Lecture 13
Hardware Multiplier, Flash Programming & TLV Structure

Texas Instruments Incorporated
University of Beira Interior (PT)

Pedro Dinis Gaspar, António Espírito Santo, Bruno Ribeiro, Humberto Santos
University of Beira Interior, Electromechanical Engineering Department
www.msp430.ubi.pt

>> Contents

Contents (1/3)

Contents (2/3)

Contents (3/3)

Hardware Multiplier Introduction (1/3)

- **Several devices in the MSP430 family contain a hardware multiplier peripheral module:**
  - 54xx;
  - FG46xx;
  - FE42x(A);
  - F47xx; F44X; F42x(A);
  - F261x; F24x(x);
  - F16x(x).

- **The MSP430FG4618 (Experimenter's board) supports multiplications using the Hardware Multiplier module, without interfering with CPU activities.**

Hardware Multiplier Introduction (2/3)

- **The hardware multiplier supports:**
  - Unsigned multiply (MPY);
  - Signed multiply (MPYS);
  - Unsigned multiply accumulate (MAC);
  - Signed multiply accumulate (MACS).

- **The multiplication operation can be:**
  - 16×16 bits;
  - 16×8 bits;
  - 8×16 bits;
  - 8×8 bits.
    - Operands are written to two registers, each one with 8 bits or 16 bits.

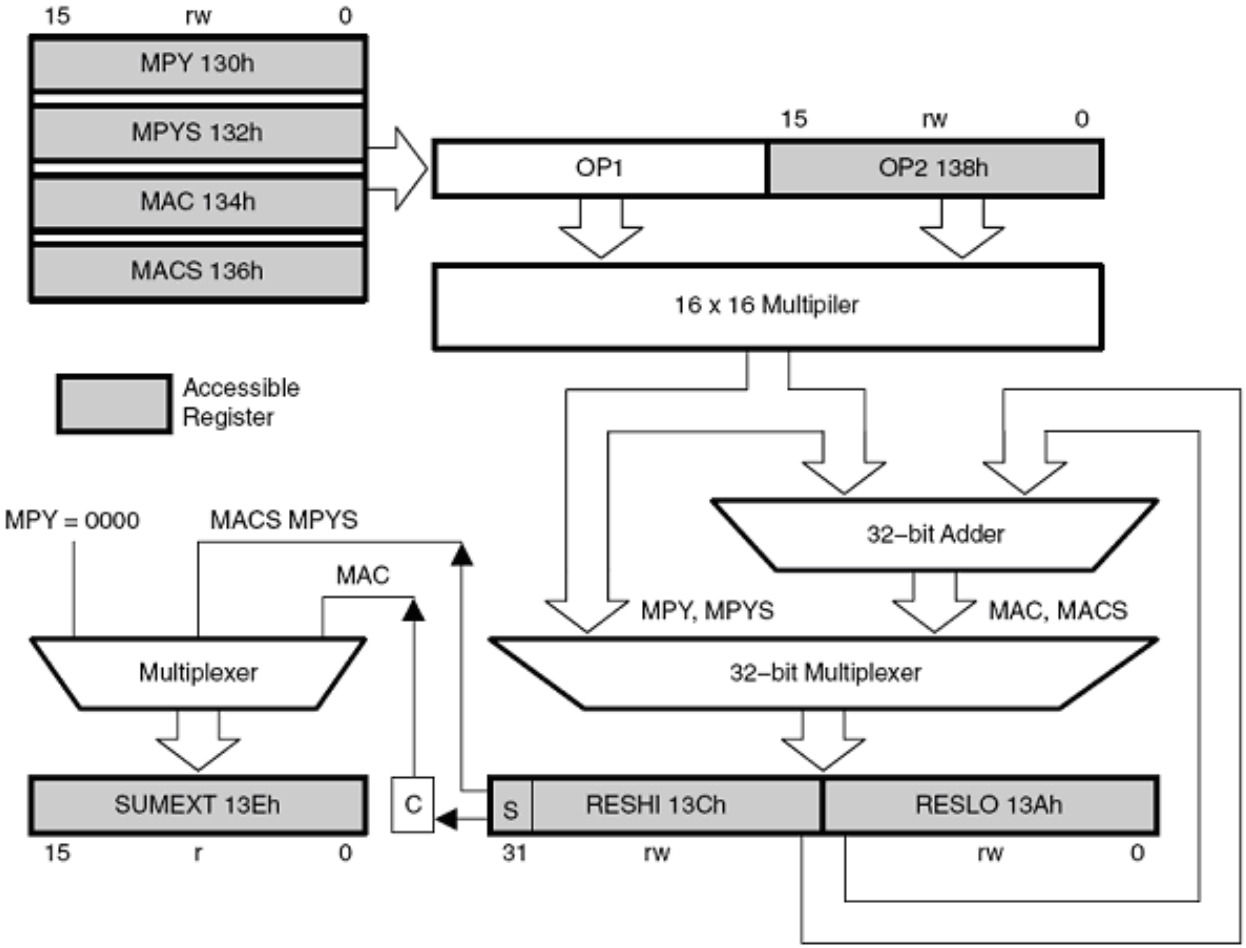
Hardware Multiplier Introduction (3/3)

- **The result of an operation can be accessed by reading two or three registers:**
  - Result low 16-bit word (bits 15 .. 0) in register RESLO;
  - Result high 16-bit word (bits 31 .. 16) in register RESHI;
  - When used MAC or MACS: bit 32 in register SUMEXT.

- **The result is available three MCLK cycles after the operands have been loaded into the peripheral registers;**

Hardware Multiplier Structure (1/4)

- **Block diagram:**



Hardware Multiplier Structure (2/4)

- **Two 16-bit operand registers:**
  - First operand register, OP1:
    - Has four addresses used to select the multiply mode:

Register name	Multiplication operation	OP1 Address
MPY	Unsigned multiply	0130h
MPYS	Signed multiply	0132h
MAC	Unsigned multiply accumulate	0134h
MACS	Signed multiply accumulate	0136h

  - Second operand register, OP2:
    - Writing to the register initiates the multiply operation.

