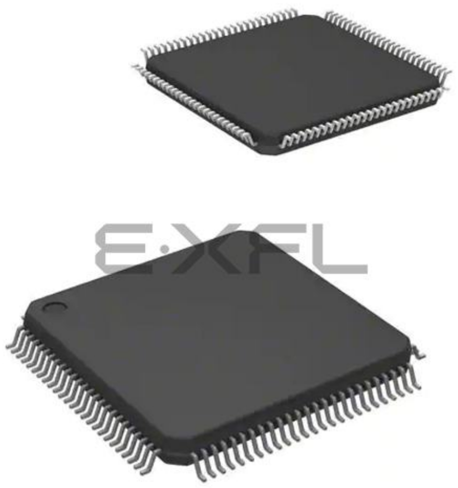

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Discontinued at Digi-Key
Core Processor	C166SV2
Core Size	16-Bit
Speed	20MHz
Connectivity	CANbus, EBI/EMI, SPI, UART/USART
Peripherals	PWM, WDT
Number of I/O	79
Program Memory Size	128KB (128K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	8K x 8
Voltage - Supply (Vcc/Vdd)	2.35V ~ 2.7V
Data Converters	A/D 14x8/10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	100-LQFP
Supplier Device Package	PG-TQFP-100-5
Purchase URL	https://www.e-xfl.com/product-detail/infineon-technologies/saf-xc164cs-16f20f-bb

Edition 2001-01

Published by Infineon Technologies AG,
St.-Martin-Strasse 53,
D-81541 München, Germany

2.5.5 The System Stack

The C166S V2 CPU supports a system stack of 64 kBytes. The stack can be located internally in one of the on-chip memories or externally. The 16-bit Stack Pointer (SP) register addresses the stack within a 64 kByte segment. The Stack Pointer Segment Register (SPSG) selects the segment in which the stack is located. A virtual stack (usually bigger then 64 kBytes) can be implemented by software. This mechanism is supported by registers STKOV and STKUN (see descriptions below).

The Stack Pointer Register SP

The non-bit addressable Stack Pointer SP register is used to point to the top of the system stack (TOS). The SP register is pre-decremented whenever data is to be pushed onto the stack, and it is post-incremented whenever data is to be popped from the stack. Therefore, the system stack grows from higher toward lower memory locations.

The SP register can be updated via any instruction capable of modifying an 16-bit SFR.

Note: Due to the internal instruction pipeline, a stack pointer initialization stalls the instruction flow until the operation is finished. A POP and RETURN instruction can immediately follow an instruction updating the SP.

SP
Stack Pointer **SFR** **Reset Value: $FC00_H$**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SP															**0**
rwh															r

Field	Bits	Type	Description
SP	[15:1]	rwh	**Modifiable portion of register SP** Specifies the top of the system stack.
0	[0]	r	Fixed to 0

number (MAH has 1 in the most significant bit), the MAE will be loaded with ones, representing the extended 40-bit negative number in 2s compliment notation. One may see that the extended 40-bit value is equal to 32-bit value without extension. In other words, after this extension, MAE does not contain significant bits. Generally, this condition is present when the highest 9 bits of the 40-bit signed result are the same.

During the accumulator operations, an overflow may happen and the result may not fit into 32-bits and the MAE will change. The extension flag "E", which is the part of the most significant byte of MSW, is set when the signed result in the accumulator has overflowed the 32-bit boundary. This condition is present when the highest 9 bits of the 40-bit signed result are not the same, i.e. MAE contains significant bits.

Most CoXXX operations specify the 40-bit accumulator register as a source and/or a destination operand.

The MAC Unit Accumulator Extension Byte MAE

The MAE register is a part of the 40-bit MAC unit accumulator register. MAE is accessed as the Least Significant Byte of **MSW**. It is implicitly used by the MAC unit for MAC operation. In case a word operand is written into MAH, the MAE register becomes sign-extended. It can be accessed via any instruction capable of accessing an SFR.

