Chapter 2

CPU Registers

This chapter describes the main registers in the C55x™ DSP CPU. Section 2.1 lists the registers in alphabetical order, and section 2.2 shows the addresses for the memory-mapped registers. The other sections contain additional details about the CPU registers.

Topic **Page**

2.1 Alphabetical Summary of Registers

Table 2−1 lists the registers in alphabetical order. For more details about a particular register, see the page given in the last column of the table.

Table 2−1. Alphabetical Summary of Registers

Register Name	Description	Size	See ...
AC0–AC3	Accumulators 0 through 3	40 bits each	Page 2-9
AR0–AR7	Auxiliary registers 0 through 7	16 bits each	Page 2-12
BK03, BK47, BKC	Circular buffer size registers	16 bits each	Page 2-16
BRC0, BRC1	Block-repeat counters 0 and 1	16 bits each	Page 2-35
BRS1	BRC1 save register	16 bits	Page 2-35
BSA01, BSA23, BSA45, BSA67, BSAC	Circular buffer start address registers	16 bits each	Page 2-15
CDP	Coefficient data pointer (low part of XCDP)	16 bits	Page 2-14
CDPH	High part of XCDP	7 bits	Page 2-14
CFCT	Control-flow context register	8 bits	Page 2-22
CSR	Computed single-repeat register	16 bits	Page 2-35
DBIER0, DBIER1	Debug interrupt enable registers 0 and 1	16 bits each	Page 2-31
DP	Data page register (low part of XDP)	16 bits	Page 2-17
DPH	High part of XDP	7 bits	Page 2-17
IER0, IER1	Interrupt enable registers 0 and 1	16 bits each	Page 2-28
IFR0, IFR1	Interrupt flag registers 0 and 1	16 bits each	Page 2-25
IVPD, IVPH	Interrupt vector pointers	16 bits each	Page 2-24
PC	Program counter	24 bits	Page 2-22
PDP	Peripheral data page register	9 bits	Page 2-18
REA0, REA1	Block-repeat end address registers 0 and 1	24 bits each	Page 2-35
RETA	Return address register	24 bits	Page 2-22
RPTC	Single-repeat counter	16 bits	Page 2-35
RSA0, RSA1	Block-repeat start address registers 0 and 1	24 bits each	Page 2-35

Table 2−1. Alphabetical Summary of Registers (Continued)

Register Name	Description	Size	See ...
SP	Data stack pointer (low part of XSP)	16 bits	Page 2-18
SPH	High part of XSP and XSSP	7 bits	Page 2-18
SSP	System stack pointer (low part of XSSP)	16 bits	Page 2-18
ST0_55–ST3_55	Status registers 0 through 3	16 bits each	Page 2-38
T0–T3	Temporary registers	16 bits each	Page 2-11
TRN0, TRN1	Transition registers 0 and 1	16 bits each	Page 2-10
XAR0–XAR7	Extended auxiliary registers 0 through 7	23 bits each	Page 2-12
XCDP	Extended coefficient data pointer	23 bits	Page 2-14
XDP	Extended data page register	23 bits	Page 2-17
XSP	Extended data stack pointer	23 bits	Page 2-18
XSSP	Extended system stack pointer	23 bits	Page 2-18

2.2 Memory-Mapped Registers

Table 2−2 shows the memory-mapped registers, which are CPU registers mapped to addresses in the data space of the DSP.

Notes:

1) ST0_55, ST1_55, and ST3_55 are each accessible at two addresses. At one address, all the TMS320C55x DSP bits are available. At the other address (the protected address), certain bits cannot be modified. The protected address is provided to support TMS320C54x DSP code that writes to ST0, ST1, and PMST (the C54x DSP counterpart of ST3_55).
2) T3, RSA0L, REA0L, and SP are each accessible at two addresses. For accesses using the DP direct addressing mode memory-mapped register accesses, the assembler substitutes the higher of the two addresses: T3 = 23h (not 0Eh), RSA0L = 3Dh (not 1Bh), REA0L = 3Fh (not 1Ch), SP = 4Dh (not 18h).
3) Any C55x DSP instruction that loads BRC1 loads the same value to BRS1.

Table 2−2. Memory-Mapped Registers

Address(es)	Register	Description	Bit Range	See ...
00 0000h	IER0	Interrupt enable register 0	15–0	Page 2-28
00 0001h	IFR0	Interrupt flag register 0	15–0	Page 2-25
00 0002h (for C55x DSP code)	ST0_55	Status register 0	15–0	Page 2-38
Note: Address 00 0002h is for native TMS320C55x DSP code that accesses ST0_55. TMS320C54x DSP code that was written to access ST0 must use address 00 0006h to access ST0_55.				
00 0003h (for C55x DSP code)	ST1_55	Status register 1	15–0	Page 2-38
Note: Address 00 0003h is for native TMS320C55x DSP code that accesses ST1_55. TMS320C54x DSP code that was written to access ST1 must use address 00 0007h to access ST1_55.				

Table 2−2. Memory-Mapped Registers (Continued)

Address(es)	Register	Description	Bit Range	See ...
00 0004h (for C55x DSP code)	ST3_55	Status register 3	15–0	Page 2-38
Note: Address 00 0004h is for native TMS320C55x DSP code that accesses ST3_55. TMS320C54x DSP code that was written to access the processor mode status register (PMST) must use address 00 001Dh to access ST3_55.				
00 0005h	–	Reserved (do not use this address)	–	–
00 0006h (for C54x DSP code)	ST0 (ST0_55)	Status register 0	15–0	Page 2-38
Note: Address 00 0006h is the protected address of ST0_55. This address is for TMS320C54x DSP code that was written to access ST0. Native TMS320C55x DSP code must use address 00 0002h to access ST0_55.				
00 0007h (for C54x DSP code)	ST1 (ST1_55)	Status register 1	15–0	Page 2-38
Note: Address 00 0007h is the protected address of ST1_55. This address is for TMS320C54x DSP code that was written to access ST1. Native TMS320C55x DSP code must use address 00 0003h to access ST1_55.				
00 0008h	AC0L	Accumulator 0	15–0	Page 2-9
00 0009h	AC0H		31–16	
00 000Ah	AC0G		39–32	
00 000Bh	AC1L	Accumulator 1	15–0	Page 2-9
00 000Ch	AC1H		31–16	
00 000Dh	AC1G		39–32	
00 000Eh	T3	Temporary register 3	15–0	Page 2-11
00 000Fh	TRN0	Transition register 0	15–0	Page 2-10
00 0010h	AR0	Auxiliary register 0	15–0	Page 2-12
00 0011h	AR1	Auxiliary register 1	15–0	Page 2-12
00 0012h	AR2	Auxiliary register 2	15–0	Page 2-12
00 0013h	AR3	Auxiliary register 3	15–0	Page 2-12
00 0014h	AR4	Auxiliary register 4	15–0	Page 2-12
00 0015h	AR5	Auxiliary register 5	15–0	Page 2-12

Table 2−2. Memory-Mapped Registers (Continued)

Address(es)	Register	Description	Bit Range	See ...
00 0016h	AR6	Auxiliary register 6	15–0	Page 2-12
00 0017h	AR7	Auxiliary register 7	15–0	Page 2-12
00 0018h	SP	Data stack pointer	15–0	Page 2-18
00 0019h	BK03	Circular buffer size register for AR0–AR3	15–0	Page 2-16
Note: In the TMS320C54x-DSP compatible mode (C54CM = 1), BK03 is used for all the auxiliary registers. C54CM is a bit in status register 1 (ST1_55). The status registers are described beginning on page 2-38.				
00 001Ah	BRC0	Block-repeat counter 0	15–0	Page 2-35
00 001Bh	RSA0L	Low part of block-repeat start address register 0	15–0	Page 2-35
00 001Ch	REA0L	Low part of block-repeat end address register 0	15–0	Page 2-35
00 001Dh (for C54x DSP code)	PMST (ST3_55)	Status register 3	15–0	Page 2-38
Note: Address 00 001Dh is the protected address of ST3_55. This address is for TMS320C54x DSP code that was written to access the processor mode status register (PMST). Native TMS320C55x DSP code must use address 00 0004h to access ST3_55.				
00 001Eh	–	Reserved (do not use this address)	–	–
00 001Fh	–	Reserved (do not use this address)	–	–
00 0020h	T0	Temporary register 0	15–0	Page 2-11
00 0021h	T1	Temporary register 1	15–0	Page 2-11
00 0022h	T2	Temporary register 2	15–0	Page 2-11
00 0023h	T3	Temporary register 3	15–0	Page 2-11
00 0024h	AC2L	Accumulator 2	15–0	Page 2-9
00 0025h	AC2H		31–16	
00 0026h	AC2G		39–32	
00 0027h	CDP	Coefficient data pointer	15–0	Page 2-14
00 0028h	AC3L	Accumulator 3	15–0	Page 2-9
00 0029h	AC3H		31–16	
00 002Ah	AC3G		39–32	

Table 2−2. Memory-Mapped Registers (Continued)

Address(es)	Register	Description	Bit Range	See ...
00 002Bh	DPH	High part of the extended data page register	6–0	Page 2-17
00 002Ch	–	Reserved (do not use these addresses)	–	–
00 002Dh	–		–	
00 002Eh	DP	Data page register	15–0	Page 2-17
00 002Fh	PDP	Peripheral data page register	8–0	Page 2-18
00 0030h	BK47	Circular buffer size register for AR4–AR7	15–0	Page 2-16
00 0031h	BKC	Circular buffer size register for CDP	15–0	Page 2-16
00 0032h	BSA01	Circular buffer start address register for AR0 and AR1	15–0	Page 2-15
00 0033h	BSA23	Circular buffer start address register for AR2 and AR3	15–0	Page 2-15
00 0034h	BSA45	Circular buffer start address register for AR4 and AR5	15–0	Page 2-15
00 0035h	BSA67	Circular buffer start address register for AR6 and AR7	15–0	Page 2-15
00 0036h	BSAC	Circular buffer start address register for CDP	15–0	Page 2-15
00 0037h	–	Reserved for BIOS. This location contains a 16-bit register that is used as a start-up storage location for the data table pointer necessary for BIOS operation.	–	–
00 0038h	TRN1	Transition register 1	15–0	Page 2-10
00 0039h	BRC1	Block-repeat counter 1	15–0	Page 2-35
00 003Ah	BRS1	BRC1 save register	15–0	Page 2-35
00 003Bh	CSR	Computed single-repeat register	15–0	Page 2-35
00 003Ch	RSA0H	Block-repeat start address register 0	23–16	Page 2-35
00 003Dh	RSA0L		15–0	

Table 2−2. Memory-Mapped Registers (Continued)

Address(es)	Register	Description	Bit Range	See ...
00 003Eh	REA0H	Block-repeat end address register 0	23–16	Page 2-35
00 003Fh	REA0L		15–0	
00 0040h	RSA1H	Block-repeat start address register 1	23–16	Page 2-35
00 0041h	RSA1L		15–0	
00 0042h	REA1H	Block-repeat end address register 1	23–16	Page 2-35
00 0043h	REA1L		15–0	
00 0044h	RPTC	Single-repeat counter	15–0	Page 2-35
00 0045h	IER1	Interrupt enable register 1	10–0	Page 2-28
00 0046h	IFR1	Interrupt flag register 1	10–0	Page 2-25
00 0047h	DBIER0	Debug interrupt enable register 0	15–0	Page 2-31
00 0048h	DBIER1	Debug interrupt enable register 1	10–0	Page 2-31
00 0049h	IVPD	Interrupt vector pointer for vectors 0−15 and 24−31	15–0	Page 2-24
00 004Ah	IVPH	Interrupt vector pointer for vectors 16−23	15–0	Page 2-24
00 004Bh	ST2_55	Status register 2	15–0	Page 2-38
00 004Ch	SSP	System stack pointer	15–0	Page 2-18
00 004Dh	SP	Data stack pointer	15–0	Page 2-18
00 004Eh	SPH	High part of the extended stack pointers	6–0	Page 2-18
00 004Fh	CDPH	High part of the extended coefficient data pointer	6–0	Page 2-14
00 0050h to 00 005Fh	−	Reserved (do not use these addresses)	−	−

2.3 Accumulators (AC0−AC3)

The CPU contains four 40-bit accumulators: AC0, AC1, AC2, and AC3 (see Figure 2−1). The primary function of these registers is to assist in data computation in the following parts of the D unit: the arithmetic logic unit (ALU), the multiply-and-accumulate units (MACs), and the shifter. The four accumulators are basically equivalent; however, some instructions are restricted to certain accumulator pair groupings; for example:

```
SWAP   AC0, AC2  ; Valid instruction
SWAP   AC1, AC3  ; Valid instruction
```

but,

```
SWAP   AC0, AC1  ; Invalid instruction
```

Each accumulator is partitioned into a low word (ACxL), a high word (ACxH), and eight guard bits (ACxG). You can access each of these portions individually by using addressing modes that access the memory-mapped registers.

