EE 444/645: Embedded Systems Design

Low Power Operation

Spring 2017. Set: **6**

Instructor: Dr. Dejan Raskovic

Power equations for CMOS logic

$$\boxed{P} \boxed{= \underbrace{ACV^2 f} + \underbrace{\tau A V I_{short} f} + \underbrace{V I_{leak}}}$$

Dynamic power consumption b/c charge/discharge
A – Activity of gates
C – Capacitance seen by gate's outputs
V – Supply voltage
f – System frequency
(this term is dominant)

Power expended b/c of the short-circuit current (between V and Gnd when CMOS switches)
A – Activity of gates
τ – Duration of I_{short}
V – Supply voltage
f – System frequency

Power lost from the leakage current
V – Supply voltage
I_{leak} – Leakage current

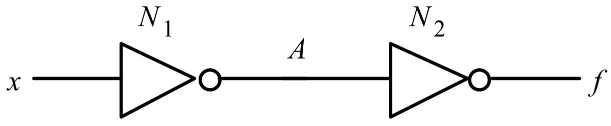
Dynamic Operation of Logic Gates

- □ Parasitic capacitance
  - ■ Side effect of fabrication
  - ■ Sources
    - □ Capacitor between the N_2 input and ground
    - □ Transistors in N_1
    - □ Metal wiring, etc.
- □ Gate capacitance

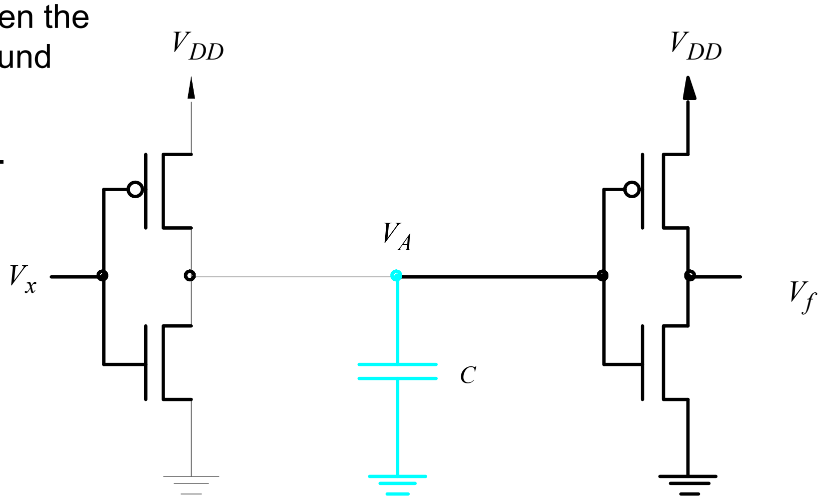
$$C_g = W \times L \times C_{ox}$$

- □ Oxide capacitance

$$C_{ox}[fF/\mu m^2]$$



(a) A NOT gate driving another NOT gate



(b) The capacitive load at node A

Dynamic Operation of Logic Gates

- □ Remember MOSFET current equations?

$$I_D = k_n' \frac{W}{L}\left[(V_{GS} - V_T)\cdot V_{DS} - \frac{1}{2}V_{DS}^2\right]$$

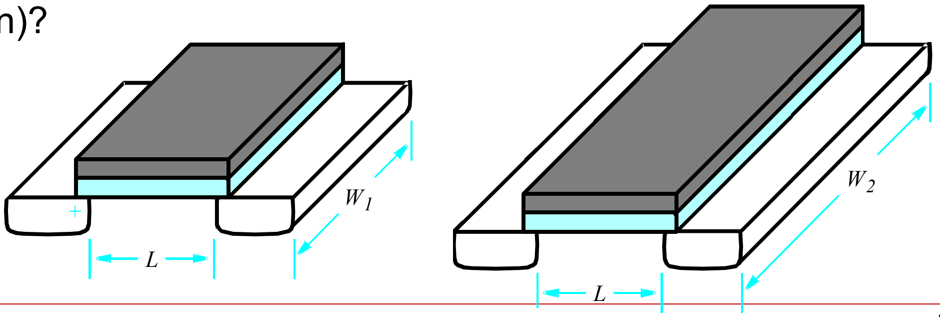
$$I_D = \frac{1}{2}k_n' \frac{W}{L}(V_{GS} - V_T)^2$$

- □ Time to charge a capacitor:

$$t_p = \frac{C\Delta V}{I_D} = \frac{CV_{DD}/2}{I_D}$$

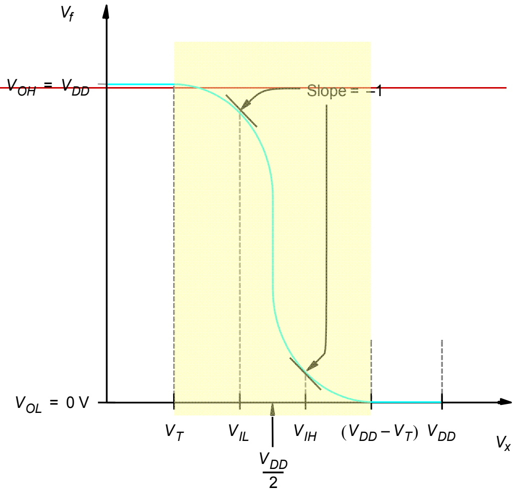
$$\boxed{t_p \cong \frac{1.7C}{k_n' \frac{W}{L} V_{DD}}}$$

- □ How can we reduce propagation delay?
  - ■ What happens to the current, capacitance? Do we have a trade-off (again)?