Hardware Multiplier Structure (3/4)

- **Three result registers, RESLO, RESHI, and SUMEXT:**
  - RESLO stores the low word of the result;
  - RESHI stores the high word of the result:
    - The contents of RESHI depend on the multiply operation:

Multiplication operation	RESHI content
Unsigned multiply (MPY)	Upper 16-bits of the result
Signed multiply (MPYS)	Bit 15 (MSB): sign Bits 14 - 0: upper 15-bits of the result Data format: Two's complement
Unsigned multiply accumulate (MAC)	Upper 16-bits of the result
Signed multiply accumulate (MACS)	Upper 16-bits of the result Data format: Two's complement

Hardware Multiplier Structure (4/4)

- **Three result registers, RESLO, RESHI, and SUMEXT:**
  - SUMEXT contains information about the result:
    - The contents of SUMEXT depend on the multiply operation:

Multiplication operation	SUMEXT content
Unsigned multiply (MPY)	SUMEXT = 0000h
Signed multiply (MPYS)	Extended sign of the result: SUMEXT = 00000h ⇒ Result was positive or zero SUMEXT = 0FFFFh ⇒ Result was negative
Unsigned multiply accumulate (MAC)	Carry of the result: SUMEXT = 0000h ⇒ No carry for result SUMEXT = 0001h ⇒ Result has a carry
Signed multiply accumulate (MACS)	Extended sign of the result: SUMEXT = 00000h ⇒ Result was positive or zero SUMEXT = 0FFFFh ⇒ Result was negative

Multiplication Operation (1/2)

Unsigned Multiply (MPY)

- The two operands are treated as unsigned numbers
  - In the range 00000h (smallest number) to 0FFFFh (largest number)
- The maximum possible result is obtained with input operands 0FFFFh and 0FFFFh:
  - 0FFFFh x 0FFFFh = 0FFFE0001h
- No carry is possible and the SUMEXT register always contains zero

Signed Multiply (MPYS)

- The two operands are treated as signed Two's complement numbers
  - In the range 08000h (most negative number, –32768 in decimal) to 07FFFh (most positive number, +32767 in decimal)

Multiplication Operation (2/2)

- The SUMEXT register contains the extended sign of the calculated result:
  - SUMEXT = 00000h: the result is positive
  - SUMEXT = 0FFFFh: the result is negative

Multiply-and-Accumulate (MAC)

- The two operands are treated as unsigned numbers (0h to 0FFFFh)
- The maximum possible result is obtained with input operands 0FFFFh and 0FFFFh:
  - 0FFFFh x 0FFFFh = 0FFFE0001h
- This result is added to the previous contents of the two sum registers (SUMLO and SUMHI)
  - SUMEXT = 00000h: no carry occurred
  - SUMEXT = 00001h: a carry occurred

Hardware Multiplier Registers

- **The hardware multiplier registers are not intended to define the type of multiplication operation;**

- **They simply contain the operands and the data result:**

Register name	Description
MPY	Operand 1 - Unsigned multiply
MPYS	Operand 1 - Signed multiply
MAC	Operand 1 - Unsigned multiply accumulate
MACS	Operand 1 - Signed multiply accumulate
OP2	Operand 2
RESLO	Result (low word)
RESHI	Result (high word)
SUMEXT	Sum extension register

Flash memory Introduction (1/4)

- **Memory in general is broadly classified as read-only memory (ROM) or random-access memory (RAM);**
- **Flash memory is a hybrid of ROM and RAM;**
- **Flash memory is:**
  - Low cost;
  - Electrically programmable;
  - Fast to read from;
  - High density;
  - Reliable;
  - Non-volatile.

Flash memory Introduction (2/4)

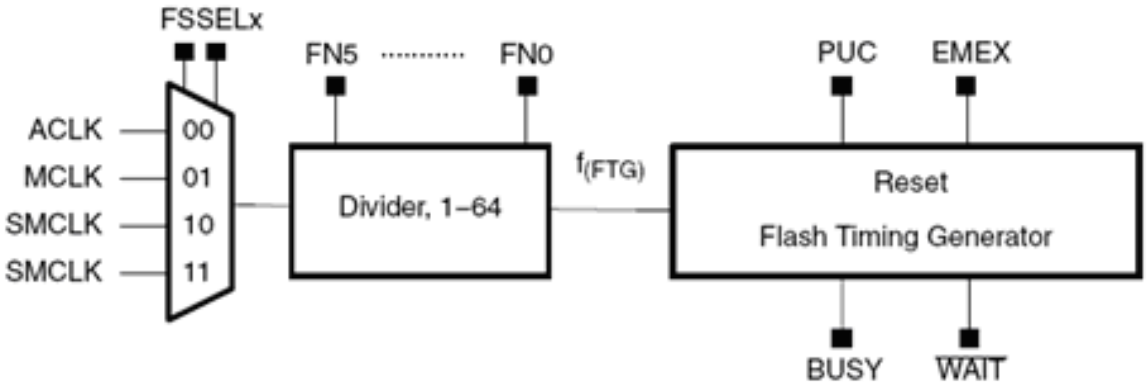
- **MSP430Fxxx(x) flash memory structure is:**
  - Divided into segments;
  - Allows bit-, byte- and word- addressing and programming;
  - Must be erased in segments.

- **Flash memory controller:**
  - Controls programming and erase operations;
  - Has 3 or 4 registers (see the device-specific data sheet);
  - Has a timing generator:
    - Sourced from ACLK, SMCLK, or MCLK;
    - Flash timing generator operating frequency: ~257 kHz < $f_{(FTG)}$ < ~476 kHz (see device-specific data);
    - The selected clock source should be divided using the FNx bits to meet the frequency requirements of $f_{(FTG)}$.