In the TMS320C54x DSP-compatible mode (C54CM = 1), accumulators AC0 and AC1 correspond to TMS320C54x DSP accumulators A and B, respectively.

Figure 2−1. Accumulators

	39−32	31−16	15−0
AC0	AC0G	AC0H	AC0L
AC1	AC1G	AC1H	AC1L
AC2	AC2G	AC2H	AC2L
AC3	AC3G	AC3H	AC3L

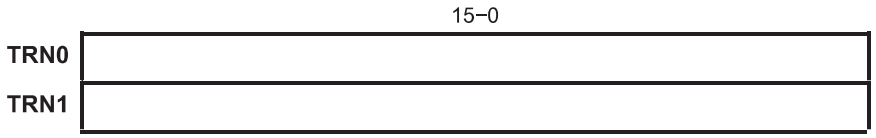
2.4 Transition Registers (TRN0, TRN1)

The two transition registers (see Figure 2−2) are used in the compare-and-select-extremum instructions:

- The syntaxes that perform two 16-bit extremum selections update TRN0 and TRN1 based on the comparison of two accumulators' high words and low words. TRN0 is updated based on the comparison of the accumulators' high words; TRN1 is updated based on the comparison of the low words.
- The syntaxes that perform a single 40-bit extremum selection update the selected transition register (TRN0 or TRN1) based on the comparison of two accumulators throughout their 40 bits.

TRN0 and TRN1 can hold transition decisions for the path to new metrics in Viterbi algorithm implementations.

Figure 2−2. Transition Registers



2.5 Temporary Registers (T0−T3)

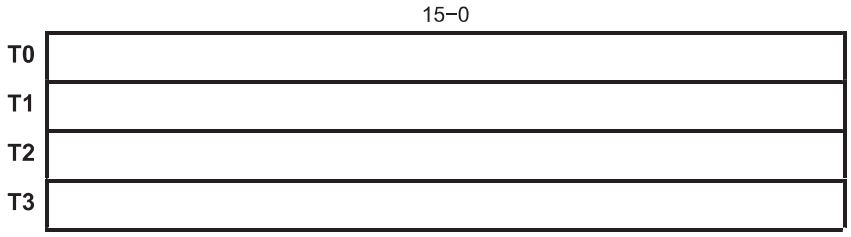
The CPU includes four 16-bit general-purpose temporary registers: T0–T3 (see Figure 2−3). Here are some of the things you can do with the temporary registers:

- Hold one of the memory multiplicands for multiply, multiply-and-accumulate, and multiply-and-subtract instructions
- Hold the shift count used in addition, subtraction, and load instructions performed in the D unit
- Keep track of more pointer values by swapping the contents of the auxiliary registers (AR0–AR7) and the temporary registers (using a swap instruction)
- Hold the transition metric of a Viterbi butterfly for dual 16-bit operations performed in the D-unit ALU

Note:

If C54CM = 1 (the TMS320C54x DSP-compatible mode is on), T2 is tied to the ASM bits of status register ST1_55 and cannot be used as a general-purpose register. For details, see the description for ASM on page 2-42.

Figure 2−3. Temporary Registers



2.6 Registers Used to Address Data Space and I/O Space

This section describes the following registers:

Register(s)	Function	See ...
XAR0−XAR7 and AR0−AR7	Point to a value in data space for accesses made with indirect addressing modes	Page 2-12
XCDP and CDP	Point to a value in data space for accesses made with indirect addressing modes	Page 2-14
BSA01, BSA23, BSA45, BSA67, BSAC	Specify a circular buffer start address to be added to a pointer	Page 2-15
BK03, BK47, BKC	Specify a circular buffer size	Page 2-16
XDP and DP	Specify the start address for accesses made with the DP direct addressing mode	Page 2-17
PDP	Identify the peripheral data page for an access to I/O space	Page 2-18
XSP and SP	Point to a value on the data stack	Page 2-18
XSSP and SSP	Point to a value on the system stack	Page 2-18

2.6.1 Auxiliary Registers (XAR0–XAR7 / AR0–AR7)

The CPU includes eight extended auxiliary registers XAR0–XAR7 (see Figure 2−4 and Table 2−3). Each high part (for example, AR0H) is used to specify the 7-bit main data page for accesses to data space. Each low part (for example, AR0) can be used as:

- A 16-bit offset to the 7-bit main data page (to form a 23-bit address)
- A bit address (in instructions that access individual bits or bit pairs)
- A general-purpose register or counter
- An index to select words relative to the start address of a circular buffer (see section 6.10, *Circular Addressing)*

Figure 2−4. Extended Auxiliary Registers and Their Parts

	22−16	15−0
XAR0	AR0H	AR0
XAR1	AR1H	AR1
XAR2	AR2H	AR2
XAR3	AR3H	AR3
XAR4	AR4H	AR4
XAR5	AR5H	AR5
XAR6	AR6H	AR6
XAR7	AR7H	AR7

Table 2−3. Extended Auxiliary Registers and Their Parts

Register	Referred To As ...	Accessibility
XARn	Extended auxiliary register n	Accessible via dedicated instructions only. XARn is not mapped to memory.
ARn	Auxiliary register n	Accessible via dedicated instructions and as a memory-mapped register
ARnH	High part of extended auxiliary register n	Not individually accessible. To access ARnH, you must access XARn.

XAR0−XAR7 or AR0−AR7 are used in the AR indirect addressing mode and the dual AR indirect addressing mode. Basic arithmetical, logical, and shift operations can be performed on AR0–AR7 in the A-unit arithmetic logic unit (ALU). These operations can be performed in parallel with address modifications performed on the auxiliary registers in the data-address generation unit (DAGEN).

The revision 3.0 CPU moves to a 23-bit flat data addressing mechanism. This removes the 64K word boundary restriction seen in the Revision 2 implementation. The primary restriction of Revision 2, from a programming point of view, is the inability to allow data tables to span across a 64KW data page boundary.

The data address generation (DAGEN) units are expanded from a 16-bit to a full 23-bit arithmetic unit. This allows address generation to cross or span the

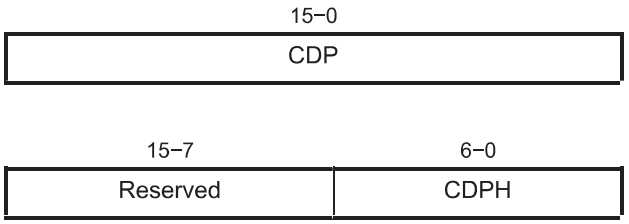
64KW data page boundaries, and this allows data tables that are greater than 64KW to be addresses directly. In Revision 2, if the pointer to a data table is incremented to a value greater than 64KW, the arithmetic value returned to the lower 16-bits does wrap around. For example, the pointer value 00 FFFFh + 1 becomes 00 0000h. This wrap around feature is not supported in Revision 3.0. All arithmetic operations for data address generation are based on 23-bit arithmetic.

There are some exceptions where the address generation cannot cross the 64KW data page boundary, and data address stays on the lower 16-bits:

- Stack pointers operation (see Section 2.6.7, *Stack Pointers*)
- Accesses to I/O space (see Section 3.4, *I/O Space*)
- Addressing register bits (see Section 6.7, *Addressing Register Bits*)
- Direct addressing mode(see Section 6.3, *Direct Addressing Modes*)

2.6.2 Coefficient Data Pointer (XCDP / CDP)

The CPU includes in its memory map a coefficient data pointer, CDP, and an associated extension register, CDPH:



The CPU can concatenate the two to form an extended CDP that is called XCDP (see Figure 2–5 and Table 2–4). The high part (CDPH) is used to specify the 7-bit main data page for accesses to data space. The low part (CDP) can be used as:

- A 16-bit offset to the 7-bit main data page (to form a 23-bit address)
- A bit address (in instructions that access individual bits or bit pairs)
- A general-purpose register or counter
- An index to select words relative to the start address of a circular buffer (see section 6.10, *Circular Addressing)*

Figure 2–5. Extended Coefficient Data Pointer and Its Parts

Table 2−4. Extended Coefficient Data Pointer and Its Parts

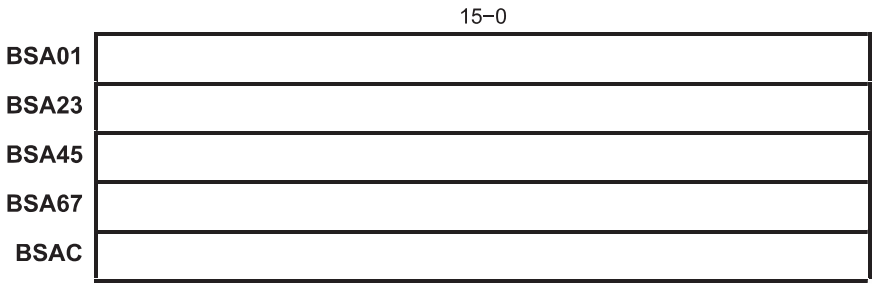
Register	Referred To As ...	Accessibility
XCDP	Extended coefficient data pointer	Accessible via dedicated instructions only. XCDP is not a register mapped to memory.
CDP	Coefficient data pointer	Accessible via dedicated instructions and as a memory-mapped register
CDPH	High part of extended coefficient data pointer	Accessible as a memory-mapped register. You can also access CDPH by accessing XCDP. There are no dedicated instructions for CDPH.

XCDP or CDP is used in the CDP indirect addressing mode and the coefficient indirect addressing mode. CDP can be used in any instruction that accesses a single data-space value; however, CDP is more advantageously used in dual multiply-and-accumulate (MAC) instructions because it provides a third, independent operand to the D-unit dual-MAC operator.

2.6.3 Circular Buffer Start Address Registers (BSA01, BSA23, BSA45, BSA67, BSAC)

The CPU includes five 16-bit circular buffer start address registers (see Figure 2−6) to enable you to define a circular buffer with a start address that is not bound by any alignment constraint.

Figure 2−6. Circular Buffer Start Address Registers



Each buffer start address register is associated with a particular pointer or pointers (see Table 2−5). A buffer start address is only added to the pointer value when the pointer is configured for circular addressing in status register ST2_55.

Table 2−5. Circular Buffer Start Address Registers and The Associated Pointers

Register	Pointer	Supplier of Main Data Page
BSA01	AR0 or AR1	AR0H for AR0 AR1H for AR1
BSA23	AR2 or AR3	AR2H for AR2 AR3H for AR3
BSA45	AR4 or AR5	AR4H for AR4 AR5H for AR5
BSA67	AR6 or AR7	AR6H for AR6 AR7H for AR7
BSAC	CDP	CDPH

As an example of using a buffer start address, consider the following instruction:

```
MOV *AR6, T2    ; Load T2 with a value from the circular
                ; buffer of words referenced by XAR6.
```

In this example, with AR6 configured for circular addressing, the address generated is of the following form. The main data page value (AR6H) is concatenated with the sum of AR6 and its associated buffer start address (BSA67).

AR6H:(BSA67 + AR6) = XAR6 + BSA67

When you run TMS320C54x DSP code in the compatible mode (C54CM = 1), make sure the buffer start address registers contain 0.

2.6.4 Circular Buffer Size Registers (BK03, BK47, BKC)

Three 16-bit circular buffer size registers (see Figure 2−7) specify the number of words (up to 65535) in a circular buffer. Each buffer size register is associated with a particular pointer or pointers (see Table 2−6).

