
                        unsigned int rwdlen1 :3;
                        unsigned int rsrvd2  :5;
                        } bitval;
                     unsigned int value;
} RCR1_REG;

/*-------------------------------------------------------------------*/
/* RCR2                                                              */
/*-------------------------------------------------------------------*/
typedef union {
                     struct {
                        unsigned int rphase  :1;
                        unsigned int rfrlen2 :7;
                        unsigned int rwdlen2 :3;
                        unsigned int rcompand:2;
                        unsigned int rfig    :1;
                        unsigned int rdatdly :2;
                        } bitval;
                     unsigned int value;
} RCR2_REG;
```

```
/*-------------------------------------------------------------------*/
/* XCR1                                                              */
/*-------------------------------------------------------------------*/
typedef union {
                     struct {
                        unsigned int rsrvd1  :1;
                        unsigned int xfrlen1 :7;
                        unsigned int xwdlen1 :3;
                        unsigned int rsrvd2  :5;
                        } bitval;
                     unsigned int value;
} XCR1_REG;

/*-------------------------------------------------------------------*/
/* XCR2                                                              */
/*-------------------------------------------------------------------*/
typedef union {
                     struct {
                        unsigned int xphase  :1;
                        unsigned int xfrlen2 :7;
                        unsigned int xwdlen2 :3;
                        unsigned int xcompand:2;
                        unsigned int xfig    :1;
                        unsigned int xdatdly :2;
                        } bitval;
                     unsigned int value;
} XCR2_REG;

/*-------------------------------------------------------------------*/
/* SRGR1                                                             */
/*-------------------------------------------------------------------*/
typedef union {
                     struct {
                        unsigned int fwid    :8;
                        unsigned int clkdiv  :8;
                        } bitval;
                     unsigned int value;
} SRGR1_REG;

/*-------------------------------------------------------------------*/
/* SRGR2                                                             */
/*-------------------------------------------------------------------*/
typedef union {
                     struct {
                        unsigned int gsync   :1;
                        unsigned int clksp   :1;
                        unsigned int clksm   :1;
                        unsigned int fsgm    :1;
                        unsigned int fper    :12;
                        } bitval;
                     unsigned int value;
} SRGR2_REG;
```

```
/*-------------------------------------------------------------------*/
/* MCR1                                                              */
/*-------------------------------------------------------------------*/
typedef union {
                     struct {
                        unsigned int rsrvd1  :7;
                        unsigned int rpbblk  :2;
                        unsigned int rpablk  :2;
                        unsigned int rcblk   :3;
                        unsigned int rsrvd2  :1;
                        unsigned int rmcm    :1;
                        } bitval;
                     unsigned int value;
} MCR1_REG;

/*-------------------------------------------------------------------*/
/* MCR2                                                              */
/*-------------------------------------------------------------------*/
typedef union {
                     struct {
                        unsigned int rsrvd1  :7;
                        unsigned int xpbblk  :2;
                        unsigned int xpablk  :2;
                        unsigned int xcblk   :3;
                        unsigned int xmcm    :2;
                        } bitval;
                     unsigned int value;
} MCR2_REG;

/*-------------------------------------------------------------------*/
/* RCERA                                                             */
/*-------------------------------------------------------------------*/
typedef union {
                     struct {
                        unsigned int RCEA15  :1;
                        unsigned int RCEA14  :1;
                        unsigned int RCEA13  :1;
                        unsigned int RCEA12  :1;
                        unsigned int RCEA11  :1;
                        unsigned int RCEA10  :1;
                        unsigned int RCEA9   :1;
                        unsigned int RCEA8   :1;
                        unsigned int RCEA7   :1;
                        unsigned int RCEA6   :1;
                        unsigned int RCEA5   :1;
                        unsigned int RCEA4   :1;
                        unsigned int RCEA3   :1;
                        unsigned int RCEA2   :1;
                        unsigned int RCEA1   :1;
                        unsigned int RCEA0   :1;
                        } bitval;
                     unsigned int value;
} RCERA_REG;
```

```
/*-------------------------------------------------------------------*/
/* RCERB                                                             */
/*-------------------------------------------------------------------*/
typedef union {
                     struct {
                        unsigned int RCEB15  :1;
                        unsigned int RCEB14  :1;
                        unsigned int RCEB13  :1;
                        unsigned int RCEB12  :1;
                        unsigned int RCEB11  :1;
                        unsigned int RCEB10  :1;
                        unsigned int RCEB9   :1;
                        unsigned int RCEB8   :1;
                        unsigned int RCEB7   :1;
                        unsigned int RCEB6   :1;
                        unsigned int RCEB5   :1;
                        unsigned int RCEB4   :1;
                        unsigned int RCEB3   :1;
                        unsigned int RCEB2   :1;
                        unsigned int RCEB1   :1;
                        unsigned int RCEB0   :1;
                        } bitval;
                     unsigned int value;
} RCERB_REG;