Flash memory Introduction (3/4)

Flash memory controller:

- Timing generator block diagram:



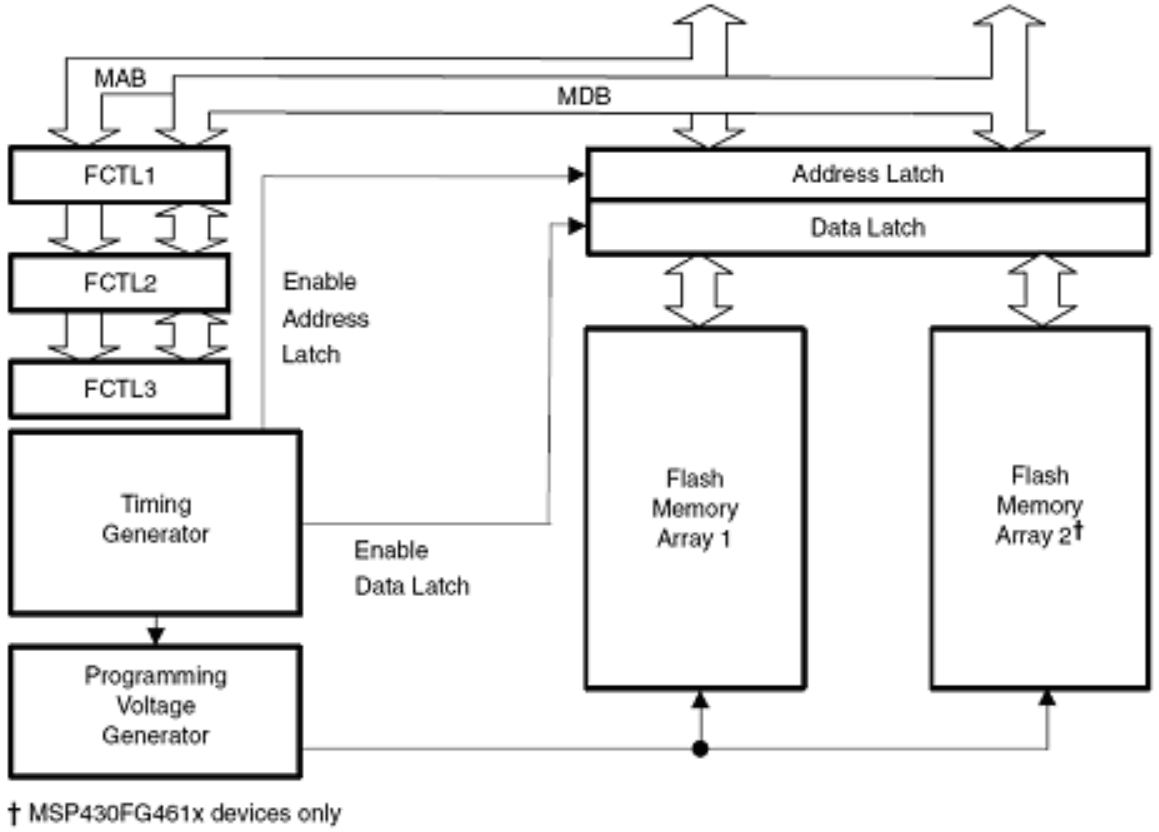
- Uses a voltage generator to supply programming and erase voltages. The output voltage must be stable.

Flash memory Introduction (4/4)

- **An MSP430 flash device can be programmed via:**
  - JTAG interface (requires four signals, ground and optionally VCC and RST/NMI);
  - Bootstrap Loader (using a UART serial interface);
  - Custom solution (using one of the interfaces available and through user developed software).

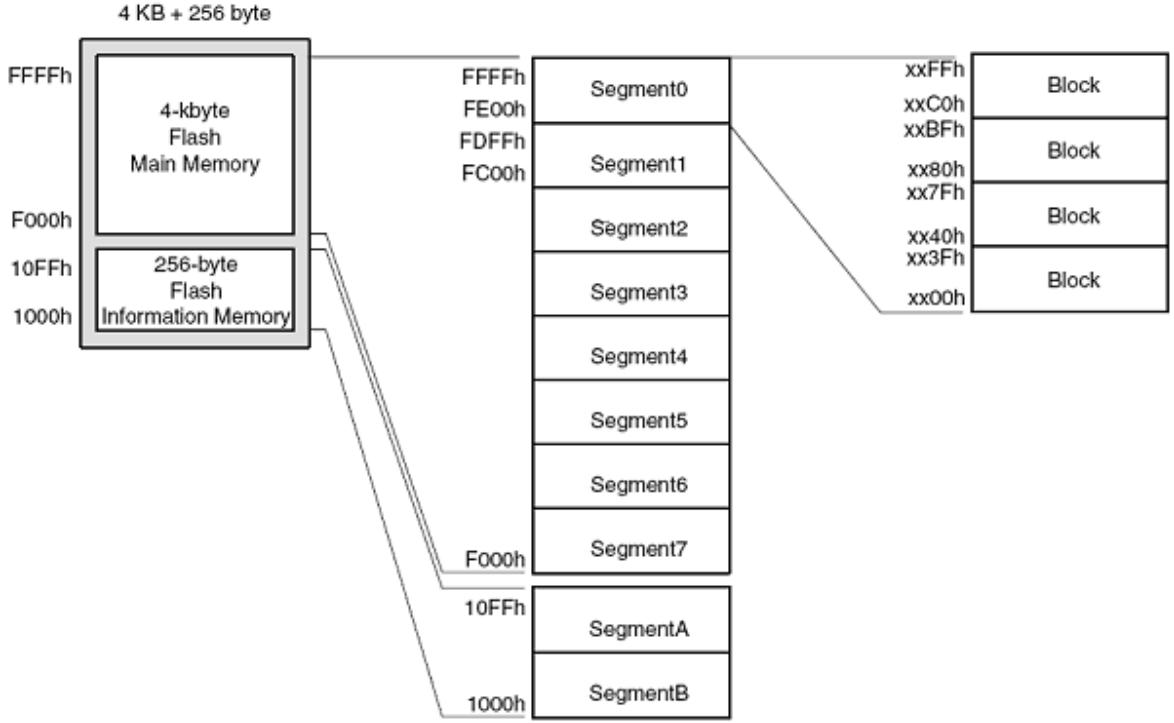
Flash memory operation and segmentation (1/3)

- **MSP430 flash memory block diagram:**
  - MSP430FG4618 (Experimenter's board) has two flash memory arrays.