Figure 2−7. Circular Buffer Size Registers

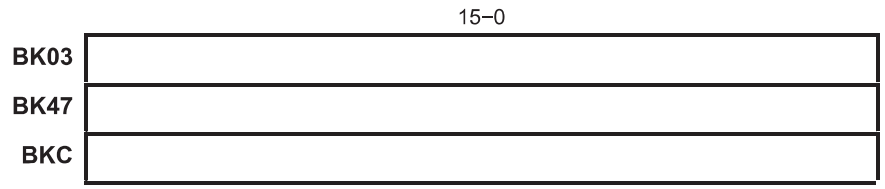


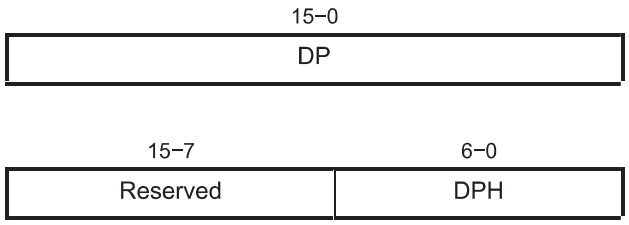
Table 2−6. Circular Buffer Size Registers and The Associated Pointers

Register	Pointer
BK03	AR0, AR1, AR2, or AR3
BK47	AR4, AR5, AR6, or AR7
BKC	CDP

In the TMS320C54x DSP-compatible mode (C54CM = 1), BK03 is used for all the auxiliary registers and BK47 is not used.

2.6.5 Data Page Register (XDP / DP)

The CPU includes in its memory map a data page register, DP, and an associated extension register, DPH:



The CPU can concatenate the two to form an extended DP that is called XDP (see Figure 2−8 and Table 2−7). The high part (DPH) is used to specify the 7-bit main data page for accesses to data space. The low part specifies a 16-bit offset (local data page) that is concatenated with the main data page to form a 23-bit address.

Figure 2−8. Extended Data Page Register and Its Parts

Table 2−7. Extended Data Page Register and Its Parts

Register	Referred To As ...	Accessibility
XDP	Extended data page register	Accessible via dedicated instructions only. XDP is not a register mapped to memory.
DP	Data page register	Accessible via dedicated instructions and as a memory-mapped register
DPH	High part of extended data page register	Accessible via dedicated instructions and as a memory-mapped register

In the DP direct addressing mode, XDP specifies a 23-bit address, and in the k16 absolute addressing mode, DPH is concatenated with a 16-bit immediate value to form a 23-bit address.

2.6.6 Peripheral Data Page Register (PDP)

For the PDP direct addressing mode, the 9-bit peripheral data page register (PDP) selects a 128-word page within the 64K-word I/O space.

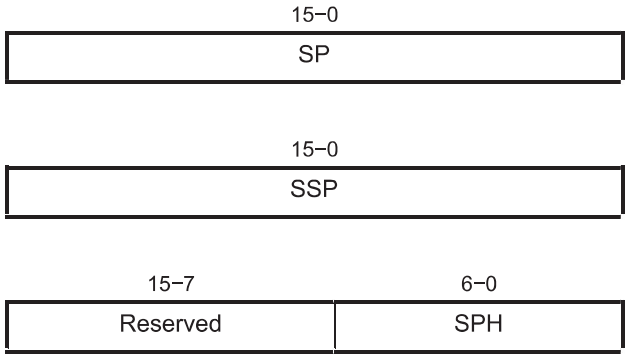
As shown in Figure 2−9, PDP is a 9-bit field within a 16-bit register location. Bits 15–9 of that location are ignored by the CPU.

This register is accessible via dedicated instructions and as a memory-mapped register.

Figure 2−9. Peripheral Data Page Register

2.6.7 Stack Pointers (XSP / SP, XSSP / SSP)

The CPU includes in its memory map a data stack pointer (SP), a system stack pointer (SSP), and an associated extension register (SPH):



See Figure 2–10 and Table 2–8. When accessing the data stack, the CPU concatenates SPH with SP to form an extended SP that is called XSP. XSP contains the address of the value last pushed onto the data stack. SPH holds the 7-bit main data page of memory, and SP points to the specific word on that page.

Similarly, when accessing the system stack, the CPU concatenates SPH with SSP to form XSSP. XSSP contains the address of the value last pushed onto the system stack.

Figure 2–10. Extended Stack Pointers

	22–16	15–0
XSP	SPH	SP
XSSP	SPH	SSP

Table 2−8. Stack Pointer Registers

Register	Referred To As ...	Accessibility
XSP	Extended data stack pointer	Accessible via dedicated instructions only. XSP is not a register mapped to memory.
SP	Data stack pointer	Accessible via dedicated instructions and as a memory-mapped register
XSSP	Extended system stack pointer	Accessible via dedicated instructions only. XSSP is not a register mapped to memory.
SSP	System stack pointer	Accessible via dedicated instructions and as a memory-mapped register
SPH	High part of XSP and XSSP	Accessible as a memory-mapped register. You can also access SPH by accessing XSP or XSSP. There are no dedicated instructions for SPH. **NOTE:** SPH is affected by writes to XSP or XSSP.

XSP is used in the SP direct addressing mode. The following instructions use and/or modify SP and SSP:

Instruction Type(s)	Description
Software interrupt, software trap, software reset, call unconditionally, call conditionally	These instructions push data onto the data stack and the system stack. SP and SSP are decremented before each pair of data values is pushed.
Push	This instruction pushes data onto the data stack only. SP is decremented before the data is pushed.
Return unconditionally, return conditionally, return from interrupt	These instructions pop data from the data stack and the system stack. SP and SSP are incremented after each pair of data values is popped.
Pop	This instruction pops data from the data stack only. SP is incremented after the data is popped.

All increments and decrements from the pointers listed above are done modulo 64K. You cannot address stack across data pages without changing the value in the extension register (SPH).

Note:

Although an increment past FFFFh, or a decrement past 0000h, causes the I/O address to wrap around, do not make use of this behavior. It is not supported.

2.7 Program Flow Registers (PC, RETA, CFCT)

Table 2−9 describes three registers used by the CPU to maintain proper program flow.

Table 2−9. Program Flow Registers

Register	Description
PC	Program counter. This 24-bit register holds the address of the 1 to 6 bytes of code being decoded in the I unit. When the CPU performs an interrupt or call, the current PC value (the return address) is stored, and then PC is loaded with a new address. When the CPU returns from an interrupt service routine or a called subroutine, the return address is restored to PC.
RETA	Return address register. If the selected stack configuration (see page 4-4) uses the fast-return process, RETA is a temporary holding place for the return address while a subroutine is being executed. RETA, along with CFCT, enables the efficient execution of multiple layers of subroutines. You can read from or write to RETA and CFCT as a pair with dedicated 32-bit load and store instructions.
CFCT	Control-flow context register. The CPU keeps a record of active repeat loops (the loop context). If the selected stack configuration (see page 4-4) uses the fast-return process, CFCT is a temporary holding place for the 8-bit loop context while a subroutine is being executed. CFCT, along with RETA, enables the efficient execution of multiple layers of subroutines. You can read from or write to RETA and CFCT as a pair with dedicated, 32-bit load and store instructions.

Note:

RETA and CFCT are cleared to 0 by a DSP hardware reset, and are not affected by push/pop instructions or by a software reset.

2.7.1 Context Bits Stored in CFCT

The CPU has internal bits for storing the loop context—the status (active or inactive) of repeat loops in a routine. When the CPU follows an interrupt or a call, the loop context is stored in CFCT. When the CPU returns from an interrupt or called subroutine, the loop context is restored from CFCT. The loop context bits have the following form in the 8-bit CFCT.

Bit(s)	Description
7	This bit reflects whether a single-repeat loop is active. 0 Not active 1 Active
6	This bit reflects whether a conditional single-repeat loop is active. 0 Not active 1 Active
5–4	Reserved
3–0	This 4-bit code reflects the status of the two possible levels of block-repeat loops, the outer (level 0) loop and the inner (level 1) loop. Depending on which type of block-repeat instruction you choose, an active loop is local (all its code is repeatedly executed from within the instruction buffer queue) or external (its code is repeatedly fetched and transferred through the buffer queue to the CPU).

Block-Repeat Code	Level 0 Loop Is ...	Level 1 Loop Is ...
0	Not active	Not active
2	Active, external	Not active
3	Active, local	Not active
7	Active, external	Active, external
8	Active, external	Active, local
9	Active, local	Active, local
Other: Reserved	–	–

2.8 Registers For Managing Interrupts

This section describes the following registers:

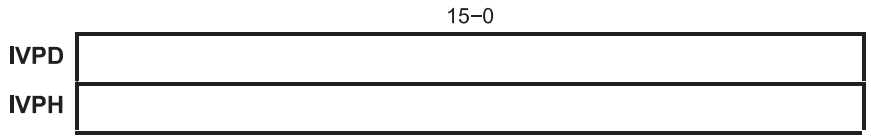
Register(s)	Function	See Section ...
IVPD	Points to interrupt vectors 0−15 and 24−31	2.8.1
IVPH	Points to interrupt vectors 16−23	2.8.1
IFR0, IFR1	Indicate which maskable interrupts have been requested	2.8.2
IER0, IER1	Enable or disable maskable interrupts	2.8.3
DBIER0, DBIER1	Configure select maskable interrupts as time-critical interrupts during debugging	2.8.4

2.8.1 Interrupt Vector Pointers (IVPD, IVPH)

Two 16-bit interrupt vector pointers IVPD and IVPH (see Figure 2−11) point to up to 32 interrupt vectors in program space. IVPD points to the 256-byte program page for interrupt vectors 0–15 and 24–31. IVPH points to the 256-byte program page for interrupt vectors 16–23.

If IVPD and IVPH have the same value, all of the interrupt vectors are in the same 256-byte program page. A DSP hardware reset loads both IVPs with FFFFh. The IVPs are not affected by a software reset instruction.

Figure 2−11. Interrupt Vector Pointers



Before you modify the IVPs, make sure that:

- Maskable interrupts are globally disabled (INTM = 1). This prevents a maskable interrupt from occurring before the IVPs are modified to point to new vectors.
- Each hardware nonmaskable interrupt has a vector and an interrupt service routine for the old IVPD value and for the new IVPD value. This prevents fetching of an illegal instruction code if a hardware nonmaskable interrupt occurs during the process of modifying the IVPD.

Table 2−10 shows how the vector addresses are formed for the different interrupt vectors. The CPU concatenates a 16-bit interrupt vector pointer with a vector number coded on 5 bits (for example, 00001 for IV1 and 10000 for IV16) and shifted left by 3 bits.

Table 2−10. Vectors and the Formation of Vector Addresses

		Vector Address		
Vector(s)	**Interrupt(s)**	**Bits 23−8**	**Bits 7−3**	**Bits 2−0**
IV0	Reset	IVPD	00000	000
IV1	Nonmaskable hardware interrupt, NMI	IVPD	00001	000
IV2–IV15	Maskable interrupts	IVPD	00010 to 01111	000
IV16–IV23	Maskable interrupts	IVPH	10000 to 10111	000
IV24	Bus error interrupt (maskable), BERRINT	IVPD	11000	000
IV25	Data log interrupt (maskable), DLOGINT	IVPD	11001	000
IV26	Real-time operating system interrupt (maskable), RTOSINT	IVPD	11010	000
IV27–IV31	General-purpose software-only interrupts INT27–INT31	IVPD	11011 to 11111	000

2.8.2 Interrupt Flag Registers (IFR0, IFR1)

The 16-bit interrupt flag registers, IFR1 and IFR0, contain flag bits for all the maskable interrupts. When a maskable interrupt request reaches the CPU, the corresponding flag is set to 1 in one of the IFRs. This indicates that the interrupt is pending or waiting for acknowledgement from the CPU. Figure 2−12 is a general representation of the C55x DSP IFRs. To see which interrupts are mapped to these bits, see the applicable C55x DSP data manual.

You can read the IFRs to identify pending interrupts, and write to the IFRs to clear pending interrupts. To clear an interrupt request (and clear its IFR bit to 0), write a 1 to the corresponding IFR bit. For example:

```
; Clear flags IF14 and IF2:
MOV #0100000000000100b, mmap(@IFR0)
```

All pending interrupts can be cleared by writing the current contents of the IFR back into the IFR. Acknowledgement of a software and a hardware interrupt request also clears the corresponding IFR bit. A device reset clears all IFR bits.

Figure 2−12. Interrupt Flag Registers

IFR1

15				11	10	9	8
Reserved					RTOSINTF	DLOGINTF	BERRINTF
R−0					R/W1C−0	R/W1C−0	R/W1C−0
7	6	5	4	3	2	1	0
IF23	IF22	IF21	IF20	IF19	IF18	IF17	IF16
R/W1C−0	R/W1C−0	R/W1C−0	R/W1C−0	R/W1C−0	R/W1C−0	R/W1C−0	R/W1C−0

IFR0

15	14	13	12	11	10	9	8
IF15	IF14	IF13	IF12	IF11	IF10	IF9	IF8
R/W1C−0	R/W1C−0	R/W1C−0	R/W1C−0	R/W1C−0	R/W1C−0	R/W1C−0	R/W1C−0
7	6	5	4	3	2	1	0
IF7	IF6	IF5	IF4	IF3	IF2	Reserved	
R/W1C−0	R/W1C−0	R/W1C−0	R/W1C−0	R/W1C−0	R/W1C−0	R−0	

Legend: R = Read access; W1C = Writing a 1 to this bit causes the CPU to clear this bit to 0; *-n* = Value after DSP hardware reset; Reserved = A write to this bit has no effect, and the bits in this field always appear as 0s during read operations.

