

## Programmable Digital Signal Processors

### Introduction:

Leading manufacturers of integrated circuits such as Texas Instruments (TI), Analog devices & Motorola manufacture the digital signal processor (DSP) chips. These manufacturers have developed a range of DSP chips with varied complexity.

The TMS320 family consists of two types of single chips DSPs: 16-bit fixed point & 32-bit floating-point. These DSPs possess the operational flexibility of high-speed controllers and the numerical capability of array processors

### Commercial Digital Signal-Processing Devices:

There are several families of commercial DSP devices. Right from the early eighties, when these devices began to appear in the market, they have been used in numerous applications, such as communication, control, computers, Instrumentation, and consumer electronics. The architectural features and the processing power of these devices have been constantly upgraded based on the advances in technology and the application needs. However, their basic versions, most of them have Harvard architecture, a single-cycle hardware multiplier, an address generation unit with dedicated address registers, special addressing modes, on-chip peripherals interfaces.

Of the various families of programmable DSP devices that are commercially available, the three most popular ones are those from Texas Instruments, Motorola, and Analog Devices. Texas Instruments was one of the first to come out with a commercial programmable DSP with the introduction of its TMS32010 in 1982.

Architectural Feature	TMS320C25	DSP 56000	ADSP2100
Data representation format	16-bit fixed	24-bit fixed point	16-bit fixed point
Hardware multiplier	16 x 16	24 x 24	16 x 16
ALU	32 bits	56 bits	40 bits
Internal buses	16-bit program bus 16-bit data bus	24-bit program bus 2 x 24-bit data buses 24-bit global	24-bit program bus bus 16-bit data bus 16-bit result

External buses	16-bit program/data bus	databus 24-bit program/data bus	bus 24-bit program bus 16-bit data bus
On-chip Memory	544 words RAM 4K words ROM	512 words PROM 2 x 256 words data RAM 2 x 256 words data ROM	-
Off-chip memory	64 K words program 64k words data	64K words program 2 x 64K words data	16K words program 16K words data 16 words program
Cache memory	-	-	125 nsecc.
Instruction cycle time	100 nsec	97.5 nsec.	
Special addressing modes	Bit reversed	Modulo Bit reversed	Modulo Bit reversed
Data address generators	1	2	2
Interfacing features	Synchronous serial I/O DMA	Synchronous and Asynchronous serial I/O DMA	DMA

### The architecture of TMS320C54xx digital signal processors:

TMS320C54xx processors retain in the basic Harvard architecture of their predecessor, TMS320C25, but have several additional features, which improve their performance over it. Figure 3.1 shows a functional block diagram of TMS320C54xx processors. They have one program and three data memory spaces with separate buses, which provide simultaneous accesses to program instruction and two data operands and enables writing of result at the same time. Part of the memory is implemented on-chip and consists of combinations of ROM, dual-access RAM, and single-access RAM. Transfers between the memory spaces are also possible.

The central processing unit (CPU) of TMS320C54xx processors consists of a 40-bit arithmetic logic unit (ALU), two 40-bit accumulators, a barrel shifter, a 17x17 multiplier, a 40-bit adder, data address generation logic (DAGEN) with its own arithmetic unit, and program address generation logic (PAGEN). These major functional units are supported by a number of registers and logic in the architecture.

A powerful instruction set with a hardware-supported, single-instruction repeat and block repeat operations, block memory move instructions, instructions that pack two or three simultaneous reads, and arithmetic instructions with parallel store and load make these devices very efficient for running high-speed DSP algorithms.

Several peripherals, such as a clock generator, a hardware timer, a wait state generator, parallel I/O ports, and serial I/O ports, are also provided on-chip. These peripherals make it convenient to interface the signal processors to the outside world.

In these following sections, we examine in detail the various architectural features of the TMS320C54xx family of processors.

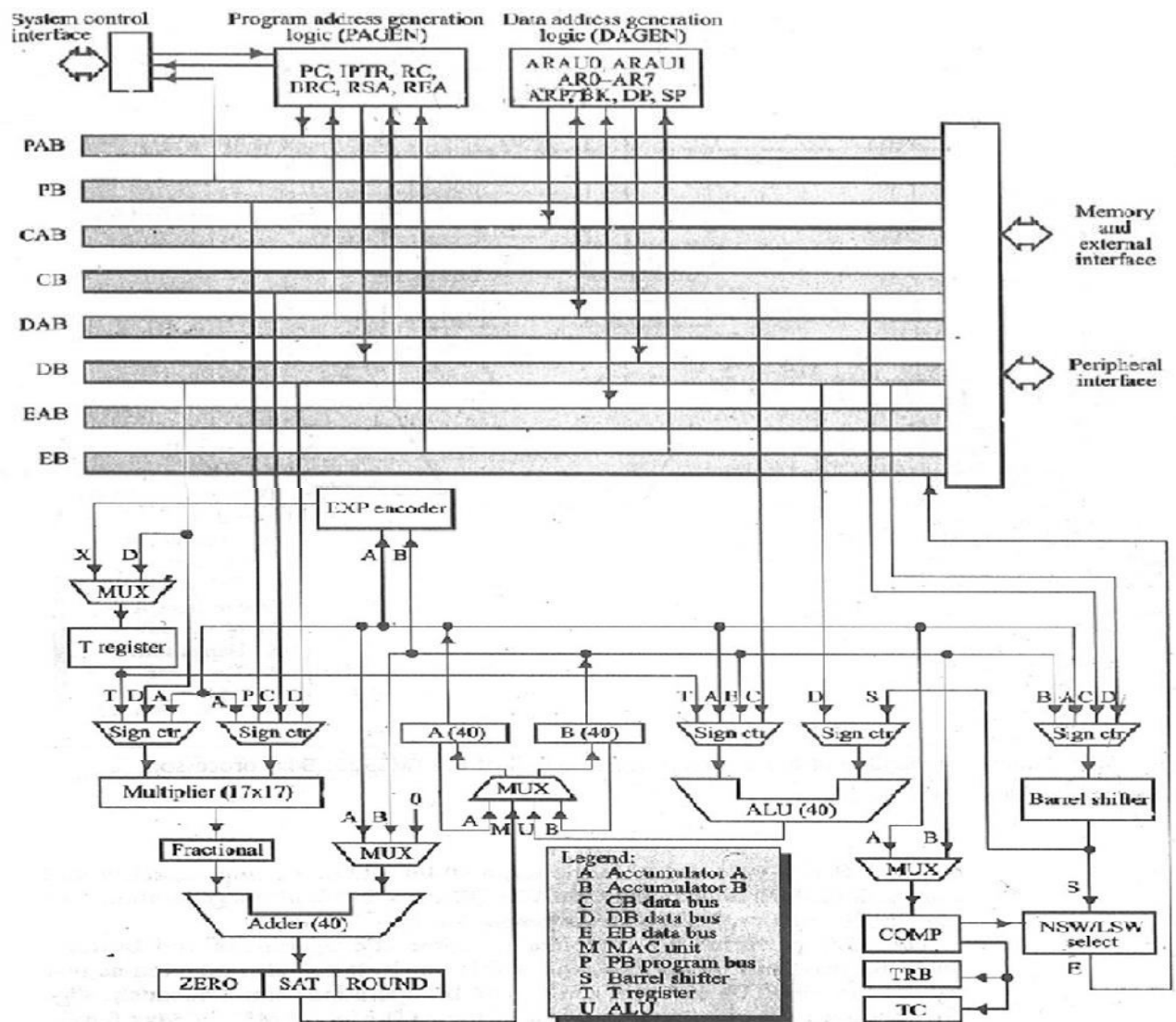


Figure 3.1. Functional architecture for TMS320C54xx processors.

## **Bus Structure:**

The performance of a processor gets enhanced with the provision of multiple buses to provide simultaneous access to various parts of memory or peripherals. The 54xx architecture is built around four pairs of 16-bit buses with each pair consisting of an address bus and a data bus. As shown in Figure 3.1, these are

The program bus pair (**PAB, PB**); which carries the instruction code from the program memory.