/*-------------------------------------------------------------------*/
/* XCERA                                                             */
/*-------------------------------------------------------------------*/
typedef union {
                     struct {
                        unsigned int XCEA15  :1;
                        unsigned int XCEA14  :1;
                        unsigned int XCEA13  :1;
                        unsigned int XCEA12  :1;
                        unsigned int XCEA11  :1;
                        unsigned int XCEA10  :1;
                        unsigned int XCEA9   :1;
                        unsigned int XCEA8   :1;
                        unsigned int XCEA7   :1;
                        unsigned int XCEA6   :1;
                        unsigned int XCEA5   :1;
                        unsigned int XCEA4   :1;
                        unsigned int XCEA3   :1;
                        unsigned int XCEA2   :1;
                        unsigned int XCEA1   :1;
                        unsigned int XCEA0   :1;
                        } bitval;
                     unsigned int value;
} XCERA_REG;
```

```
/*-------------------------------------------------------------------*/
/* XCERB                                                             */
/*-------------------------------------------------------------------*/
typedef union {
                     struct {
                        unsigned int XCEB15  :1;
                        unsigned int XCEB14  :1;
                        unsigned int XCEB13  :1;
                        unsigned int XCEB12  :1;
                        unsigned int XCEB11  :1;
                        unsigned int XCEB10  :1;
                        unsigned int XCEB9   :1;
                        unsigned int XCEB8   :1;
                        unsigned int XCEB7   :1;
                        unsigned int XCEB6   :1;
                        unsigned int XCEB5   :1;
                        unsigned int XCEB4   :1;
                        unsigned int XCEB3   :1;
                        unsigned int XCEB2   :1;
                        unsigned int XCEB1   :1;
                        unsigned int XCEB0   :1;
                        } bitval;
                     unsigned int value;
} XCERB_REG;


/*********************************************************************/
/* Structure for DMA                                                 */
/*********************************************************************/
#define DMAPREC_BASE 0x54
#define DMAPREC_ADDR ((volatile DMPREC_REG *) ((char *) DMAPREC_BASE))

#define DMSBA_BASE   0x55
#define DMSBA_ADDR   (*(volatile unsigned int *) DMSBA_BASE)

#define DMSBAI_BASE  0x56 /* Auto-incrementing Sub-Address Register */
#define DMSBAI_ADDR  (*(volatile unsigned int *) DMSBAI_BASE)

#define DMSBANOI_BASE0x57 /* Sub-Address Register without Auto-increment */
#define DMSBANOI_ADDR(*(volatile unsigned int *) DMSBANOI_BASE)

#if ASMDEFS
ASM_SET(DMAPREC_BASE);
ASM_SET(DMSBA_BASE);
#endif /* ASMDEFS */

/* Sub addressing offsets */
#define DMSRC0_SUB   0x00
#define DMDST0_SUB   0x01
#define DMCTR0_SUB   0x02
#define DMSFC0_SUB   0x03
#define DMMCR0_SUB   0x04
#define DMSRC1_SUB   0x05
```

```
#define DMDST1_SUB   0x06
#define DMCTR1_SUB   0x07
#define DMSFC1_SUB   0x08
#define DMMCR1_SUB   0x09
#define DMSRC2_SUB   0x0A
#define DMDST2_SUB   0x0B
#define DMCTR2_SUB   0x0C
#define DMSFC2_SUB   0x0D
#define DMMCR2_SUB   0x0E
#define DMSRC3_SUB   0x0F
#define DMDST3_SUB   0x10
#define DMCTR3_SUB   0x11
#define DMSFC3_SUB   0x12
#define DMMCR3_SUB   0x13
#define DMSRC4_SUB   0x14
#define DMDST4_SUB   0x15
#define DMCTR4_SUB   0x16
#define DMSFC4_SUB   0x17
#define DMMCR4_SUB   0x18
#define DMSRC5_SUB   0x19
#define DMDST5_SUB   0x1A
#define DMCTR5_SUB   0x1B
#define DMSFC5_SUB   0x1C
#define DMMCR5_SUB   0x1D
#define DMSRCP_SUB   0x1E
#define DMDSTP_SUB   0x1F
#define DMIDX0_SUB   0x20
#define DMIDX1_SUB   0x21
#define DMFRI0_SUB   0x22
#define DMFRI1_SUB   0x23
#define DMGSA_SUB    0x24
#define DMGDA_SUB    0x25
#define DMGCR_SUB    0x26
#define DMGFR_SUB    0x27

#if ASMDEFS
ASM_SET(DMSRC0_SUB);
ASM_SET(DMDST0_SUB);
ASM_SET(DMCTR0_SUB);
ASM_SET(DMSFC0_SUB);
ASM_SET(DMMCR0_SUB);
ASM_SET(DMSRC1_SUB);
ASM_SET(DMDST1_SUB);
ASM_SET(DMCTR1_SUB);
ASM_SET(DMSFC1_SUB);
ASM_SET(DMMCR1_SUB);
ASM_SET(DMSRC2_SUB);
ASM_SET(DMDST2_SUB);
ASM_SET(DMCTR2_SUB);
ASM_SET(DMSFC2_SUB);
ASM_SET(DMMCR2_SUB);
ASM_SET(DMSRC3_SUB);
ASM_SET(DMDST3_SUB);
ASM_SET(DMCTR3_SUB);
ASM_SET(DMSFC3_SUB);
ASM_SET(DMMCR3_SUB);
ASM_SET(DMSRC4_SUB);
ASM_SET(DMDST4_SUB);
ASM_SET(DMCTR4_SUB);
ASM_SET(DMSFC4_SUB);
ASM_SET(DMMCR4_SUB);
ASM_SET(DMSRC5_SUB);
ASM_SET(DMDST5_SUB);
ASM_SET(DMCTR5_SUB);
ASM_SET(DMSFC5_SUB);
ASM_SET(DMMCR5_SUB);
ASM_SET(DMSRCP_SUB);
ASM_SET(DMDSTP_SUB);
ASM_SET(DMIDX0_SUB);
ASM_SET(DMIDX1_SUB);
ASM_SET(DMFRI0_SUB);
ASM_SET(DMFRI1_SUB);
ASM_SET(DMGSA_SUB);
ASM_SET(DMGDA_SUB);
ASM_SET(DMGCR_SUB);
ASM_SET(DMGFR_SUB);
#endif /* ASMDEFS */