2.8.2.1 RTOSINTF Bit in IFR1

Bit	Name	Description	Accessibility	HW Reset
10	RTOSINTF	Interrupt flag bit for the real-time operating system interrupt, RTOSINT	Read/Write	0

When you read the RTOSINTF bit, interpret it as follows:

RTOSINTF	Description
0	RTOSINT is not pending.
1	RTOSINT is pending.

To clear this flag bit to 0 (and clear its corresponding interrupt request), write a 1 to the bit.

2.8.2.2 DLOGINTF Bit in IFR1

Bit	Name	Description	Accessibility	HW Reset
9	DLOGINTF	Interrupt flag bit for the data log interrupt, DLOGINT	Read/Write	0

When you read the DLOGINTF bit, interpret it as follows:

DLOGINTF	Description
0	DLOGINT is not pending.
1	DLOGINT is pending.

To clear this flag bit to 0 (and clear its corresponding interrupt request), write a 1 to the bit.

2.8.2.3 BERRINTF Bit in IFR1

Bit	Name	Description	Accessibility	HW Reset
8	BERRINTF	Interrupt flag bit for the bus error interrupt, BERRINT	Read/Write	0

When you read the BERRINTF bit, interpret it as follows:

BERRINTF	Description
0	BERRINT is not pending.
1	BERRINT is pending.

To clear this flag bit to 0 (and clear its corresponding interrupt request), write a 1 to the bit.

2.8.2.4 IF16–IF23 Bits in IFR1

Bit	Name	Description	Accessibility	HW Reset
0	IF16	Interrupt flag bit 16	Read/Write	0
1	IF17	Interrupt flag bit 17	Read/Write	0
2	IF18	Interrupt flag bit 18	Read/Write	0
3	IF19	Interrupt flag bit 19	Read/Write	0
4	IF20	Interrupt flag bit 20	Read/Write	0
5	IF21	Interrupt flag bit 21	Read/Write	0
6	IF22	Interrupt flag bit 22	Read/Write	0
7	IF23	Interrupt flag bit 23	Read/Write	0

When you read these bits, interpret them as follows (x is a number from 16 to 23):

IFx	Description
0	The interrupt associated with interrupt vector x is not pending.
1	The interrupt associated with interrupt vector x is pending.

To clear a flag bit to 0 (and clear its corresponding interrupt request), write a 1 to the bit.

2.8.2.5 IF2–IF15 Bits in IFR0

Bit	Name	Description	Accessibility	HW Reset
2	IF2	Interrupt flag bit 2	Read/Write	0
3	IF3	Interrupt flag bit 3	Read/Write	0
4	IF4	Interrupt flag bit 4	Read/Write	0
5	IF5	Interrupt flag bit 5	Read/Write	0
6	IF6	Interrupt flag bit 6	Read/Write	0
7	IF7	Interrupt flag bit 7	Read/Write	0
8	IF8	Interrupt flag bit 8	Read/Write	0
9	IF9	Interrupt flag bit 9	Read/Write	0
10	IF10	Interrupt flag bit 10	Read/Write	0
11	IF11	Interrupt flag bit 11	Read/Write	0
12	IF12	Interrupt flag bit 12	Read/Write	0
13	IF13	Interrupt flag bit 13	Read/Write	0
14	IF14	Interrupt flag bit 14	Read/Write	0
15	IF15	Interrupt flag bit 15	Read/Write	0

When you read these bits, interpret them as follows (x is a number from 2 to 15):

IFx	Description
0	The interrupt associated with interrupt vector x is not pending.
1	The interrupt associated with interrupt vector x is pending.

To clear a flag bit to 0 (and clear its corresponding interrupt request), write a 1 to the bit.

2.8.3 Interrupt Enable Registers (IER0, IER1)

To enable a maskable interrupt, set its corresponding bit in IER0 or IER1 to 1. To disable a maskable interrupt, clear its corresponding enable bit to 0. At a DSP hardware reset, all the IER bits are cleared to 0, disabling all the maskable interrupts. Figure 2−13 is a general representation of the C55x DSP IERs. To see which interrupts are mapped to these bits, see the applicable C55x DSP data manual.

Note:

IER1 and IER0 are not affected by a software reset instruction. Initialize these registers before globally enabling (INTM = 0) the maskable interrupts.

Figure 2−13. Interrupt Enable Registers

IER1

15				11	10	9	8
Reserved					RTOSINTE	DLOGINTE	BERRINTE
R−0					R/W−0	R/W−0	R/W−0

7	6	5	4	3	2	1	0
IE23	IE22	IE21	IE20	IE19	IE18	IE17	IE16
R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0

IER0

15	14	13	12	11	10	9	8
IE15	IE14	IE13	IE12	IE11	IE10	IE9	IE8
R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0

7	6	5	4	3	2	1	0
IE7	IE6	IE5	IE4	IE3	IE2	Reserved	
R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R−0	

Legend: R = Read; W = Write; -*n* = Value after DSP hardware reset

2.8.3.1 RTOSINTE Bit in IER1

Bit	Name	Description	Accessibility	HW Reset
10	RTOSINTE	Enable bit for the real-time operating system interrupt, RTOSINT	Read/Write	0

The RTOSINTE bit enables or disables RTOSINT:

RTOSINTE	Description
0	RTOSINT is disabled.
1	RTOSINT is enabled.

2.8.3.2 DLOGINTE Bit in IER1

Bit	Name	Description	Accessibility	HW Reset
9	DLOGINTE	Enable bit for the data log interrupt, DLOGINT	Read/Write	0

The DLOGINTE bit enables or disables DLOGINT:

DLOGINTE	Description
0	DLOGINT is disabled.
1	DLOGINT is enabled.

2.8.3.3 BERRINTE Bit in IER1

Bit	Name	Description	Accessibility	HW Reset
8	BERRINTE	Enable bit for the bus error interrupt, BERRINT	Read/Write	0

The BERRINTE bit enables or disables BERRINT:

BERRINTE	Description
0	BERRINT is disabled.
1	BERRINT is enabled.

2.8.3.4 IE16–IE23 Bits in IER1

Bit	Name	Description	Accessibility	HW Reset
0	IE16	Interrupt enable bit 16	Read/Write	0
1	IE17	Interrupt enable bit 17	Read/Write	0
2	IE18	Interrupt enable bit 18	Read/Write	0
3	IE19	Interrupt enable bit 19	Read/Write	0
4	IE20	Interrupt enable bit 20	Read/Write	0
5	IE21	Interrupt enable bit 21	Read/Write	0
6	IE22	Interrupt enable bit 22	Read/Write	0
7	IE23	Interrupt enable bit 23	Read/Write	0

The functions of these bits can be summarized as follows, where x is a number from 16 to 23:

IEx	Description
0	The interrupt associated with interrupt vector x is disabled.
1	The interrupt associated with interrupt vector x is enabled.

2.8.3.5 IE2–IE15 Bits in IER0

Bit	Name	Description	Accessibility	HW Reset
2	IE2	Interrupt enable bit 2	Read/Write	0
3	IE3	Interrupt enable bit 3	Read/Write	0
4	IE4	Interrupt enable bit 4	Read/Write	0
5	IE5	Interrupt enable bit 5	Read/Write	0
6	IE6	Interrupt enable bit 6	Read/Write	0
7	IE7	Interrupt enable bit 7	Read/Write	0
8	IE8	Interrupt enable bit 8	Read/Write	0
9	IE9	Interrupt enable bit 9	Read/Write	0
10	IE10	Interrupt enable bit 10	Read/Write	0
11	IE11	Interrupt enable bit 11	Read/Write	0
12	IE12	Interrupt enable bit 12	Read/Write	0
13	IE13	Interrupt enable bit 13	Read/Write	0
14	IE14	Interrupt enable bit 14	Read/Write	0
15	IE15	Interrupt enable bit 15	Read/Write	0

The functions of these bits can be summarized as follows, where x is a number from 2 to 15:

IEx	Description
0	The interrupt associated with interrupt vector x is disabled.
1	The interrupt associated with interrupt vector x is enabled.

2.8.4 Debug Interrupt Enable Registers (DBIER0, DBIER1)

The 16-bit debug interrupt enable registers, DBIER1 and DBIER0, are used only when the CPU is *halted* in the real-time emulation mode of the debugger. If the CPU is *running* in real-time mode, the standard interrupt-handling process is used and the DBIERs are ignored.

A maskable interrupt enabled in a DBIER is defined as a *time-critical interrupt*. When the CPU is halted in the real-time mode, the only interrupts that are serviced are time-critical interrupts that are also enabled in an interrupt enable register (IER1 or IER0).

Read the DBIERs to identify time-critical interrupts. Write the DBIERs to enable or disable time-critical interrupts. To enable an interrupt, set its corresponding bit. To disable an interrupt, clear its corresponding bit. Figure 2−14 is a general representation of the C55x DSP DBIERs. To see which interrupts are mapped to these bits, see the applicable C55x DSP data manual.

Note:

DBIER1 and DBIER0 are not affected by a software reset instruction. Initialize these registers before you use the real-time emulation mode.

All DBIER bits are cleared to 0 by a DSP hardware reset, disabling all time-critical interrupts.

Figure 2−14. Debug Interrupt Enable Registers

DBIER1

15				11	10	9	8
Reserved					RTOSINTD	DLOGINTD	BERRINTD
R−0					R/W−0	R/W−0	R/W−0

7	6	5	4	3	2	1	0
DBIE23	DBIE22	DBIE21	DBIE20	DBIE19	DBIE18	DBIE17	DBIE16
R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0

DBIER0

15	14	13	12	11	10	9	8
DBIE15	DBIE14	DBIE13	DBIE12	DBIE11	DBIE10	DBIE9	DBIE8
R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0

7	6	5	4	3	2	1	0
DBIE7	DBIE6	DBIE5	DBIE4	DBIE3	DBIE2	Reserved	
R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R−0	

Legend: R = Read; W = Write; -*n* = Value after DSP hardware reset

2.8.4.1 RTOSINTD Bit in DBIER1

Bit	Name	Description	Accessibility	HW Reset
10	RTOSINTD	Debug enable bit for the real-time operating system interrupt, RTOSINT	Read/Write	0

The RTOSINTD bit enables or disables RTOSINT as a time-critical interrupt:

RTOSINTD	Description
0	RTOSINT is disabled.
1	RTOSINT is enabled. It is configured as a time-critical interrupt.

2.8.4.2 DLOGINTD Bit in DBIER1

Bit	Name	Description	Accessibility	HW Reset
9	DLOGINTD	Debug enable bit for the data log interrupt, DLOGINT	Read/Write	0

The DLOGINTD bit enables or disables DLOGINT as a time-critical interrupt:

DLOGINTD	Description
0	DLOGINT is disabled.
1	DLOGINT is enabled. It is configured as a time-critical interrupt.

2.8.4.3 BERRINTD Bit in DBIER1

Bit	Name	Description	Accessibility	HW Reset
8	BERRINTD	Debug enable bit for the bus error interrupt, BERRINT	Read/Write	0

The BERRINTD bit enables or disables BERRINT as a time-critical interrupt:

BERRINTD	Description
0	BERRINT is disabled.
1	BERRINT is enabled. It is configured as a time-critical interrupt.

2.8.4.4 DBIE16–DBIE23 Bits in DBIER1

Bit	Name	Description	Accessibility	HW Reset
0	DBIE16	Debug interrupt enable bit 16	Read/Write	0
1	DBIE17	Debug interrupt enable bit 17	Read/Write	0
2	DBIE18	Debug interrupt enable bit 18	Read/Write	0
3	DBIE19	Debug interrupt enable bit 19	Read/Write	0
4	DBIE20	Debug interrupt enable bit 20	Read/Write	0
5	DBIE21	Debug interrupt enable bit 21	Read/Write	0
6	DBIE22	Debug interrupt enable bit 22	Read/Write	0
7	DBIE23	Debug interrupt enable bit 23	Read/Write	0

The functions of these bits can be summarized as follows, where x is a number from 16 to 23:

DBIEx	Description
0	The interrupt associated with interrupt vector x is disabled.
1	The interrupt associated with interrupt vector x is enabled. The interrupt is configured as a time-critical interrupt.