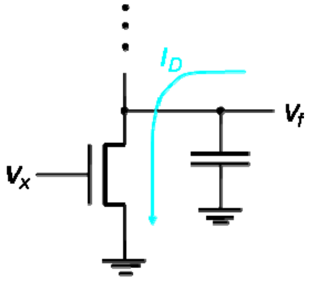


Power Dissipation

- □ There's a range of input voltages where both transistors are on
  - ■ Short-circuit current
  - ■ Usually negligible, but…*
- □ When the output is high, the amount of energy stored is: $\dfrac{CV_{DD}^2}{2}$
- □ When the output changes from V_{DD} to 0, this energy is dissipated through NMOS T.



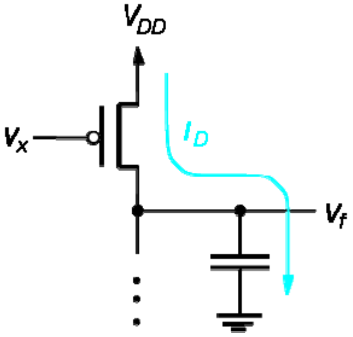
(and same here:)



Therefore, the power (dynamic) dissipated in the inverter is defined as:

$$P_D = fCV_{DD}^2$$

Number of charge/discharge cycles per second



Power equations for CMOS logic

$$P = \underbrace{ACV^2 f}_{\text{(this term is dominant)}} + \tau AVI_{\text{short}} f + VI_{\text{leak}}$$

First reaction: reduce voltage supply
Unfortunately, there's a price to be paid:

$$f_{\max} \propto (V - V_{\text{threshold}})^2 / V$$

Also, reducing voltage requires reduction of threshold, which increases the leakage current:

$$I_{\text{leak}} \propto \exp(-qV_{\text{threshold}} / (kT))$$

Power and energy consumption

- Term “power” is sometimes misleading, especially in battery powered devices
  - More concerned by energy, because batteries store limited amount of energy
- Other measures:
  - Energy/operation ratio, or it’s inverse: MIPS/W
  - Energy × delay
  - Energy + deadline

How can we reduce power/energy consumption?

- Dynamic power consumption
  - charge/discharge of the capacitive load on each gate’s output
  - frequency
- Control activity
  - reduce power supply voltage
  - reduce working frequency
  - turn off unused parts (module enables)
  - use low power modes
  - interrupt driven system
- Minimize the number of transitions
  - instruction formats, coding?

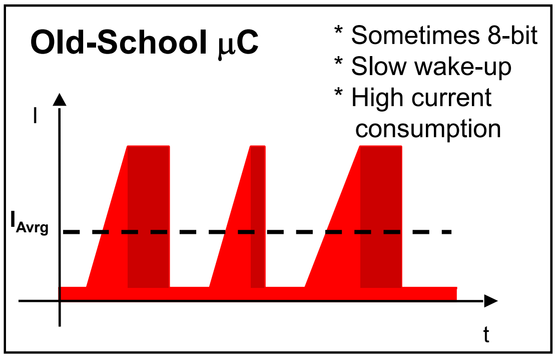
Reducing power consumption

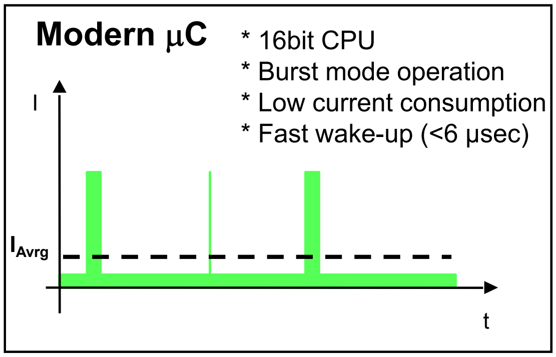
- □ Logic
  - Clock tree (up to 30% of power)
  - Clock gating (turn off branches that are not used)
  - Half frequency clock (both edges)
  - Half swing clock (half of Vcc)
  - Asynchronous logic
- □ Architecture
  - Parallelism (increased area and wiring)
  - Speculation (branch prediction)
  - Memory systems
    - □ Memory access (dynamic)
    - □ Memory banks (turn off unused)
  - Buses
    - □ 32-64 address/data, (15-20% of power)
    - □ Gray Code, Code compression