MSW
MAC Status Word **SFRb** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	MV	MSL	ME	MSV	MC	MZ	MN	MAE							
r								rwh							

Field	Bits	Type	Description
MAE	[7:0]	rwh	The most significant bits of the 40-bit Accumulator

The MAC Unit Accumulator High Word MAH

The MAH register is a part of the 40-bit MAC unit accumulator register. It is implicitly used by the MAC unit for MAC operation. In case the word operand is written into MAH, MAL acquires the zero value and the MAE register becomes sign-extended. It can be accessed via any instruction capable of accessing an SFR.

```
In-1   ........
In     ADD    R0,R1
In+1   MOV    R3,[R0]
In+2   ADD    R6,R0
In+3   ADD    R6,R1
In+4   ........
```

	T_n	T_{n+1}	T_{n+2}	T_{n+3}	T_{n+4}	T_{n+5}
DECODE	I_n= ADD R0,R1	I_{n+1}= MOV R3,[R0]	I_{n+2}	I_{n+2}	I_{n+2}	I_{n+3}
ADDRESS	I_{n-1}	I_n= ADD R0,R1	I_{n+1}= MOV R3,[R0]	I_{n+1}= MOV R3,[R0]	I_{n+1}= MOV R3,[R0]	I_{n+2}
MEMORY	I_{n-2}	I_{n-1}	I_n= ADD R0,R1			I_{n+1}= MOV R3,[R0]
EXECUTE	I_{n-3}	I_{n-2}	I_{n-1}	I_n= ADD R0,R1		
WRITE BACK	I_{n-4}	I_{n-3}	I_{n-2}	I_{n-1}	I_n= ADD R0,R1	

To avoid stalls, one multicycle or two single cycle instructions may be inserted. These instructions must not update the GPR used for indirect addressing.

```
In-1   ........
In     ADD    R0,R1
In+1   ADD    R6,R0
In+2   ADD    R6,R1
In+3   MOV    R3,[R0]
In+4   ........
```

	T_n	T_{n+1}	T_{n+2}	T_{n+3}	T_{n+4}	T_{n+5}
DECODE	I_n= ADD R0,R1	I_{n+1}= ADD R6,R0	I_{n+2}= ADD R6,R1	I_{n+3}= MOV R3,[R0]	I_{n+4}	I_{n+5}
ADDRESS	I_{n-1}	I_n= ADD R0,R1	I_{n+1}= ADD R6,R0	I_{n+2}= ADD R6,R1	I_{n+3}= MOV R3,[R0]	I_{n+4}
MEMORY	I_{n-2}	I_{n-1}	I_n= ADD R0,R1	I_{n+1}= ADD R6,R0	I_{n+2}= ADD R6,R1	I_{n+3}= MOV R3,[R0]
EXECUTE	I_{n-3}	I_{n-2}	I_{n-1}	I_n= ADD R0,R1	I_{n+1}= ADD R6,R0	I_{n+2}= ADD R6,R1
WRITE BACK	I_{n-4}	I_{n-3}	I_{n-2}	I_{n-1}	I_n= ADD R0,R1	I_{n+1}= ADD R6,R0

4.1.2 Indirect Addressing Modes

In the case of read accesses using indirect addressing modes, the Address Generation Unit uses a speculative addressing mechanism. The read data path to one of the different memory areas (DPRAM, Internal SRAM, etc.) is selected according to a history table before the address is decoded. This history table has one entry for each of the updated in the Execute Stage and is not used for control purposes in the previous stages. CPUID, ONES, and ZEROS are not changeable at all.

I_{n-1}	....	
I_n	MOV	MCW,#16
I_{n+1}	ADD	R6,R0
I_{n+2}	ADD	R6,R1
I_{n+3}	MOV	R3,[R0]
I_{n+4}	....	

	T_n	T_{n+1}	T_{n+2}	T_{n+3}	T_{n+4}	T_{n+5}
DECODE	I_n= MOV MCW,#16	I_{n+1}= ADD R6,R0	I_{n+2}= ADD R6,R1	I_{n+3}= MOV R3,[R0]	I_{n+4}	I_{n+5}
ADDRESS	I_{n-1}	I_n= MOV MCW,#16	I_{n+1}= ADD R6,R0	I_{n+2}= ADD R6,R1	I_{n+3}= MOV R3,[R0]	I_{n+4}
MEMORY	I_{n-2}	I_{n-1}	I_n= MOV MCW,#16	I_{n+1}= ADD R6,R0	I_{n+2}= ADD R6,R1	I_{n+3}= MOV R3,[R0]
EXECUTE	I_{n-3}	I_{n-2}	I_{n-1}	I_n= MOV MCW,#16	I_{n+1}= ADD R6,R0	I_{n+2}= ADD R6,R1
WRITE BACK	I_{n-4}	I_{n-3}	I_{n-2}	I_{n-1}	I_n= MOV MCW,#16	I_{n+1}= ADD R6,R0

respective control registers. The two upper bits of the interrupt priority level are set to '11_B', which limits the allowed interrupt priority level to be greater than or equal to 12.