2.8.4.5 DBIE2–DBIE15 Bits in DBIER0

Bit	Name	Description	Accessibility	HW Reset
2	DBIE2	Debug interrupt enable bit 2	Read/Write	0
3	DBIE3	Debug interrupt enable bit 3	Read/Write	0
4	DBIE4	Debug interrupt enable bit 4	Read/Write	0
5	DBIE5	Debug interrupt enable bit 5	Read/Write	0
6	DBIE6	Debug interrupt enable bit 6	Read/Write	0
7	DBIE7	Debug interrupt enable bit 7	Read/Write	0
8	DBIE8	Debug interrupt enable bit 8	Read/Write	0
9	DBIE9	Debug interrupt enable bit 9	Read/Write	0
10	DBIE10	Debug interrupt enable bit 10	Read/Write	0
11	DBIE11	Debug interrupt enable bit 11	Read/Write	0
12	DBIE12	Debug interrupt enable bit 12	Read/Write	0
13	DBIE13	Debug interrupt enable bit 13	Read/Write	0
14	DBIE14	Debug interrupt enable bit 14	Read/Write	0
15	DBIE15	Debug interrupt enable bit 15	Read/Write	0

The functions of these bits can be summarized as follows, where x is a number from 2 to 15:

DBIEx	Description
0	The interrupt associated with interrupt vector x is disabled.
1	The interrupt associated with interrupt vector x is enabled. The interrupt is configured as a time-critical interrupt.

2.9 Registers for Controlling Repeat Loops

This section describes registers that control the execution of repeat loops. Single-repeat registers are used for the repetition of a single instruction. Block-repeat registers are used for the repetition of one or more blocks of instructions.

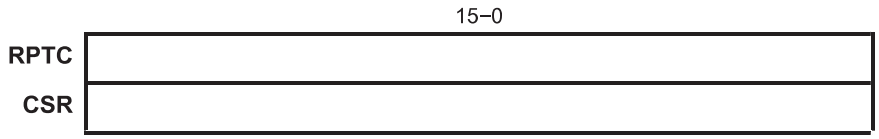
2.9.1 Single-Repeat Registers (RPTC, CSR)

The 16-bit single-repeat instruction registers RPTC and CSR enable repetition of a single-cycle instruction (or two single-cycle instructions that are executed in parallel). The number of repetitions, N, is loaded into the single-repeat counter (RPTC) before the first execution. After the first execution, the instruction is executed N more times; therefore, total execution is N+1 times.

In some syntaxes of the unconditional single-repeat instruction, you can use the computed single-repeat register (CSR) to specify the number N. The value from CSR is copied into RPTC before the first execution of the instruction or instruction pair to be repeated.

As shown in Figure 2−15, RPTC and CSR have 16 bits, enabling up to 65536 consecutive executions of an instruction (the first execution plus 65535 repetitions).

Figure 2−15. Single-Repeat Registers



2.9.2 Block-Repeat Registers (BRC0–1, BRS1, RSA0–1, REA0–1)

The block-repeat instructions enable you to form loops that repeat blocks of instructions. You can have one block-repeat loop nested inside another, creating an inner (level 1) loop and an outer (level 0) loop. Table 2−11 describes the C55x DSP registers associated with level 0 and level 1 loops. As described in the following paragraphs, the use of these registers is affected by the C54x DSP-compatible mode bit (C54CM), which is introduced in section 2.10.2.4.

If C54CM = 0: C55x DSP native mode ...
The CPU keeps a record of active repeat loops when an interrupt or call is performed while the loop is active (see the description for CFCT in section 2.7, *Program Flow Registers (PC, RETA, CFCT)*). This enables the use of level 0 resources in subroutines. When the CPU decodes a block-repeat instruction, it first determines whether a loop is already being executed. If the CPU detects an active level 0 loop, it uses the level 1 loop registers; otherwise, it uses the level 0 loop registers.

If C54CM = 1: C54x DSP-compatible mode ...
Block-repeat instructions activate the level 0 loop registers only. Level 1 loop registers are not used. Nested block-repeat operations can be implemented as on the C54x DSPs, using context saving/restoring and the block-repeat active flag (BRAF). A block-repeat instruction sets BRAF, and BRAF is cleared at the end of the block-repeat operation when BRC0 contains 0. For more details about BRAF, see section 2.10.2.2, *BRAF Bit of ST1_55*.

When a block-repeat loop begins in the C54x DSP-compatible mode (C54CM = 1), the BRAF bit is automatically set to indicate that a loop is in progress. If your program must switch modes from C54CM = 1 to C54CM = 0, the BRAF bit must be cleared before or during the switch. There are three options:

- Wait until the loop is finished (when BRAF is cleared automatically) and then clear C54CM
- Clear BRAF (this also stops the loop) and then clear C54CM
- Clear BRAF and C54CM at the same time with an instruction that modifies status register ST1_55

Note:

Make sure that the last three instructions of a level 0 loop do not write to BRC0. Likewise, make sure that the last three instructions of a level 1 loop do not write to BRC1.

Table 2−11. Block-Repeat Register Descriptions

Level 0 Loop Registers		Level 1 Loop Registers (Not Used If C54CM = 1)	
Register	**Description**	**Register**	**Description**
BRC0	Block-repeat counter 0. This 16-bit register contains the number of times to repeat the instruction block after its initial execution.	BRC1	Block-repeat counter 1. This 16-bit register contains the number of times to repeat the instruction block after its initial execution.
RSA0	Block-repeat start address register 0. This 24-bit register contains the address of the first instruction in the instruction block.	RSA1	Block-repeat start address register 1. This 24-bit register contains the address of the first instruction in the instruction block.
REA0	Block-repeat end address register 0. This 24-bit register contains the address of the last instruction in the instruction block.	REA1	Block-repeat end address register 1. This 24-bit register contains the address of the last instruction in the instruction block.
		BRS1	BRC1 save register. Whenever BRC1 is loaded, BRS1 is loaded with the same value. The content of BRS1 is not modified during the execution of the level 1 loop. Each time the level 1 loop is triggered, BRC1 is reinitialized from BRS1. This feature enables initialization of BRC1 outside of the level 0 loop, reducing the time needed for each repetition.

Note: The 24-bit register values are stored in two consecutive 16-bit locations. Bits 23–16 are stored at the lower address (the eight most significant bits in this location are ignored by the CPU). Bits 15–0 are stored at the higher address. For example, RSA0(23–16) is accessible at address 00 003Ch, and RSA0(15–0) is accessible at address 00 003Dh.

2.10 Status Registers (ST0_55−ST3_55)

These four 16-bit registers (see Figure 2−16) contain control bits and flag bits. The control bits affect the operation of the C55x DSP and the flag bits reflect the current status of the DSP or indicate the results of operations.

ST0_55, ST1_55, and ST3_55 are each accessible at two addresses (see section 2.2, *Memory-Mapped Registers*). At one address, all the TMS320C55x DSP bits are available. At the other address (the protected address), the bits highlighted in Figure 2−16 cannot be modified. The protected address is provided to support TMS320C54x DSP code that was written to access ST0, ST1, and PMST (the C54x DSP counterpart of ST3_55). Reserved bits are not available for use.

Note:

Always write 1100b (Ch) to bits 11−8 of ST3_55.

Some C55x DSP devices do not have an instruction cache; these devices do not use the CAFRZ, CAEN, and CACLR bits.

Figure 2−16. Status Registers

ST0_55

15	14	13	12	11	10	9
ACOV2†	**ACOV3**†	**TC1**†	TC2	CARRY	ACOV0	ACOV1
R/W−0	R/W−0	R/W−1	R/W−1	R/W−1	R/W−0	R/W−0

8	7	6	5	4	3	2	1	0
DP[15:7]								
R/W−0								

ST1_55

15	14	13	12	11	10	9	8
BRAF	CPL	XF	HM	INTM	**M40**†	SATD	SXMD
R/W−0	R/W−0	R/W−1	R/W−0	R/W−1	R/W−0	R/W−0	R/W−1

7	6	5	4	3	2	1	0
C16	FRCT	**C54CM**†	ASM				
R/W−0	R/W−0	R/W−1	R/W−0				

ST2_55

15	14	13	12	11	10	9	8
ARMS	Reserved		DBGM	EALLOW	RDM	Reserved	CDPLC
R/W−0	R−11b		R/W−1	R/W−0	R/W−0	R−0	R/W−0

7	6	5	4	3	2	1	0
AR7LC	AR6LC	AR5LC	AR4LC	AR3LC	AR2LC	AR1LC	AR0LC
R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0

ST3_55

15	14	13	12	11	10	9	8
CAFRZ†#	**CAEN**†#	**CACLR**†#	**HINT**†‡	Reserved (always write as 1100b)			
R/W−0	R/W−0	R/W−0	R/W−1	R/W−1100b			

7	6	5	4	3	2	1	0
CBERR†	MPNMC§	**SATA**†	Reserved	Reserved	CLKOFF	SMUL	SST
R/W−0	R/W−pins	R/W−0	R/W−0¶	R−0	R/W−0	R/W−0	R/W−0

Legend: R = Read; W = Write; *-n* = Value after DSP hardware reset

† Highlighted bit: If you write to the protected address of the status register, a write to this bit has no effect, and the bit always appears as a 0 during read operations.

‡ The HINT bit is not used for all C55x DSP host port interfaces (HPIs). Consult the documentation for the specific C55x DSP.

§ The reset value of MPNMC may be dependent on the state of predefined pins at reset. To check this for a particular C55x DSP, see its data manual.

¶ Always write 0 to this bit

Some C55x DSP devices do not have an instruction cache; these devices do not use bits CAFRZ, CAEN, and CACLR.

2.10.1 ST0_55 Bits

This section describes the bits of ST0_55 in alphabetical order.

2.10.1.1 ACOV0, ACOV1, ACOV2, and ACOV3 Bits of ST0_55

Each of the four accumulators has its own overflow flag in ST0_55:

Bit	Name	Description	Accessibility	HW Reset
9	ACOV1	AC1 overflow flag	Read/Write	0
10	ACOV0	AC0 overflow flag	Read/Write	0
14	ACOV3	AC3 overflow flag	Read/Write	0
15	ACOV2	AC2 overflow flag	Read/Write	0

For each of these flags:

- Overflow detection depends on the M40 bit in ST1_55:
  - M40 = 0: Overflow is detected at bit position 31.
  - M40 = 1: Overflow is detected at bit position 39.

  If you need compatibility with TMS320C54x DSP code, make sure M40 = 0.
- ACOVx is set when an overflow occurs in ACx, where x is 0, 1, 2, or 3.
- Once an overflow occurs, ACOVx remains set until one of the following events occurs:
  - A DSP hardware or software reset is performed.
  - The CPU executes a conditional branch, call, return, or execute instruction that tests the state of ACOVx.
  - ACOVx is explicitly cleared by a status bit clear instruction. For example, you can clear ACOV1 by using the following instruction:

    ```
    BCLR ACOV1
    ```

    (To set ACOV1, use BSET ACOV1.)

2.10.1.2 CARRY Bit of ST0_55

Bit	Name	Description	Accessibility	HW Reset
11	CARRY	Carry bit	Read/Write	1

The following are main points about the carry bit:

- Carry/borrow detection depends on the M40 bit in ST1_55:
  - M40 = 0: Carry/borrow is detected with respect to bit position 31.
  - M40 = 1: Carry/borrow is detected with respect to bit position 39.

If you need compatibility with TMS320C54x DSP code, make sure M40 = 0.