Three data bus pairs (**CAB, CB; DAB, DB; and EAB, EB**); which interconnected the various units within the CPU. In Addition the pair CAB, CB and DAB, DB are used to read from the data memory, while

The pair **EAB, EB**; carries the data to be written to the memory. The '54xx can generate up to two data-memory addresses per cycle using the two auxiliary register arithmetic unit (ARAU0 and ARAU1) in the DAGEN block. This enables accessing two operands simultaneously.

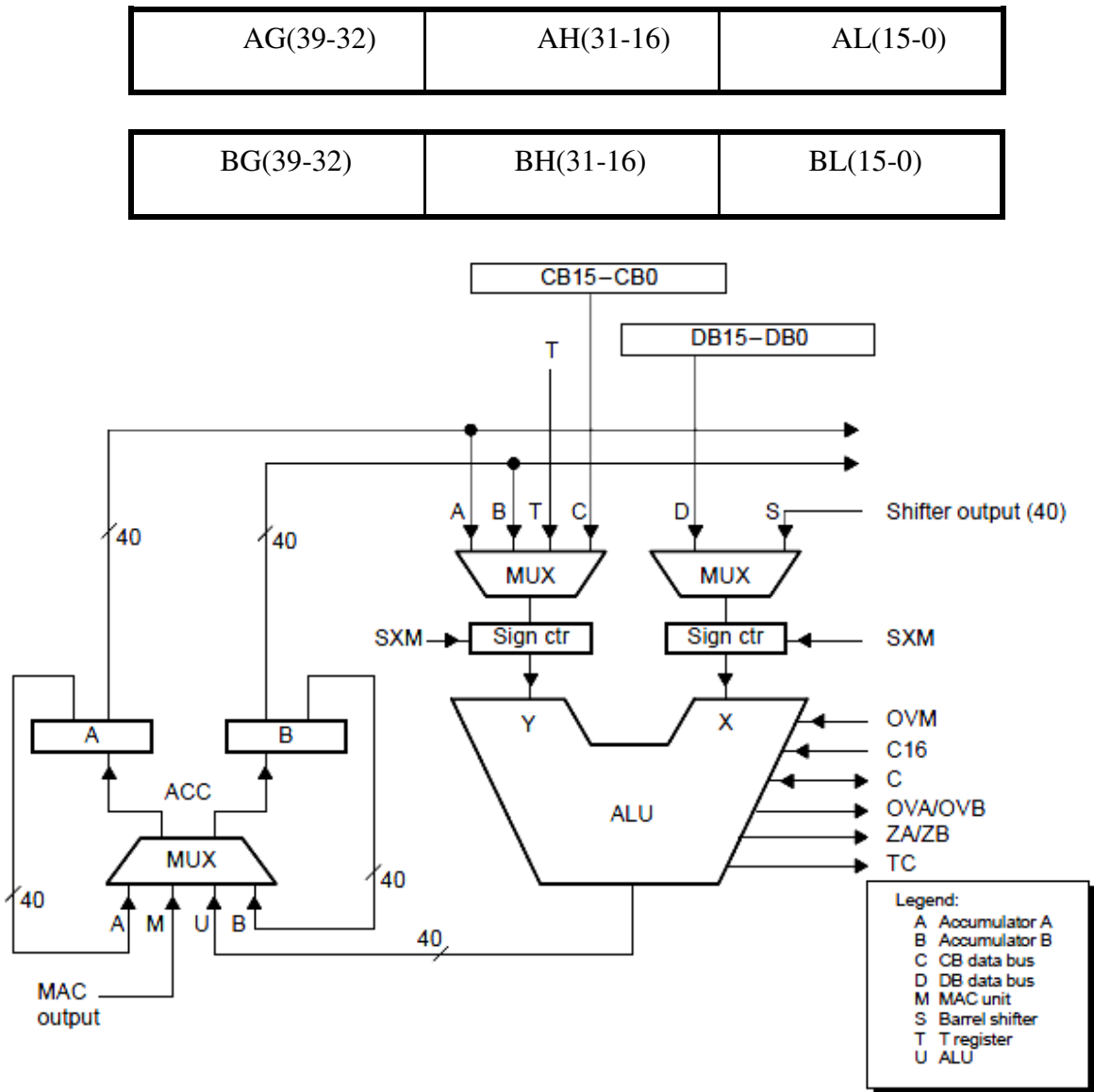
## **Central Processing Unit (CPU):**

The '54xx CPU is common to all the '54xx devices. The '54xx CPU contains a 40-bit arithmetic logic unit (**ALU**); two 40-bit accumulators (**A** and **B**); a barrel shifter; a 17 x 17-bit multiplier; a 40-bit adder; a compare, select and store unit (**CSSU**); an exponent encoder(**EXP**); a data address generation unit (**DAGEN**); and a program address generation unit (**PAGEN**).

The ALU performs 2's complement arithmetic operations and bit-level Boolean operations on 16, 32, and 40-bit words. It can also function as two separate 16-bit ALUs and perform two 16-bit operations simultaneously. Figure 3.2 show the functional diagram of the ALU of the TMS320C54xx family of devices.

**Accumulators A and B**; store the output from the ALU or the multiplier/adder block and provide a second input to the ALU. Each accumulators is divided into three parts: guards bits (bits 39-32), high-order word (bits-31-16), and low-order word (bits 15-0), which can be stored and retrieved individually.

Each accumulator is memory-mapped and partitioned. It can be configured as the destination registers. The guard bits are used as a head margin for computations.

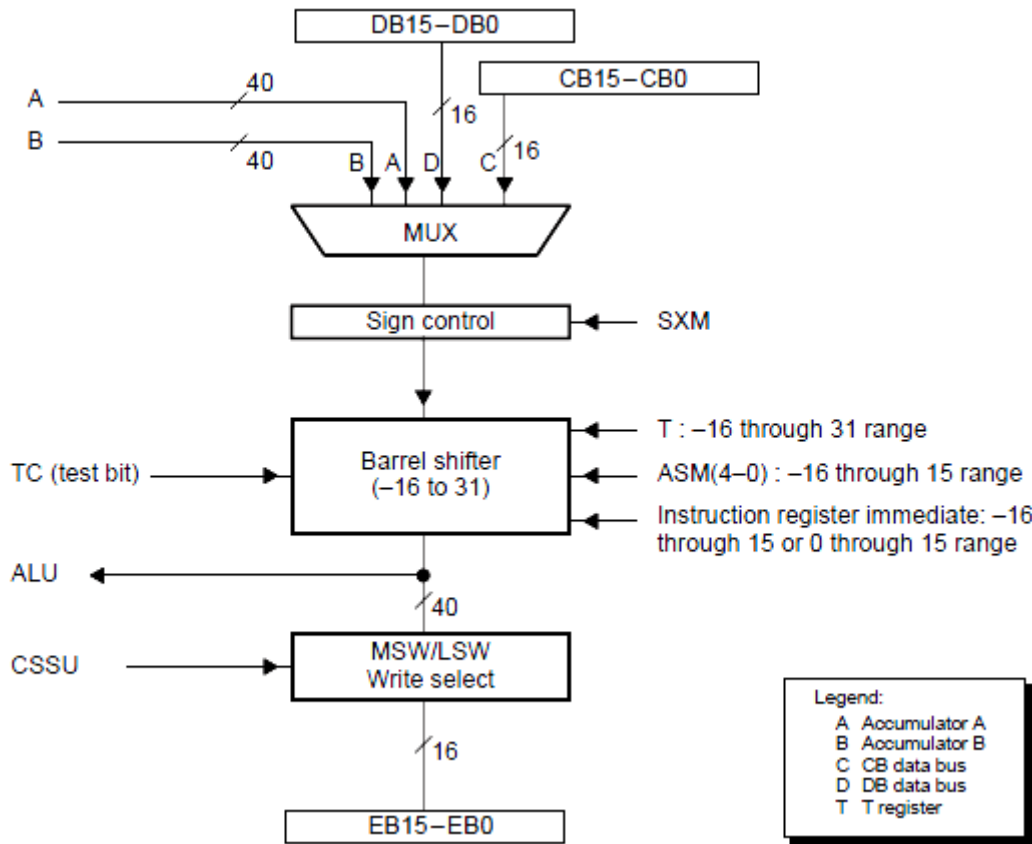


**Figure 3.2.**Functional diagram of the central processing unit of the TMS320C54xx processors.

**Barrel shifter:** provides the capability to scale the data during an operand read or write. No overhead is required to implement the shift needed for the scaling operations. The '54xx barrel shifter can produce a left shift of 0 to 31 bits or a right shift of 0 to 16 bits on the input data. The shift count field of status registers ST1, or in the temporary register T. Figure 3.3 shows the functional diagram of the barrel shifter of TMS320C54xx processors.