FINT0CSP
Fast Interrupt Control Register 0 XSFR Reset Value: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN	0	0	GPX	ILVL		GLVL		SEG							
rw	r	r	rw	rw		rw		rw							

FINT1CSP
Fast Interrupt Control Register 1 XSFR Reset Value: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN	0	0	GPX	ILVL		GLVL		SEG							
rw	r	r	rw	rw		rw		rw							

Field	Bits	Type	Description
EN	[15]	rw	**Fast Interrupt Enable** 0 The interrupt jump table cache is disabled. No fast interrupt is used. 1 The interrupt jump table cache is enabled. A fast interrupt (direct jump to the interrupt service routine) is used instead of the normal fetch from the interrupt vector table.
GPX	[12]	rw	**Group Priority Extension** This bit enables group extension for fast interrupts. (hardwired to 0 for fewer than 64 interrupt nodes)
ILVL	[11:10]	rw	**Interrupt Priority Level** This bit field selects the lower two bits of the interrupt priority level associated with this interrupt jump table cache entry. *Note: The two upper bits of the interrupt priority level are set to '11_B', which ends in an interrupt priority level greater than or equal to 12.*

7 Instruction Set

7.1 Short Instruction Summary

The following compressed cross-reference tables quickly identify specific instructions and provide basic information about them Two ordering schemes are included:

The first table (two pages) is a compressed cross-reference table that quickly associates specific hexadecimal opcodes with the corresponding mnemonics.

The second table lists instructions by their mnemonic and identifies the addressing modes that may be used with the specific instructions and indicates the instruction length for the selected addressing mode. This reference helps to optimize instruction sequences in terms of code size and/or execution time.

Description Levels

In the following sections the instructions are compiled according to different criteria in order to provide different levels of precision:

- **Cross Reference Tables** summarize all instructions in condensed tables
- **The Instruction Set Summary** groups the individual instructions into functional groups
- **The Opcode Table** references the instructions by their hexadecimal opcode

Notes on CoXXX instructions

All CoXXX instructions have a 3-bit wide extended control field 'rrr' in the operand field to control the MRW repeat counter. It is located within the CoXXX instructions at bit positions [31:29].

– '000' -> regular CoXXX instruction.
– '001' -> RESERVED
– '010' -> '- USR0 CoXXX' instruction.
– '011' -> '- USR1 CoXXX' instruction.
– '1xx' -> RESERVED.

Notes on CoXXX instructions using indirect addressing modes

These CoXXX instructions have extended control fields in the operand field to specify the special indirect addressing mode.

Bitfield 'X' is 4-bits wide and is located within CoXXX instructions at bit positions [15:12]. Bit [15] specifies one of the two IDX address pointers; the bitfield [14:12] specifies the operation concerning the IDX pointer.

Bit[15]:

– '0' -> IDX0
– '1' -> IDX1

Bitfield[14:12]

– '000' -> RESERVED
– '001' -> no-operation
– '010' -> IDX +2
– '011' -> IDX -2
– '100' -> IDX + QX0
– '101' -> IDX - QX0
– '110' -> IDX + QX1
– '111' -> IDX - QX1

Bitfield 'qqq' is 3-bits wide and is located within CoXXX instructions at bit positions [26:24]. It specifies the operation concerning the Rw pointer.

Bitfield[26:24]

– '000' -> RESERVED
– '001' -> no-operation
– '010' -> Rw +2
– '011' -> Rw -2
– '100' -> Rw + QR0
– '101' -> Rw - QR0
– '110' -> Rw + QR1
– '111' -> Rw - QR1

ÿ	(opX)	is	logically **COMPLEMENTED**

Parentheses indicate a method of addressing the used operand as follows:

opX	Specifies the immediate constant value of opX
(opX)	Specifies the contents of opX
(opX[n])	Specifies the contents of bit n of opX
((opX))	Specifies the contents of the contents of opX (ie. opX is used as pointer to the actual operand)

The following operands notation will also be used in the operational description:

CP	Context Pointer
CSP	Code Segment Pointer
IP	Instruction Pointer
MD	Multiply/Divide register (32 bits wide, consists of MDH and MDL)
MDL, MDH	Multiply/Divide Low and High registers (each 16 bit wide)
ACC	Accumulator (40 bits wide, consists of MAE, MAH and MDL)
MAH, MAL	Accumulator Low and High registers (each 16 bits wide)
MAE	Accumulator extension register (one byte wide)
PSW	Program Status Word
SP	System Stack Pointer
CPUCON1	CPU Configuration register
C	Carry condition flag in the PSW register
V	Overflow condition flag in the PSW register
SGTDIS	Segmentation Disable bit in the SYSCON register
count	Temporary variable for an intermediate storage of the number of shift or rotate cycles which remain to complete the shift or rotate operation
tmp	Temporary variable for an intermediate result
0, 1, 2,...	Constant values due to the data format of the specified operation

Data Types: This part specifies the particular data type according to the instruction. Basically, the following data types are possible:

BIT, BYTE, WORD, DOUBLEWORD, ACC = 40-bit signed value

Only CoXXX instructions and instructions which extend byte data to word data can change the data type. Note that the data types mentioned in this subsection do not cover accesses to indirect address pointers or to the system stack. These accesses are always performed with word data. Moreover, no data type is specified for System Control Instructions and for those branch instructions which do not access any explicitly addressed data.

- **Description:** This part provides a brief description of the action that is executed by the respective instruction.
- **Condition Code:** The Condition code indicates that the respective instruction is executed if the specified condition exists, and is skipped if it does not. The table below summarizes the sixteen possible condition codes that can be used within Call and Branch instructions. The table shows the abbreviations, the test that is executed for a specific condition, and a 4/5-bit number associated with condition code.

Condition Code Mnemonic cc	Test	Description	Condition Code Number c	Condition Code Number d
cc_UC	1 = 1	Unconditional	0_H	0_H
cc_Z	Z = 1	Zero	2_H	4_H
cc_NZ	Z = 0	Not zero	3_H	6_H
cc_V	V = 1	Overflow	4_H	8_H
cc_NV	V = 0	No overflow	5_H	A_H
cc_N	N = 1	Negative	6_H	C_H
cc_NN	N = 0	Not negative	7_H	E_H
cc_C	C = 1	Carry	8_H	10_H
cc_NC	C = 0	No carry	9_H	12_H
cc_EQ	Z = 1	Equal	2_H	4_H
cc_NE	Z = 0	Not equal	3_H	6_H
cc_ULT	C = 1	Unsigned less than	8_H	10_H
cc_ULE	$(Z \vee C) = 1$	Unsigned less than or equal	F_H	$1E_H$
cc_UGE	C = 0	Unsigned greater than or equal	9_H	12_H
cc_UGT	$(Z \vee C) = 0$	Unsigned greater than	E_H	$1C_H$

BAND — Bit Logical AND — BAND

Group Boolean Bit Manipulation Instructions

Syntax **BAND op1, op2**

Source Operand(s) op1, op2 $\rightarrow$ BIT

Destination Operand(s) op1 $\rightarrow$ BIT

Operation

$$(op1) \leftarrow (op1) \wedge (op2)$$

Description

Performs a single bit logical AND of the source bit specified by op2 and the destination bit specified by op1. The result is then stored in op1.

CPU Flags

E	Z	V	C	N
0	NOR	OR	AND	XOR

E Always cleared.
Z Contains the logical NOR of the two specified bits.
V Contains the logical OR of the two specified bits.
C Contains the logical AND of the two specified bits.
N Contains the logical XOR of the two specified bits.

Encoding

Mnemonic		Format	Bytes
BAND	$bitaddr_{Z.z}$, $bitaddr_{Q.q}$	6A QQ ZZ qz	4

BCMP — Bit to Bit Compare — BCMP

Group Boolean Bit Manipulation Instructions

Syntax **BCMP op1, op2**

Source Operand(s) op1, op2 $\rightarrow$ BIT

Destination Operand(s) none

Operation

$$(op1) \Leftrightarrow (op2)$$

Description

Performs a single bit comparison of the source bit specified by op1 and the source bit specified by op2. No result is written by this instruction. Only the flags are updated.

CPU Flags

E	Z	V	C	N
0	NOR	OR	AND	XOR

E	Always cleared.
Z	Contains the logical NOR of the two specified bits.
V	Contains the logical OR of the two specified bits.
C	Contains the logical AND of the two specified bits.
N	Contains the logical XOR of the two specified bits.