- When an addition is performed in the D-unit arithmetic logic unit (D-unit ALU), if the addition generates a carry, CARRY is set; if no carry is generated, CARRY is cleared. There is one exception to this behavior: When the following syntax is used (shifting Smem by 16 bits), CARRY is set for a carry but is not affected if no carry is generated.

```
ADD Smem <<#16, [ACx,] ACy
```

- When a subtraction is performed in the D-unit ALU, if the subtraction generates a borrow, CARRY is cleared; if no borrow is generated, CARRY is set. There is one exception to this behavior: When the following syntax is used (shifting Smem by 16 bits), CARRY is cleared for a borrow but is not affected if no borrow is generated.

```
SUB Smem <<#16, [ACx,] ACy
```

- CARRY is modified by the logical shift instructions.
- For signed shift instructions and rotate instructions, you can choose whether CARRY is modified.
- The following instruction syntaxes modify CARRY to indicate particular computation results when the destination register (dst) is an accumulator:

MIN [src,] dst	Minimum comparison
MAX [src,] dst	Maximum comparison
ABS [src,] dst	Absolute value
NEG [src,] dst	Negate

- You can clear and set CARRY with the following instructions:

```
BCLR CARRY   ; Clear CARRY
BSET CARRY   ; Set CARRY
```

2.10.1.3 DP Bit Field of ST0_55

Bits	Name	Description	Accessibility	HW Reset
8–0	DP	Copy of the 9 most significant bits of the data page register (DP)	Read/Write	0

This 9-bit field is provided for compatibility with code transferred from the TMS320C54x DSPs. TMS320C55x DSPs have a data page register independent of ST0_55. Any change to bits 15–7 of this data page register—DP(15–7)—is reflected in the DP status bits. Any change to the DP status bits is reflected in DP(15–7). Clear/Set status register bit instructions affecting this

bit field is not allowed. When generating addresses for the DP direct addressing mode, the CPU uses the full data page register, DP(15−0). You are not required to use the DP status bits; you can modify DP directly.

Note:

If you want to load ST0_55 but do not want the access to change the content of the data page register, use an OR or an AND operation with a mask value that does not modify the 9 least significant bits (LSBs) of ST0_55. For an OR operation, put 0s in the 9 LSBs of the mask value. For an AND operation, put 1s in the 9 LSBs of the mask value.

2.10.1.4 TC1 and TC2 Bits of ST0_55

Bit	Name	Description	Accessibility	HW Reset
12	TC2	Test/control flag 2	Read/Write	1
13	TC1	Test/control flag 1	Read/Write	1

The main function of the test/control bit is to hold the result of a test performed by specific instructions. The following are main points about the test/control bits:

- All the instructions that affect a test/control flag allow you to choose whether TC1 or TC2 is affected.
- TCx (where x = 1 or 2) or a Boolean expression of TCx can be used as a trigger in any conditional instruction.
- You can clear and set TC1 and TC2 with the following instructions:

```
BCLR TC1  ; Clear TC1
BSET TC1  ; Set TC1
BCLR TC2  ; Clear TC2
BSET TC2  ; Set TC2
```

2.10.2 ST1_55 Bits

This section describes the bits of ST1_55 in alphabetical order.

2.10.2.1 ASM Bit Field of ST1_55

Bits	Name	Description	Accessibility	HW Reset
4–0	ASM	Accumulator shift mode bit	Read/Write	00000b

ASM is not used by native TMS320C55x instructions but is available to support TMS320C54x code running on the TMS320C55x DSP. In a C54x DSP, the

ASM field supplies a signed shift count for special instructions that shift an accumulator value. The C55x DSP ASM field is used in the C54x DSP-compatible mode (C54CM = 1).

Before reading further, it is important to know that the C55x DSP register that contains ASM (status register ST1_55) is accessible at two addresses. One address, 00 0003h, is to be used by native C55x DSP instructions. The other address, 00 0007h, is provided to support C54x DSP code that accesses ST1 at 0007h.

If **C54CM = 1** (C54x DSP-compatible mode):

- Whenever ASM is loaded by a write to address 00 0007h, the 5-bit ASM value is sign-extended to 16 bits and written to temporary register 2 (T2). Clear/set status register bit instructions affecting this bit field is not allowed. When a C54x DSP instruction requests the CPU to shift an accumulator according to ASM, the CPU uses the shift count in T2.
- Whenever T2 is loaded, the five least significant bits are copied to ASM.
- Because T2 is tied to ASM, T2 is not available as a general-purpose data register.

If **C54CM = 0**:

- ASM is ignored. During an accumulator shift operation, the CPU reads the shift count from the temporary register (T0, T1, T2, or T3) that was specified in the C55x DSP instruction, or from a constant embedded in the C55x DSP instruction.
- T2 can be used as a general-purpose data register. Writing to address 00 0007h does not affect T2, and writing to T2 does not affect ASM.

If C54CM bit = 0 and an MMR write updates C54CM bit to 1 at the same time that the ASM field is updated, then the sign extension of the value loaded in T2 is done accordingly to the new value of the C54CM bit (C54CM = 1).

2.10.2.2 BRAF Bit of ST1_55

Bit	Name	Description	Accessibility	Reset Value
15	BRAF	Block-repeat active flag	Read/Write	0

If **C54CM = 0**: BRAF is not used. The status of repeat operations is maintained automatically by the CPU (see the description for CFCT in section 2.7, *Program Flow Registers (PC, RETA, CFCT)).*

If **C54CM = 1**: Reading BRAF indicates the status of a block-repeat operation:

BRAF	Block-Repeat Activity
0	No block-repeat operation is active.
1	A block-repeat operation is active.

To stop an active block-repeat operation in the C54x DSP-compatible mode, you can clear BRAF with the following instruction:

```
BCLR BRAF    ; Clear BRAF
```

You can set BRAF with the following instruction:

```
BSET BRAF    ; Set BRAF
```

BRAF also can be set or cleared with an instruction that modifies ST1_55.

If C54CM bit = 0 and an MMR write updates C54CM bit to 1 at the same time that the BRAF bit is updated, then BRAF bit operations are accordingly to the new value of the C54CM bit (C54CM = 1).

Functionality of BRAF. A block-repeat loop begins with a block-repeat instruction such as RPTB. BRAF is set at the address phase of this block-repeat instruction to indicate that a loop is active.

Each time the last instruction of the loop enters the decode phase of the pipeline, the CPU checks the values of BRAF and counter register (BRC0). If BRAF = 1 and BRC0 > 0, the CPU decrements BRC0 by 1 and begins the next iteration of the loop. Otherwise, the CPU stops the loop. (In either case, the last instruction completes its passage through the pipeline.)

BRAF is cleared in the following cases:

- The last instruction of the loop enters the decode phase, and BRC0 is decremented to 0. BRAF is automatically cleared one cycle later.
- An instruction writes 0 to the block-repeat counter, BRC0. BRAF is automatically cleared one cycle later.
- A far branch (FB) or far call (FCALL) instruction is executed. (BRAF is *not* cleared by the execution of other call or branch instructions, or by the execution of an INTR or TRAP instruction.)
- BRAF is manually cleared by a BCLR BRAF instruction or an instruction that modifies status register ST1_55.

BRAF is saved and restored with ST1_55 during the context switches caused by an interrupt and a return-from-interrupt instruction. BRAF is not saved when the CPU responds to a call instruction.

If a block-repeat loop is in progress and your program must switch modes from C54CM = 1 to C54CM = 0, the BRAF bit must be cleared before or during the switch. There are three options:

- Wait until the loop is finished (when BRAF is cleared automatically) and then clear C54CM.
- Clear BRAF (this also stops the loop) and then clear C54CM.
- Clear BRAF and C54CM at the same time with an instruction that modifies ST1_55.

Pipeline considerations. As already mentioned, the CPU clears BRAF one cycle after executing an instruction that clears BRC0. This modification of BRAF is not pipeline-protected. To ensure that BRAF is modified before another instruction reads BRAF, you may need to insert instructions between the instruction that clears BRC0 and the instruction that reads BRAF. For example:

```
MOV #0, mmap(BRC0)     ; Clear BRC0.
NOP                    ; Wait for BRAF to be cleared.
NOP
NOP
MOV mmap(ST1_55),  AR0    ; Read ST1_55 (including BRAF).
```

The number of instructions to insert depends on when the first instruction clears BRC0:

Pipeline Phase When BRC0 Is Cleared†	Instructions to Insert
Address (AD) phase	0
Execute (X) phase	2
Write (W) phase	3

† Consult the instruction set documentation for the active pipeline phase of a given syntax.

This pipeline issue can also affect when the loop ends. To ensure that BRAF is modified before the last instruction of the loop reaches the decode phase, you must insert 5 or 6 cycles between the instruction that clears BRAF and the last instruction:

Pipeline Phase When BRAF Is Modified†	Instructions to Insert
Execute (X) phase	5
Write (W) phase	6

† Consult the instruction set documentation for the active pipeline phase of a given syntax.

Updating BRAF prior to a return instruction (RET or RETI) is protected in the pipeline. After the return, if the next instruction reads BRAF, it reads the updated value.

2.10.2.3 C16 Bit of ST1_55

Bit	Name	Description	Accessibility	HW Reset
7	C16	Dual 16-bit arithmetic mode bit	Read/Write	0

In the TMS320C54x DSP-compatible mode (C54CM = 1), BRAF indicates/controls the status of a block-repeat operation. In this mode, the execution of some instructions is affected by C16. C16 determines whether such an instruction is executed in a single 32-bit operation (double-precision arithmetic) or in two parallel 16-bit operations (dual 16-bit arithmetic).

If **C54CM = 1**: The arithmetic performed in the D-unit ALU depends on C16:

C16	Dual 16-Bit Mode Is ...
0	Off. For an instruction that is affected by C16, the D-unit ALU performs one 32-bit operation.
1	On. For an instruction that is affected by C16, the D-unit ALU performs two 16-bit operations in parallel.

If **C54CM = 0**: The CPU ignores C16. The instruction syntax alone determines whether dual 16-bit arithmetic or 32-bit arithmetic is used.

You can clear and set C16 with the following instructions:

```
BCLR C16     ; Clear C16
BSET C16     ; Set C16
```

2.10.2.4 C54CM Bit of ST1_55

Bit	Name	Description	Accessibility	HW Reset
5	C54CM	TMS320C54x DSP-compatible mode bit	Read/Write	1

The C54CM bit determines whether the CPU will support code that was developed for a TMS320C54x DSP:

C54CM	C54x DSP-Compatible Mode Is ...
0	Disabled. The CPU supports code written for a TMS320C55x (C55x) DSP.
1	Enabled. This mode must be set when you are using code that was originally developed for a TMS320C54x (C54x) DSP. All the C55x DSP CPU resources remain available; therefore, as you translate code, you can take advantage of the additional features on the C55x DSP to optimize your code.

Change modes with the following instructions and assembler directives:

```
BCLR C54CM      ; Clear C54CM (happens at run time)
.C54CM_off      ; Tell assembler C54CM = 0

BSET C54CM      ; Set C54CM (happens at run time)
.C54CM_on       ; Tell the assembler C54CM = 1
```

Do not modify C54CM within a block-repeat loop as shown in this example:

```
       RPBTLOCAL end1   ; Start of loop 1
          :
       BSET C54CM
          :
end1   MOV AC0, dbl(*AR3+) ; End of loop 1
```

Also, do not modify C54CM in parallel with a block-repeat instruction such as:

```
       BCLR C54CM || RPTB end2  ; Start of loop 2
          :
end2   MOV AC1, dbl(*AR4+)      ; End of loop 2
```

2.10.2.5 CPL Bit of ST1_55

Bit	Name	Description	Accessibility	HW Reset
14	CPL	Compiler mode bit	Read/Write	0

The CPL bit determines which of two direct addressing modes is active:

CPL	Direct Addressing Mode Selected
0	DP direct addressing mode. Direct accesses to data space are made relative to the data page register (DP).
1	SP direct addressing mode. Direct accesses to data space are made relative to the data stack pointer (SP). The DSP is said to be in compiler mode.

Note: Direct addresses to I/O space are always made relative to the peripheral data page register (PDP).

Change modes with the following instructions and assembler directives:

```
BCLR CPL     ; Clear CPL (happens at run time)
.CPL_off     ; Tell assembler CPL = 0

BSET CPL     ; Set CPL (happens at run time)
.CPL_on      ; Tell the assembler CPL = 1
```

2.10.2.6 FRCT Bit of ST1_55

Bit	Name	Description	Accessibility	HW Reset
6	FRCT	Fractional mode bit	Read/Write	0

The FRCT bit turns the fractional mode on or off:

FRCT	Fractional Mode Is ...
0	Off. Results of multiply operations are not shifted.
1	On. Results of multiply operations are shifted left by 1 bit for decimal point adjustment. This is required when you multiply two signed Q15 values and you need a Q31 result.