/* Define the base addresses for auto–incrementing              */
/* Autoincrementing addresses will be denotated with a A ending */
#define DMSRC0_BASEA 0x56
#define DMSRC1_BASEA 0x56
#define DMSRC2_BASEA 0x56
#define DMSRC3_BASEA 0x56
#define DMSRC4_BASEA 0x56
#define DMSRC5_BASEA 0x56

#define DMDST0_BASEA 0x56
#define DMDST1_BASEA 0x56
#define DMDST2_BASEA 0x56
#define DMDST3_BASEA 0x56
#define DMDST4_BASEA 0x56
#define DMDST5_BASEA 0x56

#define DMCTR0_BASEA 0x56
#define DMCTR1_BASEA 0x56
#define DMCTR2_BASEA 0x56
#define DMCTR3_BASEA 0x56
#define DMCTR4_BASEA 0x56
#define DMCTR5_BASEA 0x56

#define DMSFC0_BASEA 0x56
#define DMSFC1_BASEA 0x56
#define DMSFC2_BASEA 0x56
#define DMSFC3_BASEA 0x56
#define DMSFC4_BASEA 0x56
#define DMSFC5_BASEA 0x56
```

```
#define DMMCR0_BASEA 0x56
#define DMMCR1_BASEA 0x56
#define DMMCR2_BASEA 0x56
#define DMMCR3_BASEA 0x56
#define DMMCR4_BASEA 0x56
#define DMMCR5_BASEA 0x56

#define DMSRCP_BASEA 0x56
#define DMDSTP_BASEA 0x56

#define DMIDX0_BASEA 0x56
#define DMIDX1_BASEA 0x56

#define DMFRI0_BASEA 0x56
#define DMFRI1_BASEA 0x56

#define DMSGA_BASEA  0x56
#define DMGDA_BASEA  0x56
#define DMGCR_BASEA  0x56
#define DMGFR_BASEA  0x56

#define DMSRC0_ADDRA (*(volatile unsigned int *) DMSRC0_BASEA)
#define DMSRC1_ADDRA (*(volatile unsigned int *) DMSRC1_BASEA)
#define DMSRC2_ADDRA (*(volatile unsigned int *) DMSRC2_BASEA)
#define DMSRC3_ADDRA (*(volatile unsigned int *) DMSRC3_BASEA)
#define DMSRC4_ADDRA (*(volatile unsigned int *) DMSRC4_BASEA)
#define DMSRC5_ADDRA (*(volatile unsigned int *) DMSRC5_BASEA)

#define DMDST0_ADDRA (*(volatile unsigned int *) DMDST0_BASEA)
#define DMDST1_ADDRA (*(volatile unsigned int *) DMDST1_BASEA)
#define DMDST2_ADDRA (*(volatile unsigned int *) DMDST2_BASEA)
#define DMDST3_ADDRA (*(volatile unsigned int *) DMDST3_BASEA)
#define DMDST4_ADDRA (*(volatile unsigned int *) DMDST4_BASEA)
#define DMDST5_ADDRA (*(volatile unsigned int *) DMDST5_BASEA)

#define DMCTR0_ADDRA (*(volatile unsigned int *) DMCTR0_BASEA)
#define DMCTR1_ADDRA (*(volatile unsigned int *) DMCTR0_BASEA)
#define DMCTR2_ADDRA (*(volatile unsigned int *) DMCTR0_BASEA)
#define DMCTR3_ADDRA (*(volatile unsigned int *) DMCTR0_BASEA)
#define DMCTR4_ADDRA (*(volatile unsigned int *) DMCTR0_BASEA)
#define DMCTR5_ADDRA (*(volatile unsigned int *) DMCTR0_BASEA)

#define DMSFC0_ADDRA ((volatile DMSFCn_REG *) ((char *) DMSFC0_BASEA))
#define DMSFC1_ADDRA ((volatile DMSFCn_REG *) ((char *) DMSFC1_BASEA))
#define DMSFC2_ADDRA ((volatile DMSFCn_REG *) ((char *) DMSFC2_BASEA))
#define DMSFC3_ADDRA ((volatile DMSFCn_REG *) ((char *) DMSFC3_BASEA))
#define DMSFC4_ADDRA ((volatile DMSFCn_REG *) ((char *) DMSFC4_BASEA))
#define DMSFC5_ADDRA ((volatile DMSFCn_REG *) ((char *) DMSFC5_BASEA))