Flash memory operation and segmentation (2/3)

- **The flash memory partitions (device-specific data):**
  - Main memory section (two or more 512-byte segments);
  - Information memory section (two 128-byte segments), located at lower memory addresses, in the address space immediately following RAM.



Flash memory operation and segmentation (3/3)

- **2xx family: SegmentA (information A):**
  - eZ430-F2013;
  - eZ430-RF2500.
  - Partition of the information memory
  - Can be locked to separate it from all other segments:
    - LOCKA = 1:
      - SegmentA cannot be written or erased;
      - All information memory is protected from erasure during a mass erase or production programming.
    - LOCKA = 0:
      - SegmentA can be erased and written;
      - All information memory is erased during a mass erase or production programming.

Flash memory write/erase modes

- **Default mode: read mode (memory operates like ROM):**
  - Flash memory is not being erased or written;
  - Flash timing generator off;
  - Voltage generator off.
- **The flash memory write/erase modes are selected with the BLKWRT, WRT, GMERAS, MERAS, and ERASE bits;**
- **To stop any write or erase operation before its normal completion, set the EMEX bit.**
  - When EMEX = 1:
    - All flash operations cease;
    - The flash returns to read mode;
    - All bits in the FCTL1 register are reset.

Erase modes (1/2)

- **Erase modes:**
  - Initiated from within flash memory:
    - All timing is controlled by the flash controller;
    - CPU is held during the erase cycle (dummy write);
    - CPU resumes code execution after the erase cycle finishes.
  - Initiated from RAM:
    - CPU is not held and can execute code from RAM;
    - CPU can access any flash address again when BUSY = 0 (end of the erase cycle).

Erase modes (2/2)

Erase modes:

Bits				Mode description	
GMERAS[1]	LOCKA[2]	MERAS	ERASE	MSP430FG461x	MSP430F2xxx
X	-	0	1	Segment erase	Segment erase
0	-	1	0	Mass erase (main memory segments- selected array)	Mass erase (all main memory segments)
0	0	1	1	Erase all flash memory (main and information segments – selected array)	Erase main and information flash memory
1	-	1	0	Global mass erase (all main memory segments – both arrays)	Mass erase (all main memory segments)
1	1	1	1	Erase main memory and information segments- both arrays	Erase main flash memory

[1] This bit is only present in the MSP430FG461x devices

[2] This bit is only present in the MSP430F2xxx devices

Erase mode procedure (1/3)

- **Segment Erase:**
  - Check BUSY = 0 (FCTL3 register);
  - LOCK = 0 (FCTL3 register);
  - ERASE = 1 (FCTL1 register);
  - Perform a dummy write to the segment to be erased (Any write, clear or logical operation);
  - A segment erase requires approximately 5000 cycles of the timing generator (during this period BUSY = 1);
  - Wait for BUSY = 0 (FCTL3 register).
  - LOCK = 1 (FCTL3 register) to prevent accidental writes.

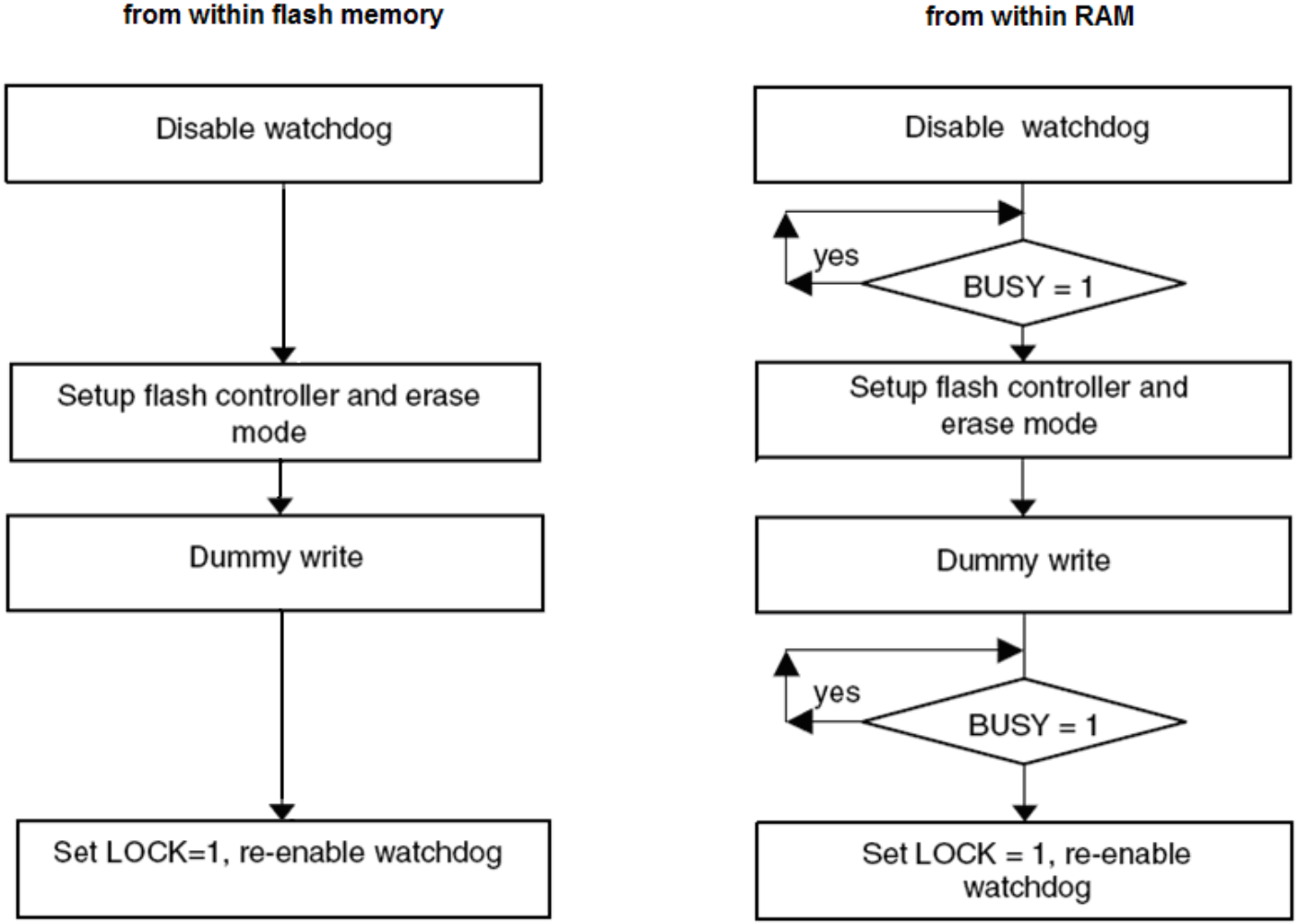
Erase mode procedure (2/3)