You can clear and set FRCT with the following instructions:

```
BCLR FRCT    ; Clear FRCT
BSET FRCT    ; Set FRCT
```

2.10.2.7 HM Bit of ST1_55

Bit	Name	Description	Accessibility	HW Reset
12	HM	Hold mode bit	Read/Write	0

When the external memory interface (EMIF) of the DSP receives a HOLD request, the DSP places the EMIF output pins in the high-impedance state. Depending on HM, the DSP may also stop internal program execution:

HM	Hold Mode
0	The DSP continues executing instructions from internal program memory.
1	The DSP stops executing instructions from internal program memory.

You can use the following instructions to clear and set HM:

```
BCLR HM      ; Clear HM
BSET HM      ; Set HM
```

2.10.2.8 INTM Bit of ST1_55

Bit	Name	Description	Accessibility	HW Reset
11	INTM	Interrupt mode bit	Read/Write	1

The INTM bit globally enables or disables the maskable interrupts as shown in the following table. This bit has no effect on nonmaskable interrupts (those that cannot be blocked by software).

INTM	Description
0	All unmasked interrupts are enabled.
1	All maskable interrupts are disabled.

The following are main points about the INTM bit:

- Modify the INTM bit with status bit clear and set instructions (see the following examples). The only other instructions that affect INTM are the software interrupt instruction and the software reset instruction, which set INTM before branching to the interrupt service routine.

  ```
  BCLR INTM    ; Clear INTM
  BSET INTM    ; Set INTM
  ```

  In CPU cores with revisions older than 2.2, there is no pipeline protection by the hardware between the INTM bit update and an interrupt jamming. Because the INTM bit is updated in the execute phase of the pipeline, an interrupt can be taken in between any of the 5 instructions following the INTM set instruction which globally disables interrupts. In CPU cores with revisions 2.2 or newer, no interrupt is taken after the INTM set instruction.

- The state of the INTM bit is automatically saved when the CPU approves an interrupt request. Specifically, the INTM bit is saved when the CPU saves ST1_55 to the data stack.

- Before executing an interrupt service routine (ISR) triggered by an INTR #5 instruction, by the RESET instruction, or by a hardware interrupt source, the CPU automatically sets the INTM bit to globally disable the maskable interrupts. The TRAP #k5 instruction does not affect the INTM bit. The ISR can re-enable the maskable interrupts by clearing the INTM bit.
- A return-from-interrupt instruction restores the INTM bit from the data stack.
- When the CPU is halted in the real-time emulation mode of the debugger, INTM is ignored and only time-critical interrupts can be serviced (see the description for the debug interrupt enable registers in section 2.8.4).

2.10.2.9 M40 Bit of ST1_55

Bit	Name	Description	Accessibility	HW Reset
10	M40	Computation mode bit for the D unit	Read/Write	0

The M40 bit selects one of two computation modes for the D unit:

M40 D-Unit Computation Mode Is ...

0 32-bit mode. In this mode:

- ❑ The sign bit is extracted from bit position 31.
- ❑ During arithmetic, the carry is determined with respect to bit position 31.
- ❑ Overflows are detected at bit position 31.
- ❑ During saturation, the saturation value is 00 7FFF FFFFh (positive overflow) or FF 8000 0000h (negative overflow).
- ❑ Accumulator comparisons versus 0 are done using bits 31–0.
- ❑ Shift or rotate operations are performed on 32-bit values.
- ❑ During left shifts or rotations of accumulators, bits shifted out are extracted from bit position 31.
- ❑ During right shifts or rotations of accumulators, bits shifted in are inserted at bit position 31.
- ❑ During signed shifts of accumulators, if SXMD = 0, 0 is copied into the accumulator's guard bits; if SXMD = 1, bit 31 is copied into the accumulator's guard bits. During any rotations or logical shifts of accumulators, the guard bits of the destination accumulator are cleared.

Note: In the TMS320C54x DSP-compatible mode (C54CM = 1), there are some exceptions: An accumulator's sign bit is extracted from bit position 39. Accumulator comparisons versus 0 are done using bits 39–0. Signed shifts are performed as if M40 = 1.

1 40-bit mode. In this mode:

- ❑ The sign bit is extracted from bit position 39.
- ❑ During arithmetic, the carry is determined with respect to bit position 39.
- ❑ Overflows are detected at bit position 39.
- ❑ During saturation, the saturation value is 7F FFFF FFFFh (positive overflow) or 80 0000 0000h (negative overflow).
- ❑ Accumulator comparisons versus 0 are done using bits 39–0.
- ❑ Shift or rotate operations are performed on 40-bit values.
- ❑ During left shifts or rotations of accumulators, bits shifted out are extracted from bit position 39.
- ❑ During right shifts or rotations of accumulators, bits shifted in are inserted at bit position 39.

You can clear and set M40 with the following instructions:

```
BCLR M40     ; Clear M40
BSET M40     ; Set M40
```

2.10.2.10 SATD Bit of ST1_55

Bit	Name	Description	Accessibility	HW Reset
9	SATD	Saturation mode bit for the D unit	Read/Write	0

The SATD bit determines whether the CPU saturates overflow results in the D unit:

SATD	Saturation Mode in the D Unit Is ...
0	Off. No saturation is performed.
1	On. If an operation performed by the D unit results in an overflow, the result is saturated. The saturation depends on the value of the M40 bit: M40 = 0 The CPU saturates the result to 00 7FFF FFFFh (positive overflow) or FF 8000 0000h (negative overflow). M40 = 1 The CPU saturates the result to 7F FFFF FFFFh (positive overflow) or 80 0000 0000h (negative overflow).

If you want compatibility with TMS320C54x DSP code, make sure M40 = 0.

You can clear and set SATD with the following instructions:

```
BCLR SATD    ; Clear SATD
BSET SATD    ; Set SATD
```

2.10.2.11 SXMD Bit of ST1_55

Bit	Name	Description	Accessibility	HW Reset
8	SXMD	Sign-extension mode bit for the D unit	Read/Write	1

The SXMD bit turns on or off the sign-extension mode, which affects accumulator loads, and also affects additions, subtractions, and signed shift operations that are performed in the D unit:

SXMD **Sign-Extension Mode Is ...**

0 Off. When sign-extension mode is off:

- For 40-bit operations, 16-bit or smaller operands are zero extended to 40 bits.
- For the conditional subtract instruction, any 16-bit divisor produces the expected result.
- When the D-unit arithmetic logic unit (ALU) is locally configured in its dual 16-bit mode (by a dual 16-bit arithmetic instruction):
  - 16-bit values used in the higher part of the D-unit ALU are zero extended to 24 bits.
  - 16-bit accumulator halves are zero extended if they are shifted right.
- During a signed shift of an accumulator, if it is a 32-bit operation (M40 = 0), 0 is copied into the accumulator's guard bits (39–32).
- During a signed right shift of an accumulator, the shifted value is zero extended.

1 On. In this mode:

- For 40-bit operations, 16-bit or smaller operands are sign extended to 40 bits.
- For the conditional subtract instruction, the 16-bit divisor must be a positive value (its most significant bit (MSB) must be 0).
- When the D-unit ALU is locally configured in its dual 16-bit mode (by a dual 16-bit arithmetic instruction):
  - 16-bit values used in the higher part of the D-unit ALU are sign extended to 24 bits.
  - 16-bit accumulator halves are sign extended if they are shifted right.
- During a signed shift of an accumulator, if it is a 32-bit operation (M40 = 0), bit 31 is copied into the accumulator's guard bits (39–32).
- During a signed right shift of an accumulator, the shifted value is sign extended, unless the uns() expression qualifier is used to designate the accumulator value as unsigned.

SXMD is ignored during some operations:

- For unsigned operations (boolean logic operations, rotate operations, and logical shift operations), input operands are always zero extended to 40 bits, regardless of the value of SXMD.
- For operations performed in a multiply-and-accumulate unit (MAC), 16-bit input operands are sign extended to 17 bits, regardless of the value of SXMD.

- If an operand in an instruction is enclosed in the operand qualifier uns(), the operand is treated as unsigned, regardless of the value of SXMD.

You can clear and set SXMD with the following instructions:

```
BCLR SXMD    ; Clear SXMD
BSET SXMD    ; Set SXMD
```

2.10.2.12 XF Bit of ST1_55

Bit	Name	Description	Accessibility	HW Reset
13	XF	External flag	Read/Write	1

The XF bit is a general-purpose output bit. This bit is directly connected to the XF pin on those C55x DSP devices that have an XF pin. Setting the XF bit drives the XF pin high. Clearing the XF bit drives the XF pin low. The following instructions clear and set XF:

```
BCLR XF      ; Clear XF
BSET XF      ; Set XF
```

2.10.3 ST2_55 Bits

This section describes the bits of ST2_55 in alphabetical order.

2.10.3.1 AR0LC–AR7LC Bits of ST2_55

The CPU has eight auxiliary registers, AR0–AR7. Each auxiliary register ARn (n = 0, 1, 2, 3, 4, 5, 6, or 7) has its own linear/circular configuration bit in ST2_55:

Bit	Name	Description	Accessibility	HW Reset
n	ARnLC	ARn linear/circular configuration bit	Read/Write	0

Each ARnLC bit determines whether ARn is used for linear addressing or circular addressing:

ARnLC	ARn Is Used For ...
0	Linear addressing
1	Circular addressing

For example, if AR3LC = 0, AR3 is used for linear addressing; if AR3LC = 1, AR3 is used for circular addressing.

You can clear and set the ARnLC bits with the status bit set/clear instruction. For example, the following instructions respectively clear and set AR3LC. To modify other ARnLC bits, replace the 3 with the appropriate number.

```
BCLR AR3LC      ; Clear AR3LC
BSET AR3LC      ; Set AR3LC
```

2.10.3.2 ARMS Bit of ST2_55

Bit	Name	Description	Accessibility	HW Reset
15	ARMS	AR mode switch	Read/Write	0

The ARMS bit determines the CPU mode used for the AR indirect addressing mode:

ARMS	AR Indirect Operands Available
0	DSP mode operands, which provide efficient execution of DSP intensive applications. Among these operands are those that use reverse carry propagation when adding to or subtracting from a pointer. Short-offset operands are not available.
1	Control mode operands, which enable optimized code size for control system applications. The short-offset operand *ARn(short(#k3)) is available. (Other offsets require a 2-byte extension on an instruction, and instructions with these extensions cannot be executed in parallel with other instructions.)

Change modes with the following instructions and assembler directives:

```
BCLR ARMS     ; Clear ARMS(happens at run time)
.ARMS_off     ; Tell assembler ARMS = 0

BSET ARMS     ; Set ARMS (happens at run time)
.ARMS_on      ; Tell assembler ARMS = 1
```

2.10.3.3 CDPLC Bit of ST2_55

Bit	Name	Description	Accessibility	HW Reset
8	CDPLC	CDP linear/circular configuration bit	Read/Write	0

The CDPLC bit determines whether the coefficient data pointer (CDP) is used for linear addressing or circular addressing:

CDPLC	CDP Is Used For ...
0	Linear addressing
1	Circular addressing

You can clear and set CDPLC with the following instructions:

```
BCLR CDPLC      ; Clear CDPLC
BSET CDPLC      ; Set CDPLC
```

2.10.3.4 DBGM Bit of ST2_55

Bit	Name	Description	Accessibility	HW Reset
12	DBGM	Debug mode bit	Read/Write	1

The DBGM bit provides the capability to block debug events during time-critical portions of a program:

DBGM	Debug Events Are ...
0	Enabled.
1	Disabled. The emulator cannot access memory or registers. Software breakpoints still cause the CPU to halt, but hardware breakpoints or halt requests are ignored.

The following are main points about the DBGM bit:

- For pipeline protection, the DBGM bit can only be modified by status bit clear and set instructions (see the following examples). No other instructions affect the DBGM bit.

```
BCLR DBGM    ; Clear DBGM
BSET DBGM    ; Set DBGM
```

- The state of the DBGM bit is automatically saved when the CPU approves an interrupt request or fetches the INTR #k5, TRAP #k5, or RESET instruction. Specifically, the DBGM bit is saved when the CPU saves ST2_55 to the data stack.

- Before executing an interrupt service routine (ISR) triggered by the INTR #k5 or RESET instruction, or by a hardware interrupt source, the CPU automatically sets the DBGM bit to disable debug events. The ISR can reenable debug events by clearing the DBGM bit.

- A return-from-interrupt instruction restores the DBGM bit from the data stack.