The barrel shifter and the exponent encoder normalize the values in an accumulator in a single cycle. The LSBs of the output are filled with 0s, and the MSBs can be either zero filled or sign extended, depending on the state of the sign-extension

mode bit in the status register ST1. An additional shift capability enables the processor to perform numerical scaling, bit extraction, extended arithmetic, and overflow prevention operations.

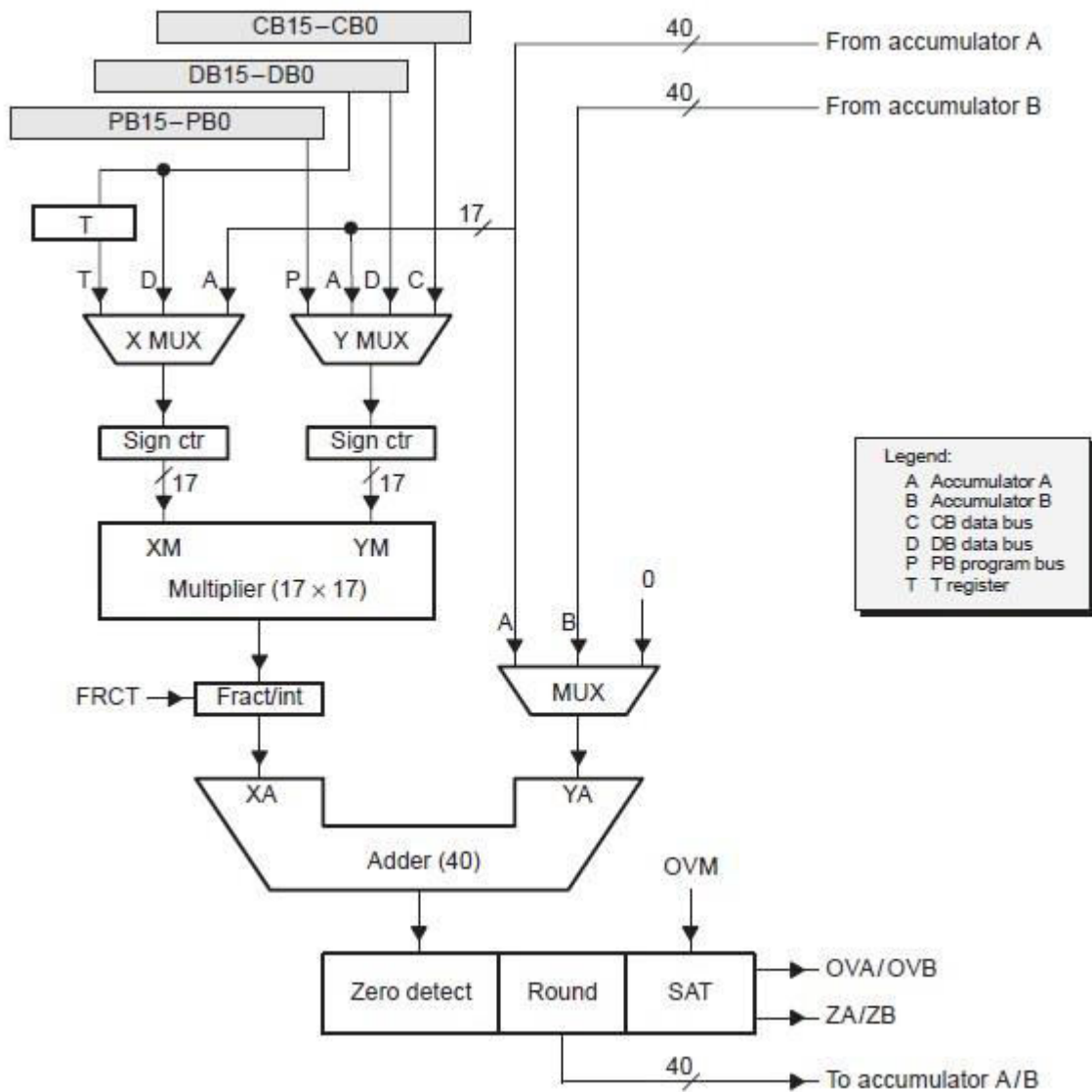


**Figure 3.3.**Functional diagram of the barrel shifter

**Multiplier/adder unit:** The kernel of the DSP device architecture is multiplier/adder unit. The multiplier/adder unit of TMS320C54xx devices performs 17 x 17 2's complement multiplication with a 40-bit addition effectively in a single instruction cycle. In addition to the multiplier and adder, the unit consists of control logic for integer and fractional computations and a 16-bit temporary storage register, T. Figure 3.4 show the functional diagram of the multiplier/adder unit of TMS320C54xx processors.

The compare, select, and store unit (CSSU) is a hardware unit specifically incorporated to accelerate the add/compare/select operation. This operation is essential to implement the *Viterbi* algorithm used in many signal-processing applications.

The exponent encoder unit supports the EXP instructions, which stores in the T register the number of leading redundant bits of the accumulator content. This information is useful while shifting the accumulator content for the purpose of scaling.



**Figure 3.4.** Functional diagram of the multiplier/adder unit of TMS320C54xx processors.

### Internal Memory and Memory-Mapped Registers:

The amount and the types of memory of a processor have direct relevance to the efficiency and performance obtainable in implementations with the processors. The '54xx memory is organized into three individually selectable spaces: program, data, and I/O spaces. All '54xx devices contain both RAM and ROM. RAM can be either dual-access

type (DARAM) or single-access type (SARAM). The on-chip RAM for these processors is organized in pages having 128 word locations on each page.

The '54xx processors have a number of CPU registers to support operand addressing and computations. The CPU registers and peripherals registers are all located on page 0 of the data memory. Figure 3.5(a) and (b) shows the internal CPU registers and peripheral registers with their addresses. The processors mode status (PMST) registers that is used to configure the processor. It is a memory-mapped register located at address 1Dh on page 0 of the RAM.

A part of on-chip ROM may contain a boot loader and look-up tables for function such as sine, cosine,  $\mu$ -law, and A-law.

NAME	DEC	HEX	DESCRIPTION
IMR	0	0	Interrupt mask register
IFR	1	1	Interrupt flag register
—	2–5	2–5	Reserved for testing
ST0	6	6	Status register 0
ST1	7	7	Status register 1
AL	8	8	Accumulator A low word (15–0)
AH	9	9	Accumulator A high word (31–16)
AG	10	A	Accumulator A guard bits (39–32)
BL	11	B	Accumulator B low word (15–0)
BH	12	C	Accumulator B high word (31–16)
BG	13	D	Accumulator B guard bits (39–32)
TREG	14	E	Temporary register
TRN	15	F	Transition register
AR0	16	10	Auxiliary register 0
AR1	17	11	Auxiliary register 1
AR2	18	12	Auxiliary register 2
AR3	19	13	Auxiliary register 3
AR4	20	14	Auxiliary register 4
AR5	21	15	Auxiliary register 5
AR6	22	16	Auxiliary register 6
AR7	23	17	Auxiliary register 7
SP	24	18	Stack pointer register
BK	25	19	Circular buffer size register
BRC	26	1A	Block repeat counter
RSA	27	1B	Block repeat start address
REA	28	1C	Block repeat end address
PMST	29	1D	Processor mode status (PMST) register
XPC	30	1E	Extended program page register
—	31	1F	Reserved

Figure 3.5(a) Internal memory-mapped registers of TMS320C54xx processors.