- **Mass Erase (all main memory segments):**
  - Similar to Segment erase;
  - Requires setting MERAS bit instead of ERASE bit in the FCTL1 register.

- **All Erase (all segments):**
  - Requires setting (GMERAS), MERAS and ERASE bits in the FCTL1 register.

Erase mode procedure (3/3)

- **Segment and mass erase modes procedure:**



Write modes

- A byte/word write cycle can be initiated from within flash memory or from RAM;
- A block write cycle cannot be initiated from within flash memory (only from RAM);
- The block write can be used to accelerate the flash write process (twice as fast as byte/word mode), when many sequential bytes or words need to be programmed.

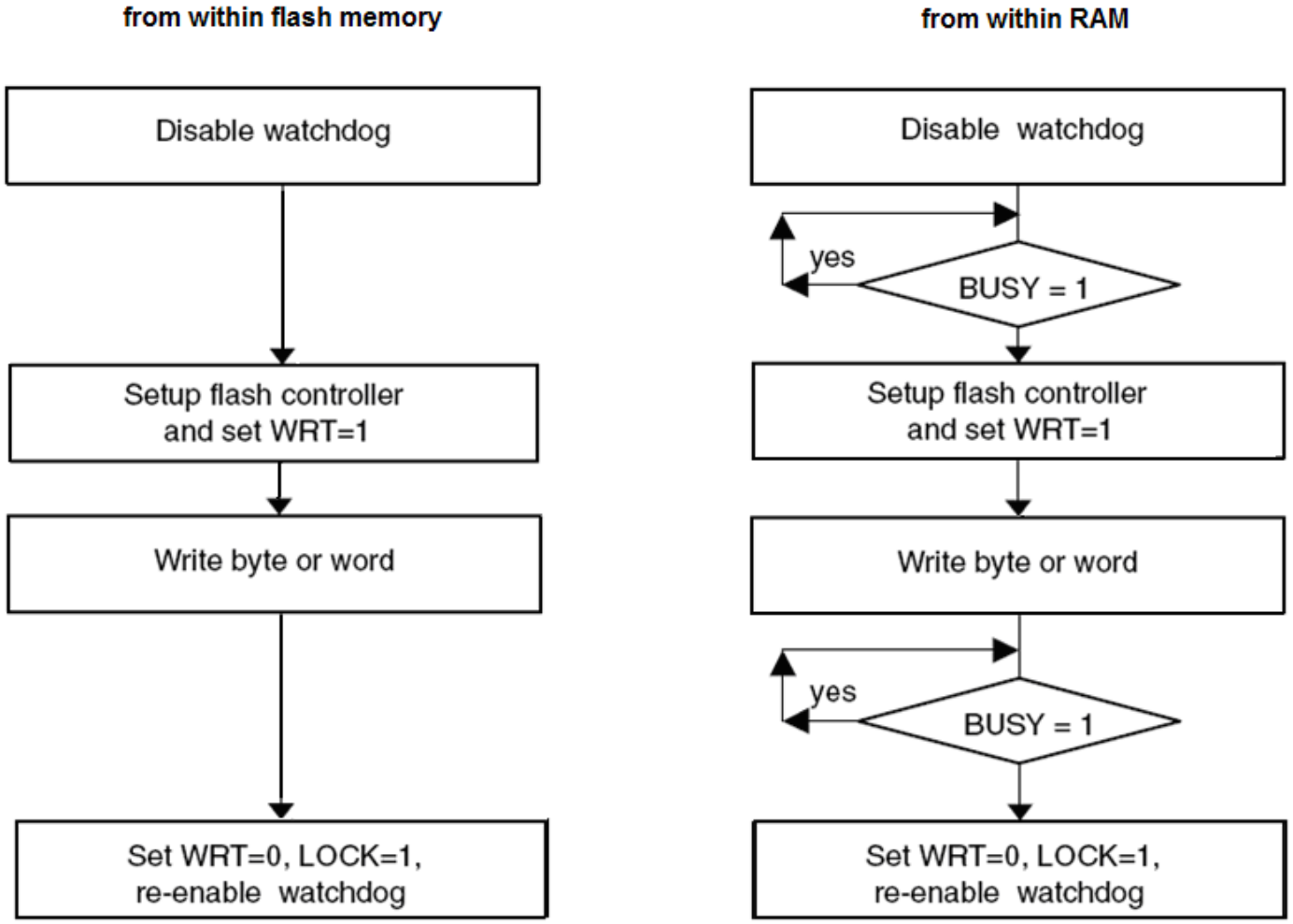
Bits		Mode description
BLKWRT	WRT	MSP430FG461x and MSP430F2xxx
0	1	Byte/word write
1	1	Block write