Encoding

Mnemonic		Format	Bytes
BCMP	$bitaddr_{Z.z}$, $bitaddr_{Q.q}$	2A QQ ZZ qz	4

EXTR Begin EXTended Register Sequence EXTR

Group System Control Instructions

Syntax **EXTR op1**

Source Operand(s) op1 $\rightarrow$ 2-bit instruction counter

Destination Operand(s) none

Operation

(count) $\leftarrow$ (op1) [$1 \leq op1 \leq 4$]
Disable interrupts and Class A traps
SFR_range $\leftarrow$ Extended
DO WHILE ((count) $\neq$ 0 AND Class_B_Trap_Condition $\neq$ TRUE)
    Next Instruction
    (count) $\leftarrow$ (count) - 1
END WHILE
(count) $\leftarrow$ 0
SFR_range $\leftarrow$ Standard
Enable interrupts and traps

Description

Causes all SFR or SFR bit accesses via the 'reg', 'bitoff' or 'bitaddr' addressing modes being made to the Extended SFR space for a specified number of instructions. During their execution, both standard and PEC interrupts and class A hardware traps are locked. The value of op1 defines the length of the affected instruction sequence.

CPU Flags

E	Z	V	C	N
-	-	-	-	-

E Not affected.
Z Not affected.
V Not affected.
C Not affected.
N Not affected.

Encoding

Mnemonic		Format	Bytes
EXTR	#irang2	D1 :10##-0	2

NEGB Integer Two's Complement NEGB

Group Arithmetic Instructions

Syntax **NEGB op1**

Source Operand(s) op1 → BYTE

Destination Operand(s) op1 → BYTE

Operation

(op1) ← 0 - (op1)

Description

Performs a binary 2s complement of the source operand specified by op1. The result is then stored in op1.

CPU Flags

E	Z	V	C	N
*	*	*	*	*

- E Set if the value of op1 represents the lowest possible negative number. Cleared otherwise. Used to signal the end of a table.
- Z Set if result equals zero. Cleared otherwise.
- V Set if an arithmetic underflow occurred, i.e. the result cannot be represented in the byte data type. Cleared otherwise.
- C Set if a borrow is generated. Cleared otherwise.
- N Set if the most significant bit of the result is set. Cleared otherwise.

Encoding

Mnemonic		Format	Bytes
NEGB	Rb_n	A1 n0	2

8.2 DSP Instruction Set

Encoding

Mnemonic		Format	Bytes
CoMACM-	[IDXi*], [Rw_m*]	93 Xm E8 rrr0:0qqq	4

CoMACR Multiply-Accumulate & Round CoMACR

Group Multiply/Multiply-Accumulate Instructions

Syntax **CoMACR op1, op2, rnd**

Source Operand(s) op1, op2 → WORD

Destination Operand(s) ACC → 40-bit signed value

Operation

IF (MP = 1) THEN
(tmp) ← ((op1) * (op2)) <<1
(ACC) ← (tmp) - (ACC) + 00 0000 8000h
ELSE
(tmp) ← (op1) * (op2)
(ACC) ← (tmp) - (ACC) + 00 0000 8000h
END IF
(MAL) ← 0

Description
Multiplies the two signed 16-bit source operands op1 and op2. The resulting signed 32-bit product is first sign-extended; then, if the MP flag is set, it is one-bit left shifted; then, the 40-bit ACC register contents are subtracted from the result. Finally, the result is 2s complement rounded before being stored in the 40-bit ACC register. The MAL register is cleared.

MAC Flags

MV	MSL	ME	MSV	MC	MZ	MN	Sat.
*	*	*	*	*	*	*	yes

MV Set if an arithmetic underflow occurred, i.e. the result cannot be represented in the 40-bit data type. Cleared otherwise.
MSL Set if the contents of ACC is automatically saturated. Not affected otherwise.
ME Set if the MAE is used. Cleared otherwise.
MSV Set if an arithmetic underflow occurred. Not affected otherwise.
MC Set if a borrow is generated. Cleared otherwise.
MZ Set if result equals zero. Cleared otherwise.
MN Set if the most significant bit of the result is set. Cleared otherwise.