NAME	ADDRESS		DESCRIPTION
	DEC	HEX	
DRR20	32	20	McBSP 0 Data Receive Register 2
DRR10	33	21	McBSP 0 Data Receive Register 1
DXR20	34	22	McBSP 0 Data Transmit Register 2
DXR10	35	23	McBSP 0 Data Transmit Register 1
TIM	36	24	Timer Register
PRD	37	25	Timer Period Register
TCR	38	26	Timer Control Register
—	39	27	Reserved
SWWSR	40	28	Software Wait-State Register
BSCR	41	29	Bank-Switching Control Register
—	42	2A	Reserved
SWCR	43	2B	Software Wait-State Control Register
HPIC	44	2C	HPI Control Register (HMODE = 0 only)
—	45–47	2D–2F	Reserved
DRR22	48	30	McBSP 2 Data Receive Register 2
DRR12	49	31	McBSP 2 Data Receive Register 1
DXR22	50	32	McBSP 2 Data Transmit Register 2
DXR12	51	33	McBSP 2 Data Transmit Register 1
SPSA2	52	34	McBSP 2 Subbank Address Register
SPSD2	53	35	McBSP 2 Subbank Data Register
—	54–55	36–37	Reserved
SPSA0	56	38	McBSP 0 Subbank Address Register
SPSD0	57	39	McBSP 0 Subbank Data Register
—	58–59	3A–3B	Reserved
GPIOCR	60	3C	General-Purpose I/O Control Register
GPIOSR	61	3D	General-Purpose I/O Status Register
CSIDR	62	3E	Device ID Register
—	63	3F	Reserved
DRR21	64	40	McBSP 1 Data Receive Register 2
DRR11	65	41	McBSP 1 Data Receive Register 1
DXR21	66	42	McBSP 1 Data Transmit Register 2
DXR11	67	43	McBSP 1 Data Transmit Register 1
—	68–71	44–47	Reserved
SPSA1	72	48	McBSP 1 Subbank Address Register
SPSD1	73	49	McBSP 1 Subbank Data Register
—	74–83	4A–53	Reserved
DMPREC	84	54	DMA Priority and Enable Control Register
DMSA	85	55	DMA Subbank Address Register

Figure 3.5(b). peripheral registers for the TMS320C54xx processors

### Status registers (ST0,ST1):

**ST0:** Contains the status of flags (OVA, OVB, C, TC) produced by arithmetic operations & bit manipulations.

**ST1:** Contain the status of various conditions & modes. Bits of ST0&ST1 registers can be set or clear with the SSBX & RSBX instructions.

**PMST:** Contains memory-setup status & control information.

**Status register0 diagram:**

ARP (15-13)	TC (12)	C (11)	OVA (10)	OVB (9)	DP (8-0)
----------------	------------	-----------	-------------	------------	-------------

Figure 3.6(a). ST0 diagram

ARP: Auxiliary register pointer.

TC: Test/control flag.

C: Carry bit.

OVA: Overflow flag for accumulator A.

OVB: Overflow flag for accumulator B.

DP: Data-memory page pointer.

**Status register1 diagram:**

BRAF(15)	CPL (14)	XF (13)	HM (12)	INTM (11)	0 (10)	OV (9)	SXM (8)	C16 (7)	FRCT(6)	CMPT(5)	ASM (4-0)
----------	-------------	------------	------------	--------------	-----------	-----------	------------	------------	---------	---------	--------------

Figure 3.6(b). ST1 diagram

BRAF: Block repeat active flag

BRAF=0, the block repeat is deactivated.

BRAF=1, the block repeat is activated.

CPL: Compiler mode

CPL=0, the relative direct addressing mode using data page pointer is selected.

CPL=1, the relative direct addressing mode using stack pointer is selected.

HM: Hold mode, indicates whether the processor continues internal execution or acknowledge for external interface.

INTM: Interrupt mode, it globally masks or enables all interrupts.

INTM=0 all unmasked interrupts are enabled.

INTM=1 all masked interrupts are disabled.

0: Always read as 0

OV: Overflow mode.

OV=1 the destination accumulator is set either the most positive value or the most negative value.

OV=0 the overflowed result is in destination accumulator.

SXM: Sign extension mode.

SXM=0 € Sign extension is suppressed.

SXM=1 € Data is sign extended

C16: Dual 16 bit/double-Precision arithmetic mode.

C16=0 € ALU operates in double-Precision arithmetic mode.

C16=1 € ALU operates in dual 16-bit arithmetic mode.

FRCT: Fractional mode.

FRCT=1 € the multiplier output is left-shifted by 1 bit to compensate an extra sign bit.

CMPT: Compatibility mode.

CMPT=0 € ARP is not updated in the indirect addressing mode.

CMPT=1 € ARP is updated in the indirect addressing mode.

ASM: Accumulator Shift Mode. 5 bit field, & specifies the Shift value within -16 to 15 range.

### **Processor Mode Status Register (PMST):**

<b>IPTR(15-7)</b>	<b>MP/MC(6)</b>	<b>OVLY(5)</b>	<b>AVIS(4)</b>	<b>DROM(3)</b>	<b>CLKOFF(2)</b>	<b>SMUL(1)</b>	<b>SST(0)</b>
-------------------	-----------------	----------------	----------------	----------------	------------------	----------------	---------------

Figure 3.6(c).PMST register diagram

INTR: Interrupt vector pointer, point to the 128-word program page where the interrupt vectors reside.

MP/MC: Microprocessor/Microcomputer mode,

MP/MC=0, the on chip ROM is enabled.

MP/MC=1, the on chip ROM is enabled.

OVLY: RAM OVERLAY, OVLY enables on chip dual access data RAM blocks to be mapped into program space.

AVIS: It enables/disables the internal program address to be visible at the address pins.

DROM: Data ROM, DROM enables on-chip ROM to be mapped into data space.

CLKOFF: CLOCKOUT off.

SMUL: Saturation on multiplication.

SST: Saturation on store.

### 3.4 Data Addressing Modes of TMS320C54X Processors:

Data addressing modes provide various ways to access operands to execute instructions and place results in the memory or the registers. The 54XX devices offer seven basic addressing modes

1. Immediate addressing.
2. Absolute addressing.
3. Accumulator addressing.
4. Direct addressing.
5. Indirect addressing.
6. Memory mapped addressing
7. Stack addressing.

Immediate addressing:

The instruction contains the specific value of the operand. The operand can be short (3,5,8 or 9 bit in length) or long (16 bits in length). The instruction syntax for short operands occupies one memory location,

Example: LD #20, DP.

RPT #0FFFFh.

Absolute Addressing:

The instruction contains a specified address in the operand.

1. Dmad addressing.

MVDK Smem,dmad

MVDM dmad,MMR

2. Pmad addressing.

MVDP Smem,pmad

MVPD pmem,Smad

3. PA addressing.

PORTR PA, Smem,

4. \*(lk) addressing .

---

Example:

```
MVKP 1000h, *AR5 ; 1000 H *AR5 (dmad addressing)
MVDP 1000h, *AR7 ; 1000h *AR7 (pmad addressing)
PORTR 05h, *AR3 ; 05h *AR3 (PA addressing)
LD *(1000h), A ; *(1000h) A (*lk addressing)
```

Accumulator Addressing:

Accumulator content is used as address to transfer data between Program and Data memory.

Ex: READA \*AR2

Direct Addressing:

Base address + 7 bits of value contained in instruction = 16 bit address. A page of 128 locations can be accessed without change in DP or SP. Compiler mode bit (CPL) in ST1 register is used.

If CPL =0 selects DP

CPL = 1 selects SP,

It should be remembered that when SP is used instead of DP, the effective address is computed by adding the 7-bit offset to SP.