2.10.3.5 EALLOW Bit of ST2_55

Bit	Name	Description	Accessibility	HW Reset
11	EALLOW	Emulation access enable bit	Read/Write	0

The EALLOW bit enables or disables write access to non-CPU emulation registers:

EALLOW	Write Access To Non-CPU Emulation Registers Is ...
0	Disabled
1	Enabled

The following are main points about the EALLOW bit:

- The state of the EALLOW bit is automatically saved when the CPU approves an interrupt request or fetches the INTR #k5, TRAP #k5, or RESET instruction. Specifically, the EALLOW bit is saved when the CPU saves ST2_55 to the data stack.
- Before executing an interrupt service routine (ISR) triggered by the INTR #k5, TRAP #k5, or RESET instruction, or by a hardware interrupt source, the CPU automatically clears the EALLOW bit to prevent accesses to the emulation registers. The ISR can re-enable access by setting the EALLOW bit:

  ```
  BSET EALLOW
  ```

  (To clear EALLOW, you can use BCLR EALLOW.)
- A return-from-interrupt instruction restores the EALLOW bit from the data stack.

2.10.3.6 RDM Bit of ST2_55

Bit	Name	Description	Accessibility	HW Reset
10	RDM	Rounding mode bit	Read/Write	0

Certain instructions executed in the D unit allow you to indicate whether an operand is to be rounded. The type of rounding performed depends on the value of the RDM bit:

RDM	Rounding Mode Selected
0	Round to the infinite. The CPU adds 8000h (2 raised to the 15th power) to the 40-bit operand. Then the CPU clears bits 15 through 0 to generate a rounded result in a 24- or 16-bit representation. For a 24-bit representation, only bits 39 through 16 of the result are meaningful. For a 16-bit representation, only bits 31 through 16 of the result are meaningful.

1	Round to the nearest. The rounding depends on bits 15 through 0 of the 40-bit operand, as shown by the following if statements. The rounded result is in a 24-bit representation (in bits 39 through 16) or a 16-bit representation (in bits 31 through 16).
	If (0 =< bits 15–0 < 8000h)
	CPU clears bits 15–0
	If (8000h < bits 15–0 < 10000h)
	CPU adds 8000h to the operand and then clears bits 15–0
	If (bits 15–0 == 8000h)
	If bits 31–16 contain an odd value
	CPU adds 8000h to the operand and then clears bits 15–0
	If bits 31−16 contain an even value
	CPU clears bits 15−0

If you need compatibility with TMS320C54x DSP code, make sure RDM = 0 and C54CM = 1. When C54CM = 1 (C54x DSP-compatible mode enabled), the following instructions do not clear bits 15–0 of the result after the rounding:

SATR [ACx,] ACy	Saturate with rounding
RND [ACx,] ACy	Round
LMS Xmem,Ymem,ACx,ACy	Least mean square

You can clear and set RDM with the following instructions:

```
BCLR RDM     ; Clear RDM
BSET RDM     ; Set RDM
```

2.10.4 ST3_55 Bits

This section describes the bits of ST3_55 in alphabetical order.

2.10.4.1 CACLR Bit of ST3_55

Bit	Name	Description	Accessibility	HW Reset
13	CACLR	Cache clear bit	Read/Write	0

To clear (or flush) the instruction cache (invalidate all the lines of its data arrays), set the CACLR bit. You can set CACLR using the following instruction:

```
BSET CACLR      ; Set CACLR
```

Once set, CACLR remains 1 until the flush process is complete, at which time CACLR is automatically reset to 0. Therefore, you can poll CACLR to get the status:

CACLR	The Flush Process Is...
0	Complete
1	Not complete. All cache blocks are invalid. The number of cycles required to flush the cache depends on the memory architecture. When the cache is flushed, the content of the prefetch queue in the instruction buffer unit is automatically flushed.

2.10.4.2 CAEN Bit of ST3_55

Bit	Name	Description	Accessibility	HW Reset
14	CAEN	Cache enable bit	Read/Write	0

The CAEN bit enables or disables the program cache:

CAEN	Cache Is ...
0	Disabled. The cache controller never receives a program request. All program requests are handled either by the internal memory or the external memory, depending on the address decoded.
1	Enabled. Program code is fetched from the cache, from the internal memory, or from the external memory, depending on the address decoded.

Some important notes:

- When the cache is disabled by clearing the CAEN bit, the content of the instruction buffer queue in the I unit is automatically flushed.
- You can clear and set CAEN using the following instructions

```
BCLR CAEN    ; Clear CAEN
BSET CAEN    ; Set CAEN
```

2.10.4.3 CAFRZ Bit of ST3_55

Bit	Name	Description	Accessibility	HW Reset
15	CAFRZ	Cache freeze bit	Read/Write	0

CAFRZ enables you to lock the instruction cache, so that its contents are not updated on a cache miss but are still available for cache hits. The contents of the cache remain undisturbed until CAFRZ is cleared. The following table summarizes the role of CAFRZ:

CAFRZ	Description
0	The cache is in its default operating mode.
1	The cache is frozen (the cache content is locked).

You can clear and set CAFRZ using the following instructions:

```
BCLR CAFRZ      ; Clear CAFRZ
BSET CAFRZ      ; Set CAFRZ
```

2.10.4.4 CBERR Bit of ST3_55

Bit	Name	Description	Accessibility	HW Reset
7	CBERR	CPU bus error flag	Read/Write (Can only write 0)	0

The CBERR bit is set when an internal bus error is detected. This error causes the CPU to set the bus error interrupt flag (BERRINTF) in interrupt flag register 1 (IFR1). Some important points follow:

- Writing a 1 to the CBERR bit has no effect. This bit is 1 only if an internal bus error has occurred.
- The interrupt service routine for the bus error interrupt (BERRINT) must clear the CBERR bit before it returns control to the interrupted program code:

```
BCLR CBERR      ; Clear CBERR
```

The CBERR bit can be summarized as follows:

CBERR	Description
0	The flag has been cleared by your program or by a reset.
1	An internal bus error has been detected.

Note:

When a bus error occurs, the functionality of the instruction that caused the error, and of any instruction executed in parallel, can not be assured.

2.10.4.5 CLKOFF Bit of ST3_55

Bit	Name	Description	Accessibility	HW Reset
2	CLKOFF	CLKOUT disable bit	Read/Write	0

When CLKOFF = 0, the CLKOUT pin is enabled; the associated clock signal appears on the pin. When CLKOFF = 1, the CLKOUT pin is disabled.

You can clear and set CLKOFF with the following instructions:

```
BCLR CLKOFF     ; Clear CLKOFF
BSET CLKOFF     ; Set CLKOFF
```

2.10.4.6 HINT Bit of ST3_55

Bit	Name	Description	Accessibility	HW Reset
12	HINT	Host interrupt bit	Read/Write	1

Use the HINT bit to send an interrupt request to a host processor by way of the host port interface. You produce an active-low interrupt pulse by clearing and then setting the HINT bit:

```
BCLR HINT    ; Clear HINT
BSET HINT    ; Set HINT
```

Note:

The HINT bit is not used for all C55x DSP host port interfaces (HPIs). Consult the documentation for the specific C55x DSP.

2.10.4.7 MPNMC Bit of ST3_55

Bit	Name	Description	Accessibility	HW Reset
6	MPNMC	Microprocessor/ microcomputer mode bit	Read/Write	May be dependent on the state of predefined pins at reset. To check this for a particular C55x DSP, see its data manual.

The MPNMC bit enables or disables the on-chip ROM:

MPNMC	Mode
0	Microcomputer mode. The on-chip ROM is enabled; it is addressable in program space.
1	Microprocessor mode. The on-chip ROM is disabled; it is not in the program-space map.

Some important notes:

- The reset value of the MPNMC bit may be dependent on the state of predefined pins at reset. To check this for a particular C55x DSP, see its data manual.
- The software reset instruction does not affect the MPNMC bit.
- You can clear and set MPNMC using the following instructions:

```
BCLR MPNMC      ; Clear MPNMC
BSET MPNMC      ; Set MPNMC
```

- An instruction that changes the MPNMC bit must not be followed too closely by a branch instruction. Otherwise, the CPU may use the old MPNMC value and, as a result, fetch the next instruction from the incorrect memory location. The minimum number of instruction cycles needed to separate an MPNMC-update instruction and a branch instruction depends on the type of branch instruction used. Table 2−12 divides branch instructions into three categories, and Table 2−13 shows the minimum number of separation cycles needed for each category.

Table 2−12. Categories of Branch Instructions

Category I	Category II	Category III
B L7	RET (when slow return selected)	B ACx
B L16	RETCC (when slow return selected)	CALL ACx
B P24	RETI (when slow return selected)	
BCC I4, cond		
BCC L8, cond		
BCC L16, cond		
BCC P24, cond		
CALL L16		
CALL P24		
CALLCC L16, cond		
CALLCC P24, cond		
RET (when fast return selected)		
RETCC (when fast return selected)		
RETI (when fast return selected)		

Table 2−13. Minimum Number of Instruction Cycles Required Between an MPNMC-Update Instruction and a Branch Instruction

	Cycles Required Before Subsequent Branch Instruction		
MPNMC-Update Instruction	**Category I**	**Category II**	**Category III**
One of the following instructions: BSET MPNMC BSET k4, ST3_55 BCLR MPNMC BCLR k4, ST3_55	4	0	0
An instruction that changes MPNMC when writing to the memory-mapped address for ST3_55	5	1	0

Consider the following example in which the BSET instruction changes MPNMC. Table 2−12 specifies CALL as a category I branch instruction. Table 2−13 indicates that 4 cycles are needed between the BSET MPNMC instruction and a category I branch instruction. In this example, the 4 cycles are provided by inserting four NOP (no operation) instructions. Other instructions could be placed here instead.

```
BSET  MPNMC
NOP
NOP
NOP
NOP
CALL #SubroutineA
```

2.10.4.8 SATA Bit of ST3_55

Bit	Name	Description	Accessibility	HW Reset
5	SATA	Saturation mode bit for the A unit	Read/Write	0

The SATA bit determines whether the CPU saturates overflow results of the A-unit arithmetic logic unit (A-unit ALU):

SATA	Saturation Mode in the A Unit Is ...
0	Off. No saturation is performed.
1	On. If a calculation in the A-unit ALU results in an overflow, the result is saturated to 7FFFh (for overflow in the positive direction) or 8000h (for overflow in the negative direction).

You can clear and set SATA with the following instructions:

```
BCLR SATA    ; Clear SATA
BSET SATA    ; Set SATA
```

2.10.4.9 SMUL Bit of ST3_55

Bit	Name	Description	Accessibility	HW Reset
1	SMUL	Saturation-on-multiplication mode bit	Read/Write	0

The SMUL bit turns the saturation-on-multiplication mode on or off:

SMUL	Saturation-On-Multiplication Mode Is ...
0	Off
1	On. When SMUL = 1, FRCT = 1, and SATD = 1, the result of 18000h × 18000h is saturated to 7FFF FFFFh (regardless of the value of the M40 bit). This forces the product of the two negative numbers to be a positive number.

For multiply-and-accumulate/subtract instructions, the saturation is performed after the multiplication and before the addition/subtraction.

You can clear and set SMUL with the following instructions:

```
BCLR SMUL    ; Clear SMUL
BSET SMUL    ; Set SMUL
```

2.10.4.10 SST Bit of ST3_55

Bit	Name	Description	Accessibility	Reset Value
0	SST	Saturate-on-store mode bit	Read/Write	0

In the TMS320C54x DSP-compatible mode (C54CM = 1), the execution of some accumulator-store instructions is affected by SST. When SST is 1, the 40-bit accumulator value is saturated to a 32-bit value before the store operation. If the accumulator value is shifted, the CPU performs the saturation after the shift.

If **C54CM = 1**: SST turns the saturation-on-store mode on or off.

SST	Saturation-On-Store Mode Is ...
0	Off
1	On. For an instruction that is affected by SST, the CPU saturates a shifted or unshifted accumulator value before storing it. The saturation depends on the value of the sign-extension mode bit (SXMD): SXMD = 0 The 40-bit value is treated as unsigned. If the 40-bit value is greater than 00 7FFF FFFFh, the CPU produces the 32-bit result 7FFF FFFFh. SXMD = 1 The 40-bit value is treated as signed. If the 40-bit value is less than 00 8000 0000h, the CPU produces the 32-bit result 8000 0000h. If the 40-bit value is greater than 00 7FFF FFFFh, the CPU produces 7FFF FFFFh.

If **C54CM = 0**: The CPU ignores SST. The instruction syntax alone determines whether saturation occurs.

You can clear and set SST with the following instructions:

```
BCLR SST     ; Clear SST
BSET SST     ; Set SST
```