#define DMMCR0_ADDRA ((volatile DMMCRn_REG *) ((char *) DMMCR0_BASEA))
#define DMMCR1_ADDRA ((volatile DMMCRn_REG *) ((char *) DMMCR1_BASEA))
#define DMMCR2_ADDRA ((volatile DMMCRn_REG *) ((char *) DMMCR2_BASEA))
#define DMMCR3_ADDRA ((volatile DMMCRn_REG *) ((char *) DMMCR3_BASEA))
#define DMMCR4_ADDRA ((volatile DMMCRn_REG *) ((char *) DMMCR4_BASEA))
#define DMMCR5_ADDRA ((volatile DMMCRn_REG *) ((char *) DMMCR5_BASEA))
```

```
#define DMSRCP_ADDRA ((volatile DMSRCP_REG *) ((char *) DMSRCP_BASEA))
#define DMDSTP_ADDRA ((volatile DMDSTP_REG *) ((char *) DMDSTP_BASEA))

#define DMIDX0_ADDRA (*(volatile unsigned int *) DMIDX0_BASEA)
#define DMIDX1_ADDRA (*(volatile unsigned int *) DMIDX1_BASEA)

#define DMFRI0_ADDRA (*(volatile unsigned int *) DMFRI0_BASEA)
#define DMFRI1_ADDRA (*(volatile unsigned int *) DMFRI1_BASEA)

#define DMSGA_ADDRA  (*(volatile unsigned int *) DMSGA_BASEA)
#define DMGDA_ADDRA  (*(volatile unsigned int *) DMGDA_BASEA)
#define DMGCR_ADDRA  (*(volatile unsigned int *) DMGCR_BASEA)
#define DMGFR_ADDRA  (*(volatile unsigned int *) DMGFR_BASEA)

#if ASMDEFS
ASM_SET(DMSRC0_BASEA);
ASM_SET(DMSRC1_BASEA);
ASM_SET(DMSRC2_BASEA);
ASM_SET(DMSRC3_BASEA);
ASM_SET(DMSRC4_BASEA);
ASM_SET(DMSRC5_BASEA);

ASM_SET(DMDST0_BASEA);
ASM_SET(DMDST1_BASEA);
ASM_SET(DMDST2_BASEA);
ASM_SET(DMDST3_BASEA);
ASM_SET(DMDST4_BASEA);
ASM_SET(DMDST5_BASEA);

ASM_SET(DMCTR0_BASEA);
ASM_SET(DMCTR1_BASEA);
ASM_SET(DMCTR2_BASEA);
ASM_SET(DMCTR3_BASEA);
ASM_SET(DMCTR4_BASEA);
ASM_SET(DMCTR5_BASEA);

ASM_SET(DMSFC0_BASEA);
ASM_SET(DMSFC1_BASEA);
ASM_SET(DMSFC2_BASEA);
ASM_SET(DMSFC3_BASEA);
ASM_SET(DMSFC4_BASEA);
ASM_SET(DMSFC5_BASEA);

ASM_SET(DMMCR0_BASEA);
ASM_SET(DMMCR1_BASEA);
ASM_SET(DMMCR2_BASEA);
ASM_SET(DMMCR3_BASEA);
ASM_SET(DMMCR4_BASEA);
ASM_SET(DMMCR5_BASEA);

ASM_SET(DMSRCP_BASEA);
ASM_SET(DMDSTP_BASEA);
ASM_SET(DMIDX0_BASEA);
ASM_SET(DMIDX1_BASEA);

ASM_SET(DMFRI0_BASEA);
ASM_SET(DMFRI1_BASEA);

ASM_SET(DMSGA_BASEA);
ASM_SET(DMGDA_BASEA);
ASM_SET(DMGCR_BASEA);
ASM_SET(DMGFR_BASEA);
#endif /* ASMDEFS */

/* Define the base addresses without auto–incrementing  */
#define DMSRC0_BASE  0x57
#define DMSRC1_BASE  0x57
#define DMSRC2_BASE  0x57
#define DMSRC3_BASE  0x57
#define DMSRC4_BASE  0x57
#define DMSRC5_BASE  0x57

#define DMDST0_BASE  0x57
#define DMDST1_BASE  0x57
#define DMDST2_BASE  0x57
#define DMDST3_BASE  0x57
#define DMDST4_BASE  0x57
#define DMDST5_BASE  0x57

#define DMCTR0_BASE  0x57
#define DMCTR1_BASE  0x57
#define DMCTR2_BASE  0x57
#define DMCTR3_BASE  0x57
#define DMCTR4_BASE  0x57
#define DMCTR5_BASE  0x57

#define DMSFC0_BASE  0x57
#define DMSFC1_BASE  0x57
#define DMSFC2_BASE  0x57
#define DMSFC3_BASE  0x57
#define DMSFC4_BASE  0x57
#define DMSFC5_BASE  0x57

#define DMMCR0_BASE  0x57
#define DMMCR1_BASE  0x57
#define DMMCR2_BASE  0x57
#define DMMCR3_BASE  0x57
#define DMMCR4_BASE  0x57
#define DMMCR5_BASE  0x57

#define DMSRCP_BASE  0x57
#define DMDSTP_BASE  0x57

#define DMIDX0_BASE  0x57
#define DMIDX1_BASE  0x57
```

```
#define DMFRI0_BASE  0x57
#define DMFRI1_BASE  0x57

#define DMSGA_BASE   0x57
#define DMGDA_BASE   0x57
#define DMGCR_BASE   0x57
#define DMGFR_BASE   0x57