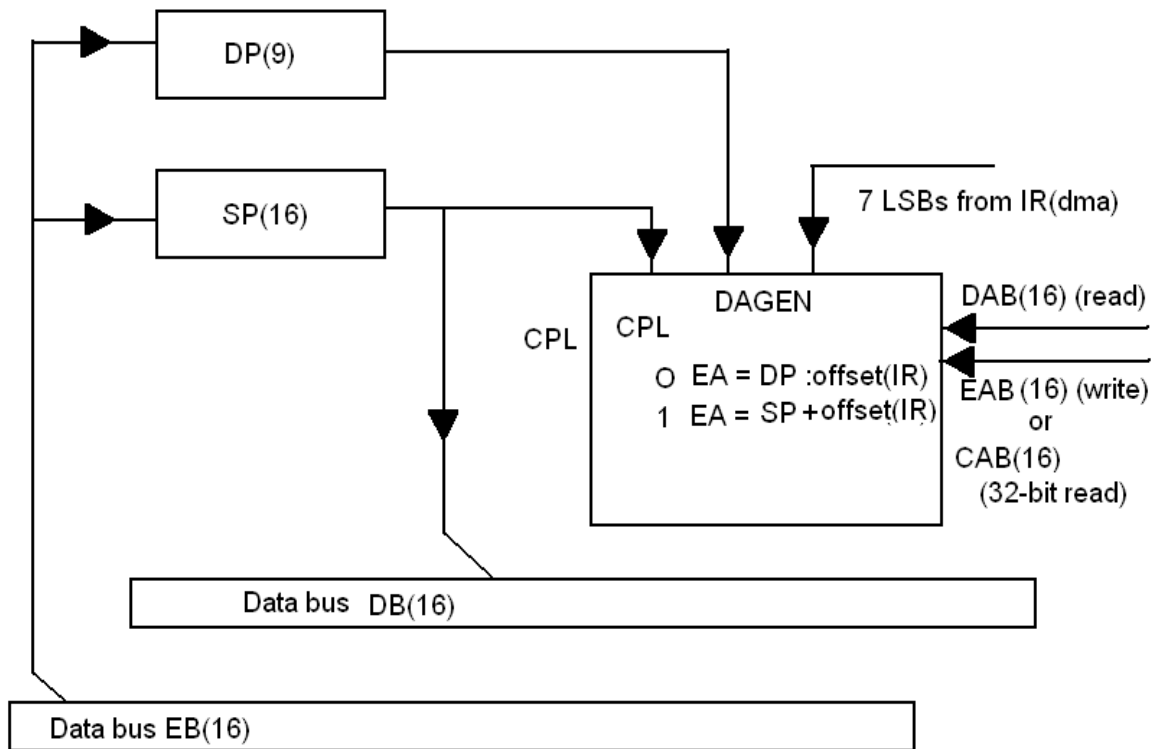


Figure 3.7 Block diagram of the direct addressing mode for TMS320C54xx Processors.

### Indirect Addressing:

- Data space is accessed by address present in an auxiliary register.
- 54xx have 8, 16 bit auxiliary register (AR0 – AR 7). Two auxiliary register arithmetic units (ARAU0 & ARAU1)
- Used to access memory location in fixed step size. AR0 register is used for indexed and bit reverse addressing modes.

- For single – operand addressing

MOD € type of indirect addressing  
ARF € AR used for addressing

- ARP depends on (CMPT) bit in ST1

CMPT = 0, Standard mode, ARP set to zero

CMPT = 1, Compatibility mode, Particularly AR selected by ARP

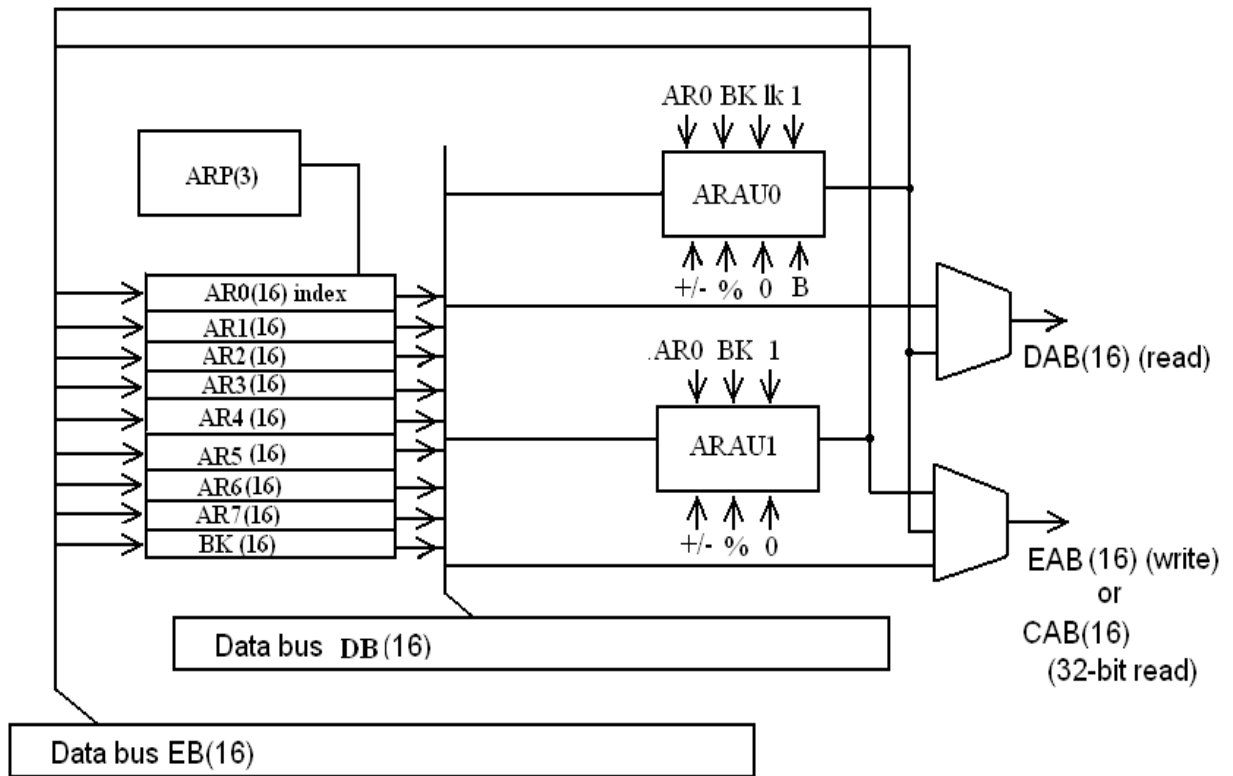


Figure 3.8 Block diagram of the indirect addressing mode for TMS320C54xx Processors.

Operand syntax	Function
*AR <sub>x</sub>	Addr = AR <sub>x</sub> ;
*AR <sub>x</sub> -	Addr = AR <sub>x</sub> ; AR <sub>x</sub> = AR <sub>x</sub> - 1
*AR <sub>x</sub> +	Addr = AR <sub>x</sub> ; AR <sub>x</sub> = AR <sub>x</sub> + 1
*+AR <sub>x</sub>	Addr = AR <sub>x</sub> +1; AR <sub>x</sub> = AR <sub>x</sub> + 1
*AR <sub>x</sub> - 0B	Addr = AR <sub>x</sub> ; AR <sub>x</sub> = B(AR <sub>x</sub> - AR <sub>0</sub> )
*AR <sub>x</sub> - 0	Addr = AR <sub>x</sub> ; AR <sub>x</sub> = AR <sub>x</sub> - AR <sub>0</sub>
*AR <sub>x</sub> + 0	Addr = AR <sub>x</sub> ; AR <sub>x</sub> = AR <sub>x</sub> + AR <sub>0</sub>
*AR <sub>x</sub> + 0B	Addr = AR <sub>x</sub> ; AR <sub>x</sub> = B(AR <sub>x</sub> + AR <sub>0</sub> )
*AR <sub>x</sub> - %	Addr = AR <sub>x</sub> ; AR <sub>x</sub> = circ(AR <sub>x</sub> - 1)

*+AR - 0%	Addr = ARx; ARx = circ(ARx - AR0)
*ARx + %	Addr = ARx ; ARx = circ (ARx + 1)

Table 3.2 Indirect addressing options with a single data –memory operand.  
Circular Addressing;

- Used in convolution, correlation and FIR filters.
- A circular buffer is a sliding window contains most recent data. Circular buffer of size R must start on a N-bit boundary, where  $2^N > R$ .
- The circular buffer size register (BK): specifies the size of circular buffer.
- Effective base address (EFB): By zeroing the N LSBs of a user selected AR (ARx).
- End of buffer address (EOB) : By replacing the N LSBs of ARx with the N LSBs of BK.