CoMACRu Unsigned Multiply-Accumulate CoMACRu

Group Multiply/Multiply-Accumulate Instructions

Syntax **CoMACRu op1, op2**

Source Operand(s) op1, op2 → WORD

Destination Operand(s) ACC → 40-bit signed value

Operation

(tmp) ← (op1) * (op2)
(ACC) ← (tmp) - (ACC)

Description

Multiplies the two unsigned 16-bit source operands op1 and op2. The resulting unsigned 32-bit product is first zero-extended and then the 40-bit ACC register contents are subtracted from the result before being stored in the 40-bit ACC register.

MAC Flags

MV	MSL	ME	MSV	MC	MZ	MN	Sat.
*	*	*	*	*	*	*	yes

- MV Set if an arithmetic underflow occurred, i.e. the result cannot be represented in the 40-bit data type. Cleared otherwise.
- MSL Set if the contents of ACC is automatically saturated. Not affected otherwise.
- ME Set if the MAE is used. Cleared otherwise.
- MSV Set if an arithmetic underflow occurred. Not affected otherwise.
- MC Set if a borrow is generated. Cleared otherwise.
- MZ Set if result equals zero. Cleared otherwise.
- MN Set if the most significant bit of the result is set. Cleared otherwise.

Encoding

Mnemonic		Format	Bytes
CoMACRu	Rw_n , Rw_m	A3 nm 30 rrr0:0000	4
CoMACRu	Rw_n , [Rw_m*]	83 nm 30 rrr0:0qqq	4
CoMACRu	[IDXi*] , [Rw_m*]	93 Xm 30 rrr0:0qqq	4

CoMUL Signed Multiply with Round CoMUL

Group Multiply/Multiply-Accumulate Instructions

Syntax **CoMUL op1, op2, rnd**

Source Operand(s) op1, op2 $\rightarrow$ WORD

Destination Operand(s) ACC $\rightarrow$ 40-bit signed value

Operation

```
IF (MP = 1) THEN
        (ACC) ← ((op1) * (op2)) <<1 + 00 0000 8000h
ELSE
        (ACC) ← (op1) * (op2) + 00 0000 8000h
END IF
(MAL) ← 0
```

Description

Multiplies the two signed 16-bit source operands op1 and op2. The resulting signed 32-bit product is first sign-extended; then, if the MP flag is set, it is one-bit left shifted. Finally, the result is 2s complement rounded before being stored in the 40-bit ACC register. The MAL register is cleared.

MAC Flags

MV	MSL	ME	MSV	MC	MZ	MN	Sat.
0	*	*	-	0	*	*	yes

MV Always cleared.

MSL Not affected when MP or MS are cleared, otherwise, only set in case of 8000h by 8000h multiplication.

ME Set when MP is set and MS is cleared and in case of 8000h by 8000h multiplication. Cleared otherwise.

MSV Not affected.

MC Always cleared.

MZ Set if result equals zero. Cleared otherwise.

MN Set if the most significant bit of the result is set. Cleared otherwise.

CoSUBR Subtract CoSUBR

Group Arithmetic Instructions

Syntax **CoSUBR op1, op2**

Source Operand(s) op1, op2 $\rightarrow$ WORD

Destination Operand(s) ACC $\rightarrow$ 40-bit signed value

Operation

(tmp) $\leftarrow$ (op2) II (op1)
(ACC) $\leftarrow$ (tmp) - (ACC)

Description
Subtracts the 40-bit ACC contents from a 40-bit operand and stores the result in the ACC register. The 40-bit operand is a sign-extended result of the concatenation of the two source operands op1 (LSW) and op2 (MSW).

MAC Flags

MV	MSL	ME	MSV	MC	MZ	MN	Sat.
*	*	*	*	*	*	*	yes

MV Set if an arithmetic underflow occurred, i.e. the result cannot be represented in the 40-bit data type. Cleared otherwise.
MSL Set if the contents of ACC is automatically saturated. Not affected otherwise.
ME Set if the MAE is used. Cleared otherwise.
MSV Set if an arithmetic underflow occurred. Not affected otherwise.
MC Set if a borrow is generated. Cleared otherwise.
MZ Set if result equals zero. Cleared otherwise.
MN Set if the most significant bit of the result is set. Cleared otherwise.

Encoding

Mnemonic		Format	Bytes
CoSUBR	Rw_n , Rw_m	A3 nm 12 rrr0:0000	4
CoSUBR	Rw_n , [Rw_m*]	83 nm 12 rrr0:0qqq	4
CoSUBR	[IDXi*] , [Rw_m*]	93 Xm 12 rrr0:0qqq	4