#define DMSRC0_ADDR  (*(volatile unsigned int *) DMSRC0_BASE)
#define DMSRC1_ADDR  (*(volatile unsigned int *) DMSRC1_BASE)
#define DMSRC2_ADDR  (*(volatile unsigned int *) DMSRC2_BASE)
#define DMSRC3_ADDR  (*(volatile unsigned int *) DMSRC3_BASE)
#define DMSRC4_ADDR  (*(volatile unsigned int *) DMSRC4_BASE)
#define DMSRC5_ADDR  (*(volatile unsigned int *) DMSRC5_BASE)

#define DMDST0_ADDR  (*(volatile unsigned int *) DMDST0_BASE)
#define DMDST1_ADDR  (*(volatile unsigned int *) DMDST1_BASE)
#define DMDST2_ADDR  (*(volatile unsigned int *) DMDST2_BASE)
#define DMDST3_ADDR  (*(volatile unsigned int *) DMDST3_BASE)
#define DMDST4_ADDR  (*(volatile unsigned int *) DMDST4_BASE)
#define DMDST5_ADDR  (*(volatile unsigned int *) DMDST5_BASE)

#define DMCTR0_ADDR  (*(volatile unsigned int *) DMCTR0_BASE)
#define DMCTR1_ADDR  (*(volatile unsigned int *) DMCTR1_BASE)
#define DMCTR2_ADDR  (*(volatile unsigned int *) DMCTR2_BASE)
#define DMCTR3_ADDR  (*(volatile unsigned int *) DMCTR3_BASE)
#define DMCTR4_ADDR  (*(volatile unsigned int *) DMCTR4_BASE)
#define DMCTR5_ADDR  (*(volatile unsigned int *) DMCTR5_BASE)

#define DMSFC0_ADDR  ((volatile DMSFCn_REG *) ((char *) DMSFC0_BASE))
#define DMSFC1_ADDR  ((volatile DMSFCn_REG *) ((char *) DMSFC1_BASE))
#define DMSFC2_ADDR  ((volatile DMSFCn_REG *) ((char *) DMSFC2_BASE))
#define DMSFC3_ADDR  ((volatile DMSFCn_REG *) ((char *) DMSFC3_BASE))
#define DMSFC4_ADDR  ((volatile DMSFCn_REG *) ((char *) DMSFC4_BASE))
#define DMSFC5_ADDR  ((volatile DMSFCn_REG *) ((char *) DMSFC5_BASE))

#define DMMCR0_ADDR  ((volatile DMMCRn_REG *) ((char *) DMMCR0_BASE))
#define DMMCR1_ADDR  ((volatile DMMCRn_REG *) ((char *) DMMCR1_BASE))
#define DMMCR2_ADDR  ((volatile DMMCRn_REG *) ((char *) DMMCR2_BASE))
#define DMMCR3_ADDR  ((volatile DMMCRn_REG *) ((char *) DMMCR3_BASE))
#define DMMCR4_ADDR  ((volatile DMMCRn_REG *) ((char *) DMMCR4_BASE))
#define DMMCR5_ADDR  ((volatile DMMCRn_REG *) ((char *) DMMCR5_BASE))

#define DMSRCP_ADDR  ((volatile DMSRCP_REG *) ((char *) DMSRCP_BASE))
#define DMDSTP_ADDR  ((volatile DMDSTP_REG *) ((char *) DMDSTP_BASE))

#define DMIDX0_ADDR  (*(volatile unsigned int *) DMIDX0_BASE)
#define DMIDX1_ADDR  (*(volatile unsigned int *) DMIDX1_BASE)

#define DMFRI0_ADDR  (*(volatile unsigned int *) DMFRI0_BASE)
#define DMFRI1_ADDR  (*(volatile unsigned int *) DMFRI1_BASE)
```

```
#define DMSGA_ADDR   (*(volatile unsigned int *) DMSGA_BASE)
#define DMGDA_ADDR   (*(volatile unsigned int *) DMGDA_BASE)
#define DMGCR_ADDR   (*(volatile unsigned int *) DMGCR_BASE)
#define DMGFR_ADDR   (*(volatile unsigned int *) DMGFR_BASE)

#if ASMDEFS
ASM_SET(DMSRC0_BASE);
ASM_SET(DMSRC1_BASE);
ASM_SET(DMSRC2_BASE);
ASM_SET(DMSRC3_BASE);
ASM_SET(DMSRC4_BASE);
ASM_SET(DMSRC5_BASE);

ASM_SET(DMDST0_BASE);
ASM_SET(DMDST1_BASE);
ASM_SET(DMDST2_BASE);
ASM_SET(DMDST3_BASE);
ASM_SET(DMDST4_BASE);
ASM_SET(DMDST5_BASE);

ASM_SET(DMCTR0_BASE);
ASM_SET(DMCTR1_BASE);
ASM_SET(DMCTR2_BASE);
ASM_SET(DMCTR3_BASE);
ASM_SET(DMCTR4_BASE);
ASM_SET(DMCTR5_BASE);

ASM_SET(DMSFC0_BASE);
ASM_SET(DMSFC1_BASE);
ASM_SET(DMSFC2_BASE);
ASM_SET(DMSFC3_BASE);
ASM_SET(DMSFC4_BASE);
ASM_SET(DMSFC5_BASE);