If  $0 \leq \text{index} + \text{step} < \text{BK}$  ; index = index +step;  
 else if  $\text{index} + \text{step} \geq \text{BK}$  ; index = index + step - BK;  
 else if  $\text{index} + \text{step} < 0$ ; index + step + BK



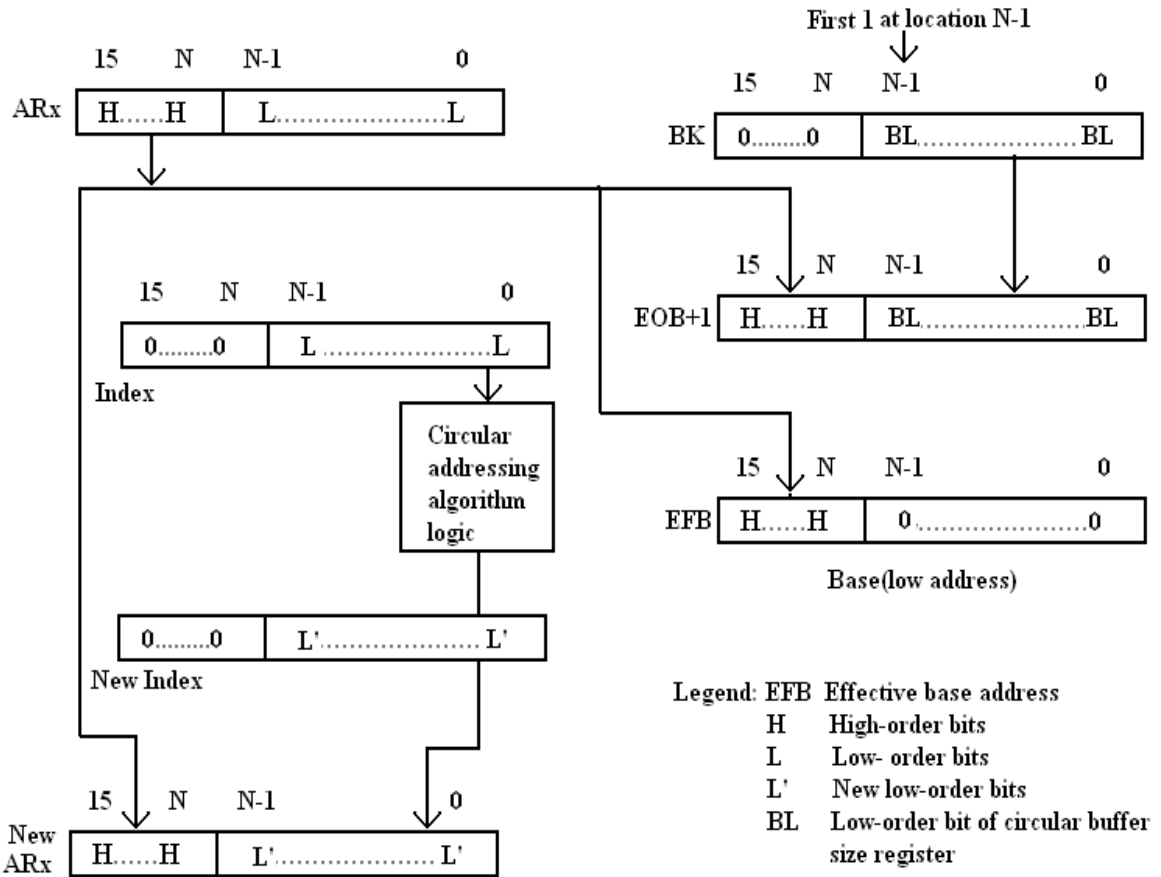


Figure 3.9 Block diagram of the circular addressing mode for TMS320C54xx Processors.

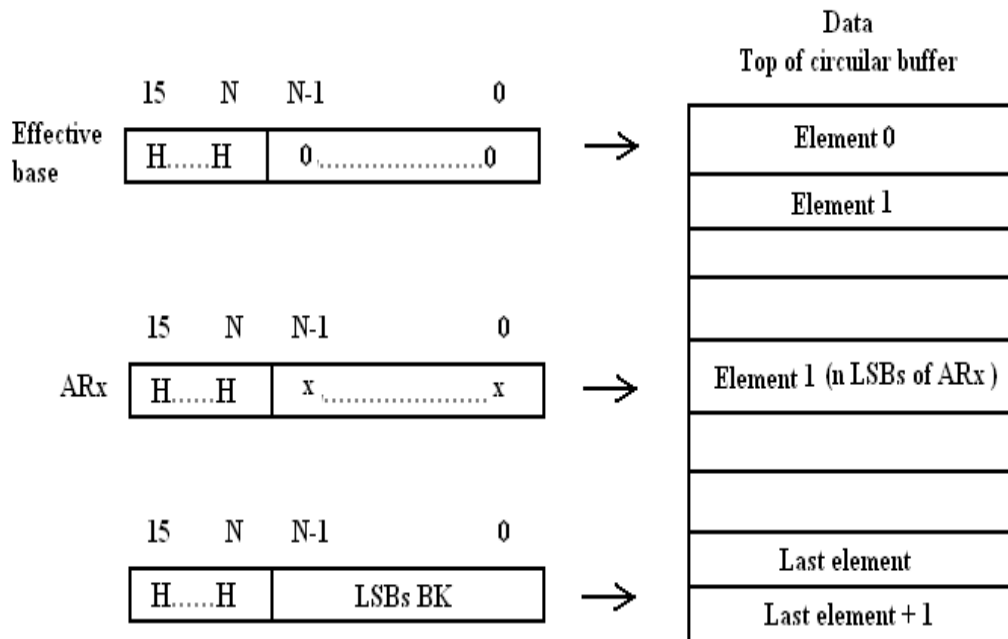


Figure 3.10 circular addressing mode implementation for TMS320C54xx Processors.

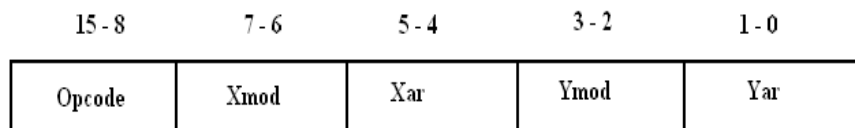
### Bit-Reversed Addressing:

- Used for FFT algorithms.
- AR0 specifies one half of the size of the FFT.
- The value of AR0 =  $2N-1$ : N = integer FFT size =  $2N$
- AR0 + AR (selected register) = bit reverse addressing.
- The carry bit propagating from left to right.

### Dual-Operand Addressing:

Dual data-memory operand addressing is used for instruction that simultaneously perform two reads (32-bit read) or a single read (16-bit read) and a parallel store (16-bit store) indicated by two vertical bars, II. These instructions access operands using indirect addressing mode.

If in an instruction with a parallel store the source operand the destination operand point to the same location, the source is read before writing to the destination. Only 2 bits are available in the instruction code for selecting each auxiliary register in this mode. Thus, just four of the auxiliary registers, AR2-AR5, can be used, The ARAUs together with these registers, provide capability to access two operands in a single cycle. Figure 3.11 shows how an address is generated using dual data-memory operand addressing.



Name	Function
Opcode	This field contains the operation code for the instruction
Xmod	Defined the type of indirect addressing mode used for accessing the Xmem operand
XAR	Xmem AR selection field defines the AR that contains the address of Xmem
Ymod	Defies the type of inderect addressing mode used for accessing the Ymem operand
Yar	Ymem AR selection field defines the AR that contains the address of Ymem