Write modes procedure (1/4)

- **Byte/word write:**
  - Check BUSY = 0 (FCTL3 register);
  - LOCK = 0 (FCTL3 register);
  - WRT = 1 (FCTL1 register);
  - Write the byte or word (element) to the appropriate address (starts the timing generator);
  - To write an element requires 33 cycles of the timing generator (during this period BUSY = 1);
  - Wait for BUSY = 0 (FCTL3 register);
  - LOCK = 1 (FCTL3 register) to prevent accidental writes.

Write modes procedure (2/4)

- **Byte/word write:**

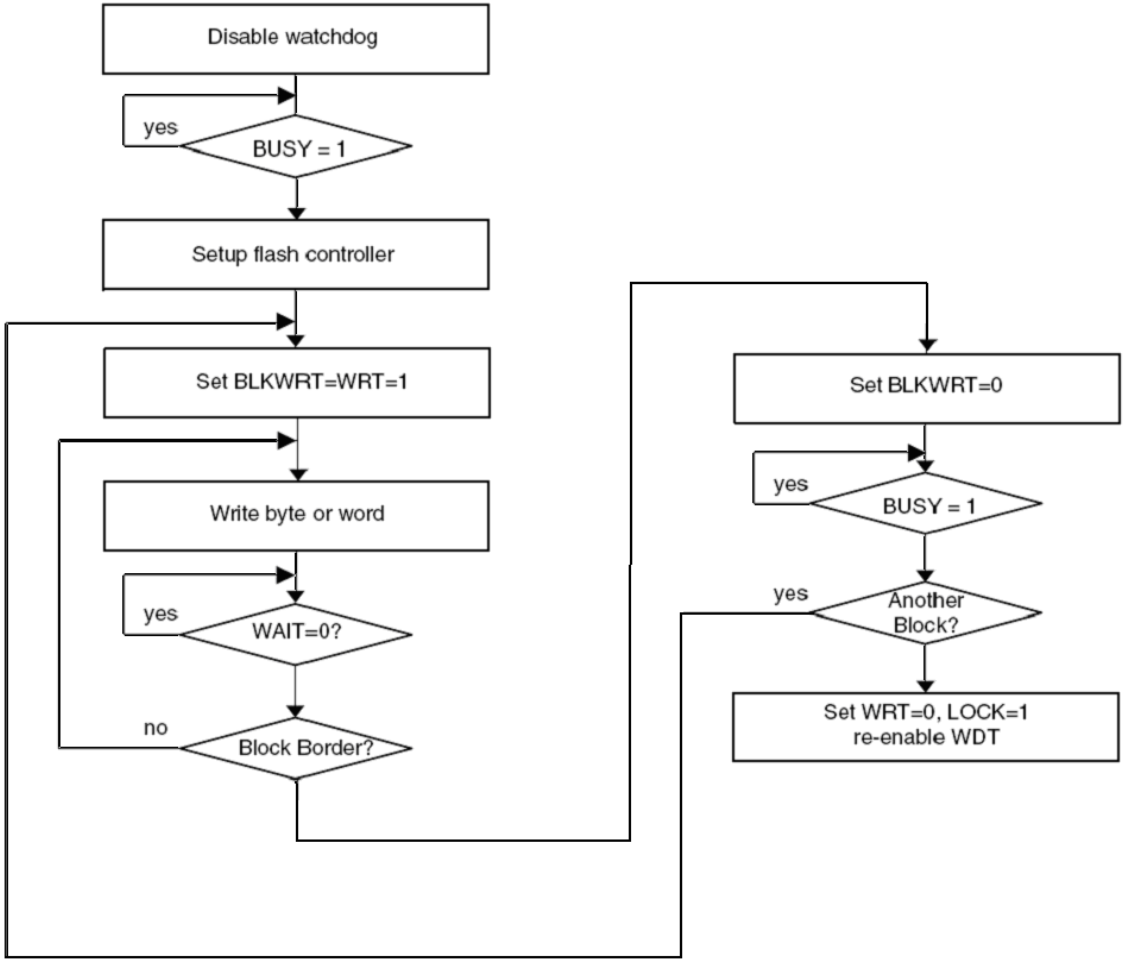


Write modes procedure (3/4)

- **Write a block (successive write of 64 bytes in a block):**
  - Check BUSY = 0 (FCTL3 register);
  - LOCK = 0 (FCTL3 register);
  - WRT = 1 and BLKWRT = 1 (FCTL1 register);
  - Write the element in the block to the appropriate address (starts the timing generator);
  - Loop until WAIT = 1 (FCTL3 register);
  - Repeat write next element until all have been written;
  - Set WRT = 0 and BLKWRT = O (FCTL1 register);
  - A block write requires 20 cycles Timing Generator/element, + overhead: 15 more cycles (during this period BUSY = 1);
  - Wait for BUSY = 0 (FCTL3 register);
  - LOCK = 1 (FCTL3 register) to prevent accidental writes.