ASM_SET(DMMCR0_BASE);
ASM_SET(DMMCR1_BASE);
ASM_SET(DMMCR2_BASE);
ASM_SET(DMMCR3_BASE);
ASM_SET(DMMCR4_BASE);
ASM_SET(DMMCR5_BASE);

ASM_SET(DMSRCP_BASE);
ASM_SET(DMDSTP_BASE);

ASM_SET(DMIDX0_BASE);
ASM_SET(DMIDX1_BASE);

ASM_SET(DMFRI0_BASE);
ASM_SET(DMFRI1_BASE);

ASM_SET(DMSGA_BASE);
ASM_SET(DMGDA_BASE);
ASM_SET(DMGCR_BASE);
ASM_SET(DMGFR_BASE);
#endif /* ASMDEFS */
```

```
/*-------------------------------------------------------------------*/
/* DMPREC                                                            */
/*-------------------------------------------------------------------*/
typedef union {
                     struct {
                        unsigned int free    :1;
                        unsigned int rsvd    :1;
                        unsigned int dprc    :6;
                        unsigned int intosel :2;
                        unsigned int de      :6;
                        } bitval;
                     unsigned int value;
} DMPREC_REG;

/*-------------------------------------------------------------------*/
/* DMFCn                                                             */
/*-------------------------------------------------------------------*/
typedef union {
                     struct {
                        unsigned int dsyn       :4;
                        unsigned int dblw       :1;
                        unsigned int rsrvd      :3;
                        unsigned int framecount :8;
                        } bitval;
                     unsigned int value;
} DMSFCn_REG;

/*-------------------------------------------------------------------*/
/* DMMCRn                                                            */
/*-------------------------------------------------------------------*/
typedef union {
                     struct {
                        unsigned int autoinit:1;
                        unsigned int dinm    :1;
                        unsigned int imod    :1;
                        unsigned int ctmod   :1;
                        unsigned int rsrvd1  :1;
                        unsigned int sind    :2;
                        unsigned int dms     :2;
                        unsigned int rsrvd2  :1;
                        unsigned int dind    :3;
                        unsigned int dmd     :2;
                        } bitval;
                     unsigned int value;
} DMMCRn_REG;
```

```
/*-------------------------------------------------------------------*/
/* DMSRCP                                                            */
/*-------------------------------------------------------------------*/
typedef union {
                     struct {
                        unsigned int rsvd    :9;
                        unsigned int source  :7;
                        } bitval;
                     unsigned int value;
} DMSRCP_REG;

/*-------------------------------------------------------------------*/
/* DMDSTP                                                            */
/*-------------------------------------------------------------------*/
typedef union {
                     struct {
                        unsigned int rsvd :9;
                        unsigned int dest :7;
                        } bitval;
                     unsigned int value;
} DMDSTP_REG;

#elif C549
/*********************************************************************/
/* Structure for BSP                                                 */
/*********************************************************************/
#define BDRR0_BASE   0x20 /* BSP 0 Data Receive Register */
#define BDRR0_ADDR   (*(volatile unsigned int *)BDDR0_BASE)

#define BDXR0_BASE   0x21 /* BSP 0 data–transmit register */
#define BDXR0_ADDR   (*(volatile unsigned int *)BDXR0_BASE)

#define BSPC0_BASE   0x22 /* BSP 0 control register */
#define BSPC0_ADDR   ((volatile BSPC_REG *)   ((char *) BSPC0_BASE))

#define BSPCE0_BASE  0x23 /* BSP 0 control extension register */
#define BSPCE0_ADDR  ((volatile BSPCE_REG *)  ((char *) BSPCE0_BASE))

#if ASMDEFS
ASM_SET(BDRR0_BASE);
ASM_SET(BDXR0_BASE);
ASM_SET(BSPC0_BASE);
ASM_SET(BSPCE0_BASE);
#endif /* ASMDEFS */

#define TRCV_BASE    0x30 /* TDM port data–receive register  */
#define TRCV_ADDR    (*(volatile unsigned int *)TRCV_BASE)

#define TDXR_BASE    0x31 /* TDM port data–transmit register */
#define TDXR_ADDR    (*(volatile unsigned int *)TDXR_BASE)

#define TSPC_BASE    0x32 /* TDM serial port control register */
#define TSPC_ADDR    ((volatile TSPC_REG *)   ((char *) TSPC_BASE))
```

```
#define TCSR_BASE    0x33 /* TDM channel–select register */
#define TCSR_ADDR    ((volatile TCSR_REG *)   ((char *) TCSR_BASE))

#define TRTA_BASE    0x34 /* TDM receive/ transmit register */
#define TRTA_ADDR    ((volatile TRTA_REG *)   ((char *) TRTA_BASE))

#define TRAD_BASE    0x35 /* TDM receive/ address register */
#define TRAD_ADDR    ((volatile BSPCE_REG *)  ((char *) _BASE))

#if ASMDEFS
ASM_SET(TRCV_BASE);
ASM_SET(TDXR_BASE);
ASM_SET(TSPC_BASE);
ASM_SET(TCSR_BASE);
ASM_SET(TRTA_BASE);
ASM_SET(TRAD_BASE);
#endif /* ASMDEFS */