Table 3.3.Function of the different field in dual data memory operand addressing

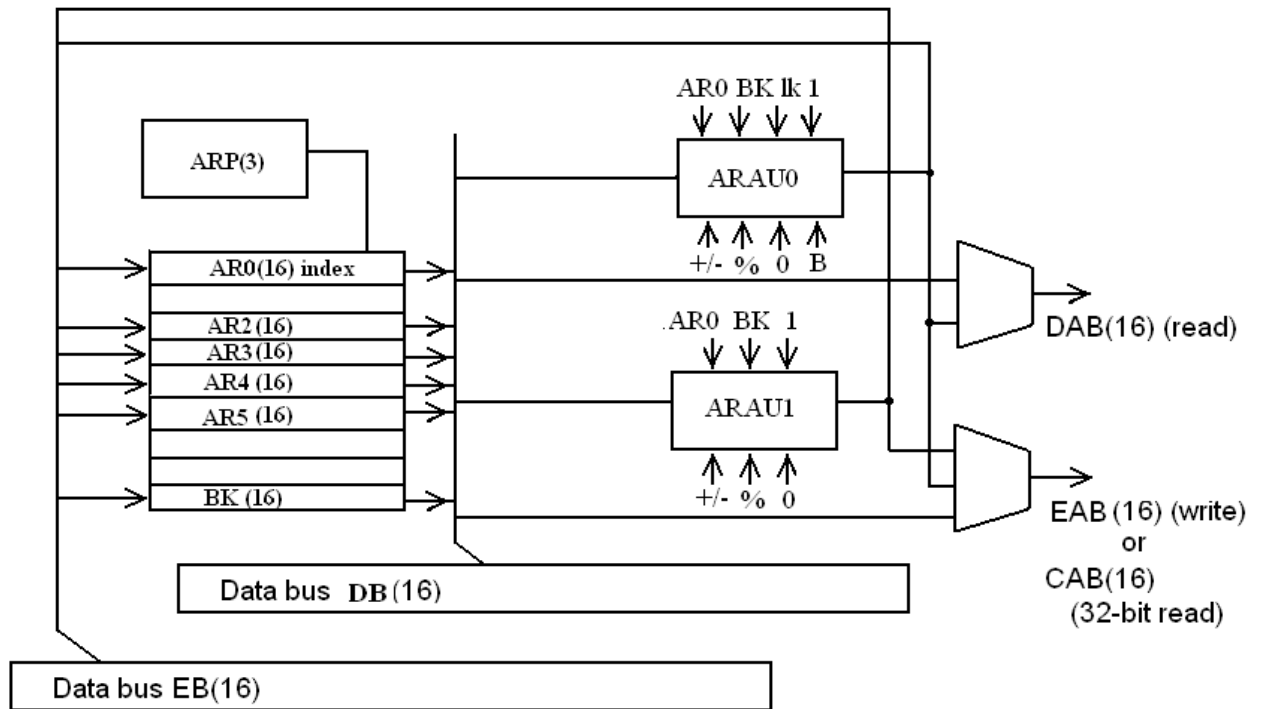


Figure 3.11 Block diagram of the Indirect addressing options with a dual data –memory operand.

### Memory-Mapped Register Addressing:

- Used to modify the memory-mapped registers without affecting the current data-page pointer (DP) or stack-pointer (SP)
  - Overhead for writing to a register is minimal
  - Works for direct and indirect addressing
  - Scratch –pad RAM located on data PAGE0 can be modified
- STM #x, DIRECT
- STM #tbl, AR1

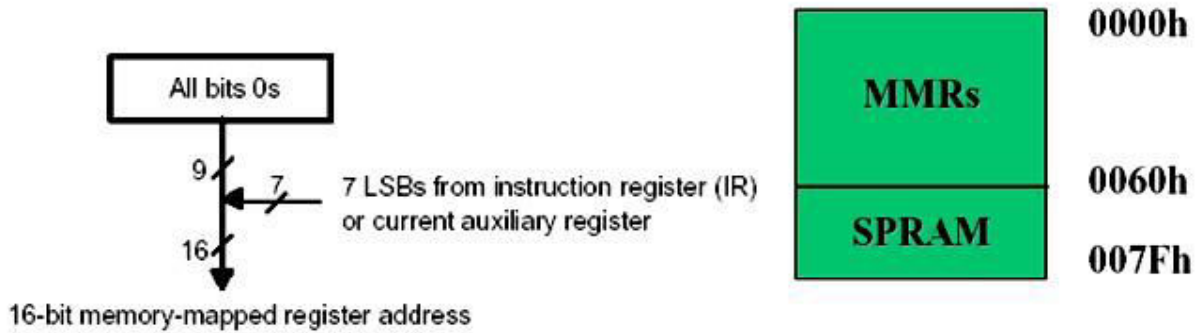


Figure 3.12.16 bit memory mapped register address generation.

### Stack Addressing:

- Used to automatically store the program counter during interrupts and subroutines.
- Can be used to store additional items of context or to pass data values.
- Uses a 16-bit memory-mapped register, the stack pointer (SP).
- PSHD X2

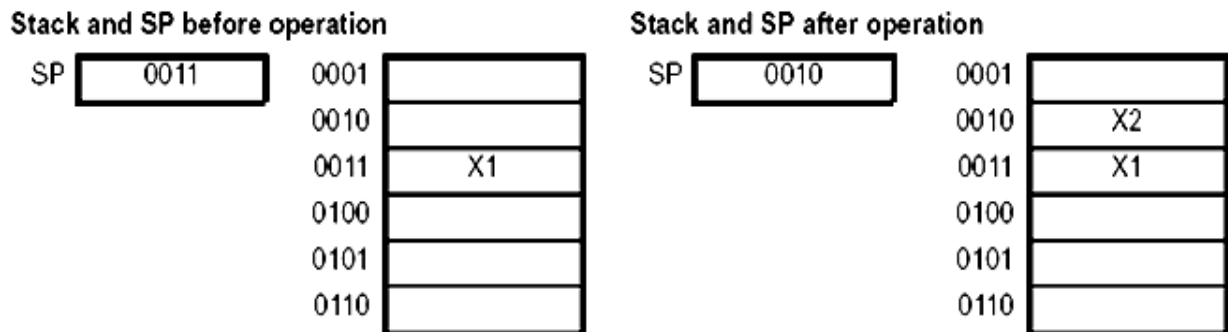


Figure 3.13. Values of stack &SP before and after operation.

### Example Problem

- Assuming the current content of AR3 to be 200h, what will be its contents after each of the following TMS320C54xx addressing modes is used? Assume that the contents of AR0 are 20h.
  - \*AR3+0
  - \*AR3-0

- c. \*AR3+
- d. \*AR3
- e. \*AR3
- f. \*+AR3(40h)
- g. \*+AR3(-40h)

Solution:

- a.  $AR3 \cdot AR3 + AR0$ ;  
 $AR3 = 200h + 20h = 220h$
- b.  $AR3 \cdot AR3 - AR0$ ;  
 $AR3 = 200h - 20h = 1E0h$
- c.  $AR3 \cdot AR3 + 1$ ;  
 $AR3 = 200h + 1 = 201h$
- d.  $AR3 \cdot AR3 - 1$ ;  
 $AR3 = 200h - 1 = 1FFh$
- e. AR3 is not modified.  
 $AR3 = 200h$
- f.  $AR3 \cdot AR3 + 40h$ ;  
 $AR3 = 200 + 40h = 240h$
- g.  $AR3 \cdot AR3 - 40h$ ;  
 $AR3 = 200 - 40h = 1C0h$

2. Assume that the register AR3 with contents 1020h is selected as the pointer for the circular buffer. Let BK = 40h to specify the circular buffer size as 40h. Determine the start and the end addresses for the buffer. What will be the contents of register AR3 after the execution to the instruction LD\*AR3 + 0%, A, if the contents of register AR0 are 0025h?

Solution:

AR3 = 1020h means that currently it points to location 1020h. Masking the lower 6 bits zeros gives the start address of the buffer as 1000h. Replacing the same bits with the BK gives the end address as 1040h.

The Instruction LD\*AR3 + 0%, A modifies AR3 by adding AR0 to it and applying the circular modification. It yields

$$AR3 = \text{circ}(1020h + 0025h) = \text{circ}(1045h) = 1045h - 40h = 1005h.$$

Thus the location 1005h is the one pointed to by AR3.

3. Assuming the current contents of AR3 to be 200h, what will be its contents after each of the following TMS320C54xx addressing modes is used? Assume that the contents of AR0 are 20h

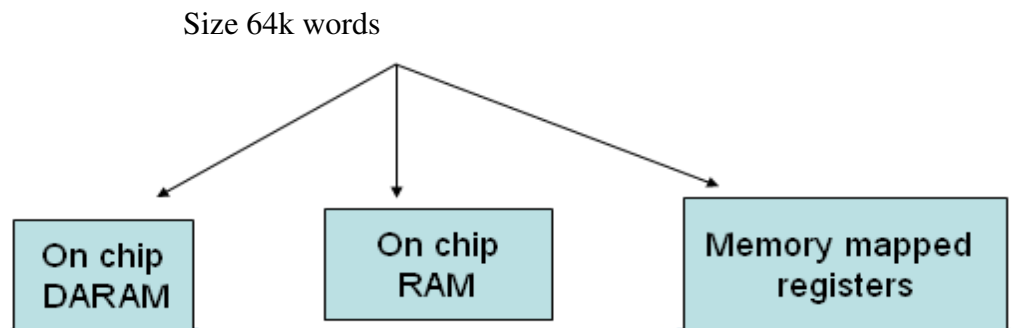
- a. \*AR3 + 0B
- b. \*AR3 - 0B

Solution:

- a. AR3 = AR3 + AR0 with reverse carry propagation;  
AR3 = 200h + 20h (with reverse carry propagation) = 220h.
- b. AR3 = AR3 - AR0 with reverse carry propagation;  
AR3 = 200h - 20h (with reverse carry propagation) = 23Fh.