Write modes procedure (4/4)

- **Block (64-byte blocks) write:**



Flash memory access during write/erase

- **When BUSY = 1, any write or any erase operation initiated from RAM or from flash memory triggers the following conditions:**

Flash operation	Flash access	Wait	Result
Any erase Byte/word write	Read	0	ACCVIFG = 0 Value read: 03FFFh
	Write	0	ACCVIFG = 0 Write ignored
	Instruction fetch	0	ACCVIFG = 0 CPU fetch: 03FFFh
Block write	Any	0	ACCVIFG = 1 LOCK = 1
	Read	1	ACCVIFG = 0 Value read: 03FFFh
	Write	1	ACCVIFG = 0 Flash written
	Instruction fetch	1	ACCVIFG = 1 LOCK = 1

Registers (1/3)

❑ FCTL1, Flash Memory Control Register (MSP430FG4618)

15	14	13	12	11	10	9	8
(FCTLx password) Read: FRKEY = 096h Write (must be): FWKEY = 0A5h							

7	6	5	4	3	2	1	0
BLKWRT	WRT	Reserved	EEIEX(1)	EEI(1) GMERAS(2)	MERAS	ERASE	Reserved

(1) MSP430F2xx(x) family devices. Not present on MSP430F2013.

(2) MSP430FG461x devices.

Bit		Description
7	BLKWRT	Block write mode when BLKWRT = 1 (WRT must also be set)
6	WRT	Write when WRT = 1
4	EEIEX(1)	Enable Emergency Interrupt Exit when EEIEX = 1 and GIE = 1
3	EEI(1) GMERAS(2)	(1) Enable segment Erase to be interrupted by an interrupt request when EEI = 1 (2) Global mass erase
2	MERAS	Mass erase
1	ERASE	Erase

Registers (2/3)

❑ FCTL2, Flash Memory Control Register (MSP430FG4618)

15	14	13	12	11	10	9	8
(FCTLx password) Read: FRKEY = 096h Write (must be): FWKEY = 0A5h							

7	6	5	4	3	2	1	0
FSSELX		FNx					

Bit		Description
7-6	FSSELx	Flash controller clock source: FSSEL1 FSSEL0 = 00 ⇒ ACLK FSSEL1 FSSEL0 = 01 ⇒ MCLK FSSEL1 FSSEL0 = 10 ⇒ SMCLK FSSEL1 FSSEL0 = 11 ⇒ SMCLK
5-0	FNx	Flash controller clock divider FNx=00h ⇒ /1 … FNx=03Fh ⇒ /64

Registers (3/3)

❑ FCTL3, Flash Memory Control Register (MSP430FG4618)

15	14	13	12	11	10	9	8
(FCTLx password) Read: FRKEY = 096h Write (must be): FWKEY = 0A5h							

7	6	5	4	3	2	1	0
FAIL(1)	LOCKA(1)	EMEX	LOCK	WAIT	ACCVIFG	KEYV	BUSY

Bit		Description
7	FAIL	When FAIL = 1, operation failure of the clock source, $f_{(FTG)}$, or a flash operation is aborted from an interrupt when EEIEX = 1
6	LOCKA(1)	Segment A locked and all information memory is protected from erasure during a mass erase when LOCKA = 1
5	EMEX	Emergency exit when EMEX = 1
4	LOCK	Locks the flash memory for writing or erasing when LOCK = 1
3	WAIT	WAIT = 0 ⇒ while flash memory is being written to WAIT = 1 ⇒ when flash memory is ready for the next byte/word write
2	ACCVIFG	Access violation interrupt flag ACCVIFG = 1 when interrupt is pending
1	KEYV	Flash security key violation KEYV = 1 when FCTLx password was written incorrectly (not 0A5h)
0	BUSY	Flash timing generator is busy when BUSY = 1

TLV Introduction (1/3)

- **The Tag-Length-Value (TLV) structure is used in selected MSP430x2xx devices to provide device-specific information in the device's flash memory SegmentA, such as calibration data.**

Word Address	Upper Byte	Lower Byte	Tag Address and Offset
0x10FE	CALBC1_1MHZ	CALDCO1_1MHZ	0x10F6 + 0x0008
0x10FC	CALBC1_8MHZ	CALDCO1_8MHZ	0x10F6 + 0x0006
0x10FA	CALBC1_12MHZ	CALDCO1_12MHZ	0x10F6 + 0x0004
0x10F8	CALBC1_16MHZ	CALDCO1_16MHZ	0x10F6 + 0x0002
0x10F6	0x08 (LENGTH)	TAG_DCO_30	0x10F6
0x10F4	0xFF	0xFF	
0x10F2	0xFF	0xFF	
0x10F0	0xFF	0xFF	
0x10EE	0xFF	0xFF	
0x10EC	0x08 (LENGTH)	TAG_EMPTY	0x10EC
0x10EA	CAL_ADC_25T85		0x10DA + 0x0010
0x10E8	CAL_ADC_25T30		0x10DA + 0x000E
0x10E6	CAL_ADC_25VREF_FACTOR		0x10DA + 0x000C
0x10E4	CAL_ADC_15T85		0x10DA + 0x000A
0x10E2	CAL_ADC_15T30		0x10DA + 0x0008
0x10E0	CAL_ADC_15VREF_FACTOR		0x10DA + 0x0006
0x10DE	CAL_ADC_OFFSET		0x10DA + 0x0004
0x10DC	CAL_ADC_GAIN_FACTOR		0x10DA + 0x0002
0x10DA	0x10 (LENGTH)	TAG_ADC12_1	0x10DA
0x10D8	0xFF	0xFF	
0x10D6	0xFF	0xFF	
0x10D4	0xFF	0xFF	
0x10D2	0xFF	0xFF	
0x10D0	0xFF	0xFF	
0x10CE	0xFF	0xFF	
0x10CC	0xFF	0xFF	
0x10CA	0xFF	0xFF	
0x10C8	0xFF	0xFF	
0x10C6	0xFF	0xFF	
0x10C4	0xFF	0xFF	
0x10C2	0x16 (LENGTH)	TAG_EMPTY	0x10C2
0x10C0	2th complement of bit-wise XOR		0x10C0

TLV Introduction (2/3)

- **The first two bytes of SegmentA (0x10C0 and 0x10C1) hold the checksum of the remainder of the segment (addresses 0x10C2 to 0x10FF):**
  - The checksum is a bit-wise XOR of 31 words stored in the twos-complement data format.
- **The first tag is located at address 0x10C2:**
  - In this example, the TAG_EMPTY tag.
- **The following byte (0x10C3) holds the length of the structure:**
  - The length of this TAG_EMPTY structure is 0x16;
  - The next tag, TAG_ADC12_1, is found at address 0x10DA;
  - The following byte holds the length of the TAG_ADC12_1 structure.