#define AXR0_BASE    0x38 /* ABU 0 transmit–address register */
#define AXR0_ADDR    (*(volatile unsigned int *)AXR0_BASE)

#define BKX0_BASE    0x39 /* ABU 0 transmit–buffer–size register */
#define BKX0_ADDR    (*(volatile unsigned int *)BKX0_BASE)

#define ARR0_BASE    0x3A /* ABU 0 receive–address register */
#define ARR0_ADDR    (*(volatile unsigned int *)ARR0_BASE)

#define BKR0_BASE    0x3B /* ABU 0 receive–buffer–size register */
#define BKR0_ADDR    (*(volatile unsigned int *)BKR0_BASE)

#define AXR1_BASE    0x3C /* ABU 1 transmit–address register  */
#define AXR1_ADDR    (*(volatile unsigned int *)AXR1_BASE)

#define BKX1_BASE    0x3D /* ABU 1 transmit–buffer–size register */
#define BKX1_ADDR    (*(volatile unsigned int *)BKX1_BASE)

#define ARR1_BASE    0x3E /* ABU 1 receive–address register */
#define ARR1_ADDR    (*(volatile unsigned int *)ARR1_BASE)

#define BKR1_BASE    0x3F /* ABU 1 receive–buffer–size register */
#define BKR1_ADDR    (*(volatile unsigned int *)BKR1_BASE)

#if ASMDEFS
ASM_SET(AXR0_BASE);
ASM_SET(BKX0_BASE);
ASM_SET(ARR0_BASE);
ASM_SET(BKR0_BASE);
ASM_SET(AXR1_BASE);
ASM_SET(BKX1_BASE);
ASM_SET(ARR1_BASE);
ASM_SET(BKR1_BASE);
#endif /* ASMDEFS */
```

```
#define BDRR1_BASE   0x40 /* BSP 1 data–receive register */
#define BDRR1_ADDR   (*(volatile unsigned int *)BDDR1_BASE)

#define BDXR1_BASE   0x41 /* BSP 1 data–transmit register */
#define BDXR1_ADDR   (*(volatile unsigned int *)BDXR1_BASE)

#define BSPC1_BASE   0x42 /* BSP 1 control register */
#define BSPC1_ADDR   ((volatile BSPC_REG *)  ((char *) BSPC1_BASE))

#define BSPCE1_BASE  0x43 /* BSP 1 control extension register */
#define BSPCE1_ADDR  ((volatile BSPCE_REG *)  ((char *) BSPCE1_BASE))

#if ASMDEFS
ASM_SET(BDRR1_BASE);
ASM_SET(BDXR1_BASE);
ASM_SET(BSPC1_BASE);
ASM_SET(BSPCE1_BASE);
#endif /* ASMDEFS */

typedef union {
                     struct {
                        unsigned int free    :1;
                        unsigned int soft    :1;
                        unsigned int rsrfull :1;
                        unsigned int xsrempty:1;
                        unsigned int xrdy    :1;
                        unsigned int rrdy    :1;
                        unsigned int in1     :1;
                        unsigned int in0     :1;
                        unsigned int rrst    :1;
                        unsigned int xrst    :1;
                        unsigned int txm     :1;
                        unsigned int mcm     :1;
                        unsigned int fsm     :1;
                        unsigned int fo      :1;
                        unsigned int dlb     :1;
                        unsigned int res     :1;
                        } bitval;
                     unsigned int value;
} BSPC_REG;

typedef union {
                     struct {
                        unsigned int haltr   :1;
                        unsigned int rh      :1;
                        unsigned int bre     :1;
                        unsigned int haltx   :1;
                        unsigned int xh      :1;
                        unsigned int bxe     :1;
                        unsigned int pcm     :1;
                        unsigned int fig     :1;
                        unsigned int fe      :1;
                        unsigned int clkp    :1;
                        unsigned int fsp     :1;
                        unsigned int clkdv   :5;
                        } bitval;
                     unsigned int value;
} BSPCE_REG;

typedef union {
                     struct {
                        unsigned int free :1;
                        unsigned int soft :1;
                        unsigned int res2 :2;
                        unsigned int xrdy :1;
                        unsigned int rrdy :1;
                        unsigned int in1  :1;
                        unsigned int in0  :1;
                        unsigned int rrst :1;
                        unsigned int xrst :1;
                        unsigned int txm  :1;
                        unsigned int mcm  :1;
                        unsigned int res1 :1;
                        unsigned int zero :2;
                        unsigned int tdm  :1;
                        } bitval;
                     unsigned int value;
} TSPC_REG;

typedef union {
                     struct {
                        unsigned int res  :8;
                        unsigned int ch7  :1;
                        unsigned int ch6  :1;
                        unsigned int ch5  :1;
                        unsigned int ch4  :1;
                        unsigned int ch3  :1;
                        unsigned int ch2  :1;
                        unsigned int ch1  :1;
                        unsigned int ch0  :1;
                        } bitval;
                     unsigned int value;
} TCSR_REG;