### Memory Space of TMS320C54xx Processors

- A total of 128k words extendable up to 8192k words.
- Total memory includes RAM, ROM, EPROM, EEPROM or Memory mapped peripherals.
- Data memory: To store data required to run programs & for external memory mapped registers.



- Program memory: To store program instructions & tables used in the execution of programs.

Organized into 128 pages, each of 64k word size

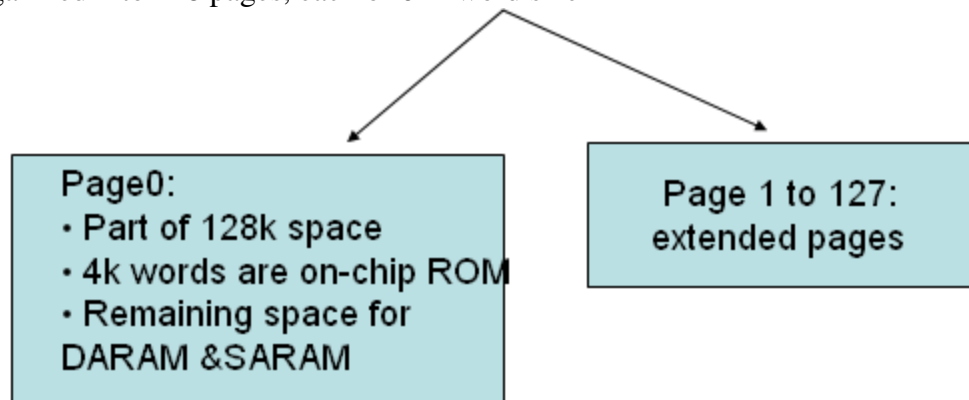


Table 3.4. Function of different pin PMST register

PMST bit	Logic	On-chip memory configuration
MP/MC	0	ROM enabled
	1	ROM not available
OVLY	0	RAM in data space
	1	RAM in program space
DROM	0	ROM not in data space
	1	ROM in data space

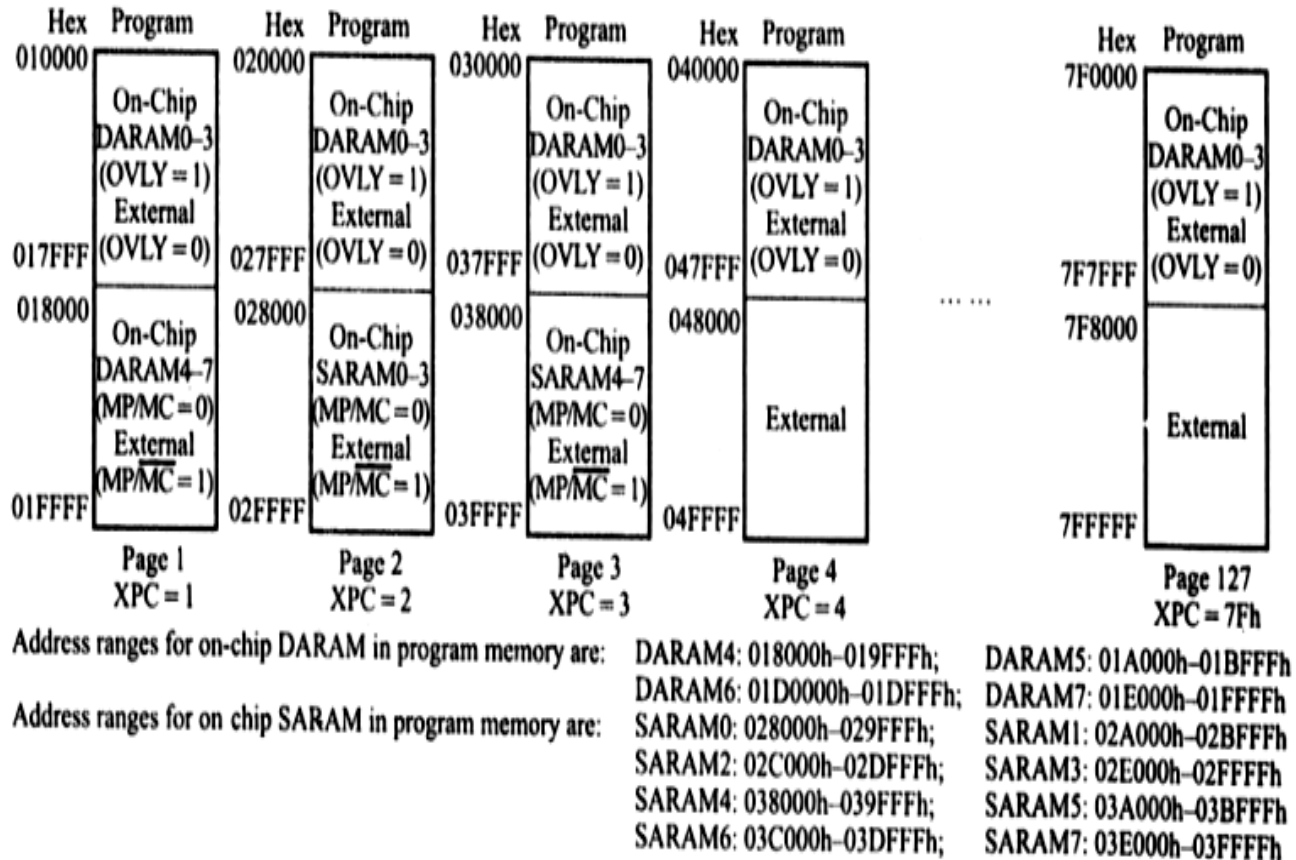
Hex	Page 0 Program	Hex	Page 0 Program	Hex	Data
0000	Reserved (OVLY = 1)	0000	Reserved (OVLY = 1)	0000	Memory-Mapped Registers
007F	External (OVLY = 0)	007F	External (OVLY = 0)	005F	
0080	On-Chip DARAM0-3 (OVLY = 1)	0080	On-Chip DARAM0-3 (OVLY = 1)	0060	Scratch-Pad RAM
7FFF		External (OVLY = 0)		7FFF	
8000	External	8000	External	8000	On-Chip DARAM0-3 (32K × 16-bit)
FF7F		FF7F		On-Chip ROM (16K × 16-bit)	
FF80	Interrupts (External)	FF00	Reserved	FF80	On-Chip DARAM4-7 (DROM = 1) or External (DROM = 0)
FFFF		FFFF		Interrupts (On-Chip)	

$\overline{\text{MP/MC}} = 1$   
(Microprocessor Mode)
 $\overline{\text{MP/MC}} = 0$   
(Microcomputer Mode)

Address ranges for on-chip DARAM in data memory are:

DARAM0: 0080h-1FFFh;	DARAM1: 2000h-3FFFh
DARAM2: 4000h-5FFFh;	DARAM3: 6000h-7FFFh
DARAM4: 8000h-9FFFh;	DARAM5: A000h-BFFFh
DARAM6: C000h-DFFFh;	DARAM7: E000h-FFFFh

(a)



(b)

Figure 3.14 Memory map for the TMS320C5416 Processor.

### Program Control

- ❑ It contains program counter (PC), the program counter related H/W, hard stack, repeat counters & status registers.
- ❑ PC addresses memory in several ways namely:
  - Branch: The PC is loaded with the immediate value following the branch instruction
  - Subroutine call: The PC is loaded with the immediate value following the call instruction
  - Interrupt: The PC is loaded with the address of the appropriate interrupt vector.



---

➤ Instructions such as BACC, CALA, etc ;The PC is loaded with the contents of the accumulator low word

- ❑ End of a block repeat loop: The PC is loaded with the contents of the block repeat program address start register.
- ❑ Return: The PC is loaded from the top of the stack.