TLV Introduction (3/3)

- **The TLV structure maps the entire address range 0x10C2 to 0x10FF of the SegmentA:**
  - A program routine looking for tags starting at the SegmentA address 0x10C2 can extract all information even if it is stored at a different (device-specific) absolute address.

Supported Tags (1/5)

- **Each device contains a subset of the tags:**
  - See the device-specific data sheet for details.

Tag	Description	Value
TAG_EMPTY	Identifies an unused memory area	0xFE
TAG_DCO_30	Calibration values for the DCO at room temperature and DV_{CC} = 3 V	0x01
TAG_ADC12_1	Calibration values for the ADC12 module	0x08

- **TAG_ADC12_1 Calibration TLV Structure:**
  - Consists of eight words.

Label	Description	Offset
CAL_ADC_25T85	VREF2_5 = 1, T_A = 85°C ± 2K, 12-bit conversion result	0x0E
CAL_ADC_25T30	VREF2_5 = 1, T_A = 30°C ± 2K, 12-bit conversion result	0x0C
CAL_ADC_25VREF_FACTOR	VREF2_5 = 1, T_A = 30°C ± 2K	0x0A
CAL_ADC_15T85	VREF2_5 = 0, T_A = 85°C ± 2K, 12-bit conversion result	0x08
CAL_ADC_15T30	VREF2_5 = 0, T_A = 30°C ± 2K, 12-bit conversion result	0x06
CAL_ADC_15VREF_FACTOR	VREF2_5 = 0, T_A = 30°C ± 2K	0x04
CAL_ADC_OFFSET	Ve_{REF} = 2.5V, T_A = 85°C ± 2K, $f_{ADC12CLK}$ = 5 MHz	0x02
CAL_ADC_GAIN_FACTOR	Ve_{REF} = 2.5V, T_A = 85°C ± 2K, $f_{ADC12CLK}$ = 5 MHz	0x00

Supported Tags (2/5)

- **Temperature Sensor Calibration Data:**
  - The temperature sensor is calibrated using the internal voltage references;
  - At VREF2_5 = 0 and 1, the conversion result at 30°C and 85°C is written at the respective SegmentA location:
    - See Table for TAG_ADC12_1 Calibration Data.

- **Integrated Voltage Reference Calibration Data**
  - The reference voltages (VREF2_5 = 0 and 1) are measured at room temperature;
  - The measured value is normalized by 1.5/2.5V before stored into the flash information memory SegmentA:

$$CAL\_ADC\_15VREF\_FACTOR = \frac{Ve_{REF}}{1.5V} \times 2^{15}$$

Supported Tags (3/5)

- The conversion result is corrected by multiplying it with the CAL_ADC_15VREF_FACTOR (or CAL_ADC_25VREF_FACTOR) and dividing the result by 2^{15}:

$$\text{ADC(corrected)} = \text{ADC(raw)} \times \text{CAL\_ADC\_15VREF\_FACTOR} \times \frac{1}{2^{15}}$$

❑ Offset and Gain Calibration Data:

- The offset of the ADC12 is determined and stored as a two's complement number in SegmentA;
- The offset error correction is done by adding the CAL_ADC_OFFSET to the conversion result:

$$\text{ADC(offset\_corrected)} = \text{ADC(raw)} + \text{CAL\_ADC\_OFFSET}$$

Supported Tags (4/5)

- The gain of the ADC12, stored at offset 0x00, is calculated by the following equation:

$$\text{CAL\_ADC\_GAIN\_FACTOR} = \frac{1}{\text{GAIN}} \times 2^{15}$$

- The conversion result is gain corrected by multiplying it with the CAL_ADC_GAIN_FACTOR and dividing the result by 2^{15}:

$$\text{ADC(gain\_corrected)} = \text{ADC(raw)} \times \text{CAL\_ADC\_GAIN\_FACTOR} \times \frac{1}{2^{15}}$$

- If both gain and offset are corrected, the gain correction is done first.

$$\text{ADC(gain\_corrected)} = \text{ADC(raw)} \times \text{CAL\_ADC\_GAIN\_FACTOR} \times \frac{1}{2^{15}}$$

$$\text{ADC(final)} = \text{ADC(gain\_corrected)} + \text{CAL\_ADC\_OFFSET}$$

Supported Tags (5/5)

DCO Calibration TLV Structure:

- The BCS+ registers (BCSCTL1 and DCOCTL) are used.
- The values stored in the flash information memory SegmentA are written to the BCS+ registers.

Label	Description	Offset
CALBC1_1MHZ	Value for the BCSCTL1 register for 1 MHz, $T_A = 25^{\circ}C$	0x07
CALDCO_1MHZ	Value for the DCOCTL register for 1 MHz, $T_A = 25^{\circ}C$	0x06
CALBC1_8MHZ	Value for the BCSCTL1 register for 8 MHz, $T_A = 25^{\circ}C$	0x05
CALDCO_8MHZ	Value for the DCOCTL register for 8 MHz, $T_A = 25^{\circ}C$	0x04
CALBC1_12MHZ	Value for the BCSCTL1 register for 12 MHz, $T_A = 25^{\circ}C$	0x03
CALDCO_12MHZ	Value for the DCOCTL register for 12 MHz, $T_A = 25^{\circ}C$	0x02
CALBC1_16MHZ	Value for the BCSCTL1 register for 16 MHz, $T_A = 25^{\circ}C$	0x01
CALDCO_16MHZ	Value for the DCOCTL register for 16 MHz, $T_A = 25^{\circ}C$	0x00