typedef union {
                     struct {
                        unsigned int ta7 :1;
                        unsigned int ta6 :1;
                        unsigned int ta5 :1;
                        unsigned int ta4 :1;
                        unsigned int ta3 :1;
                        unsigned int ta2 :1;
                        unsigned int ta1 :1;
                        unsigned int ta0 :1;
                        unsigned int ra7 :1;
                        unsigned int ra6 :1;
                        unsigned int ra5 :1;
                        unsigned int ra4 :1;
                        unsigned int ra3 :1;
                        unsigned int ra2 :1;
                        unsigned int ra1 :1;
                        unsigned int ra0 :1;
                        } bitval;
                     unsigned int value;
} TRTA_REG;

typedef union {
                     struct {
                        unsigned int res :2;
                        unsigned int x2  :1;
                        unsigned int x1  :1;
                        unsigned int x0  :1;
                        unsigned int s2  :1;
                        unsigned int s1  :1;
                        unsigned int s0  :1;
                        unsigned int a7  :1;
                        unsigned int a6  :1;
                        unsigned int a5  :1;
                        unsigned int a4  :1;
                        unsigned int a3  :1;
                        unsigned int a2  :1;
                        unsigned int a1  :1;
                        unsigned int a0  :1;
                        } bitval;
                     unsigned int value;
} TRAD_REG;

#endif /* McBSP and DMA endif */

/*******************************************************************/
/* TIMER REGISTERS ADDRESSES                                       */
/*******************************************************************/
#define TIM_BASE     0x24
#define TIM_ADDR     *(volatile unsigned int *)0x24

#define PRD_BASE     0x25
#define PRD_ADDR     *(volatile unsigned int *)0x25

#define TCR_BASE     0x26
#define TCR_ADDR     ((volatile TCR_REG *)((char *) TCR_BASE))

#if ASMDEFS
ASM_SET(TIM_BASE);
ASM_SET(PRD_BASE);
ASM_SET(TCR_BASE);
#endif /* ASMDEFS */
```

```
#if C5402
#define TIM1_BASE    0x30
#define TIM1_ADDR    *(volatile unsigned int *)0x30

#define PRD1_BASE    0x31
#define PRD1_ADDR    *(volatile unsigned int *)0x31

#define TCR1_BASE    0x32
#define TCR1_ADDR    ((volatile TCR_REG *)((char *) TCR1_BASE))

#if ASMDEFS
ASM_SET(TIM1_BASE);
ASM_SET(PRD1_BASE);
ASM_SET(TCR1_BASE);
#endif /* ASMDEFS */

#endif /* ’5402 endif */

typedef union {
                     struct {
                        unsigned int rsrvd   :4;
                        unsigned int soft    :1;
                        unsigned int free    :1;
                        unsigned int psc     :4;
                        unsigned int trb     :1;
                        unsigned int tss     :1;
                        unsigned int tddr    :4;
                        } bitval;
                     unsigned int value;
} TCR_REG;

/*********************************************************************/
/* CLOCK MODE REGISTER ADDRESS                                       */
/*********************************************************************/
#define CLKMD_BASE   0x58
#define CLKMD_ADDR   ((volatile CLKMD_REG *) ((char *) CLKMD_BASE))

#if ASMDEFS
ASM_SET(CLKMD_BASE);
#endif /* ASMDEFS */

typedef union {
                     struct {
                        unsigned int pllmul      :4;
                        unsigned int plldiv      :1;
                        unsigned int pllcount    :8;
                        unsigned int pllon_off   :1;
                        unsigned int pllndiv     :1;
                        unsigned int pllstatus   :1;
                        } bitval;
                     unsigned int value;
} CLKMD_REG;
```

```
#if C5402
/******************************************************************/
/* GPIO                                                           */
/******************************************************************/
#define GPIOCR_BASE  0x3c
#define GPIOCR_ADDR  ((volatile GPIOCR_REG *) ((char *) GPIOCR_BASE))

#define GPIOSR_BASE  0x3d
#define GPIOSR_ADDR  ((volatile GPIOSR_REG *) ((char *) GPIOSR_BASE))

#if ASMDEFS
ASM_SET(GPIOCR_BASE);
ASM_SET(GPIOSR_BASE);
#endif /* ASMDEFS */

typedef union {
                     struct {
                        unsigned int tout1   :1;
                        unsigned int rsvd    :7;
                        unsigned int dir7    :1;
                        unsigned int dir6    :1;
                        unsigned int dir5    :1;
                        unsigned int dir4    :1;
                        unsigned int dir3    :1;
                        unsigned int dir2    :1;
                        unsigned int dir1    :1;
                        unsigned int dir0    :1;
                        } bitval;
                     unsigned int value;
} GPIOCR_REG;

typedef union {
                     struct {
                        unsigned int rsvd :8;
                        unsigned int io7  :1;
                        unsigned int io6  :1;
                        unsigned int io5  :1;
                        unsigned int io4  :1;
                        unsigned int io3  :1;
                        unsigned int io2  :1;
                        unsigned int io1  :1;
                        unsigned int io0  :1;
                        } bitval;
                     unsigned int value;
} GPIOSR_REG;
#endif /* ’5402 endif */

#define __MMREGS
#endif
```

IMPORTANT NOTICE

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgment, including those pertaining to warranty, patent infringement, and limitation of liability.

TI warrants performance of its semiconductor products to the specifications applicable at the time of sale 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.

Customers are responsible for their applications using TI components.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance or customer product design. TI does not 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. TI's publication of information regarding any third party's products or services does not constitute TI's approval, warranty or endorsement thereof.