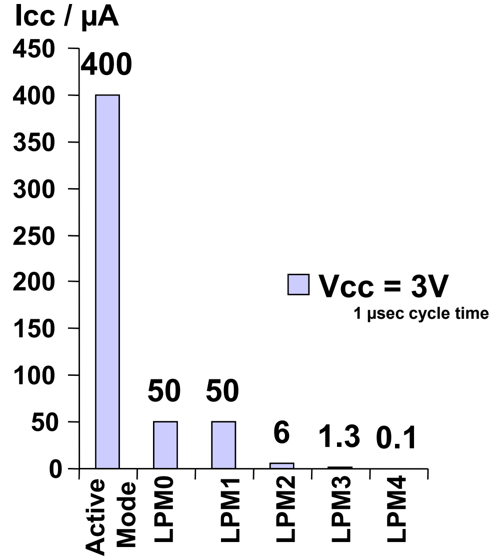
Reducing power consumption #2

- □ Operating System
  - Finish computation “when necessary”
  - Scale the voltage
    - □ Application driven
    - □ Automatic
- □ System Architecture
  - Power efficient and specialized processing cores
  - Trade-off (+ technology)
    - □ AMD K6 / 400MHz / 64KB cache – 12W
    - □ XScale with the same cache 450 mW @ 600 MHz (40mW@150MHz)

The Ultra-Low-Power Concept





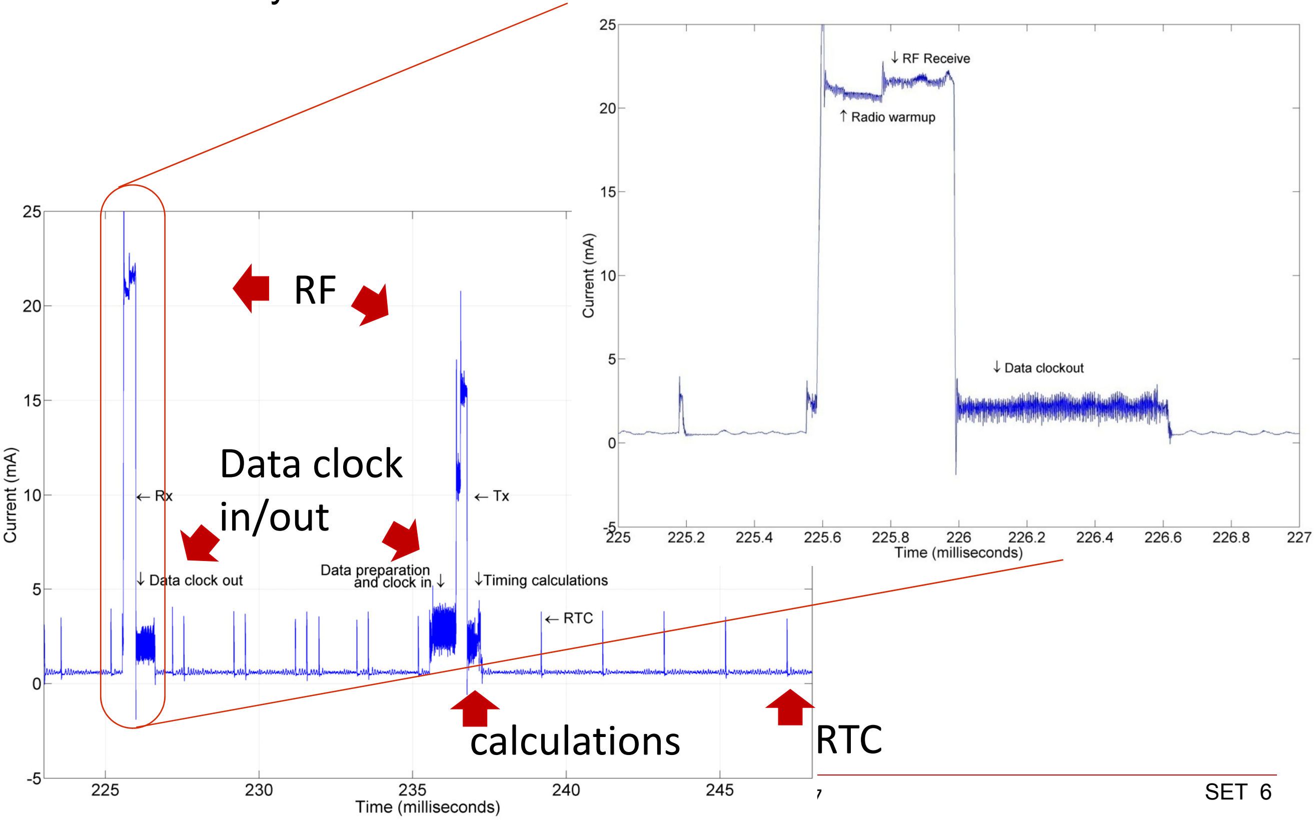


LPM3, Standby Mode (MSP430C31x)

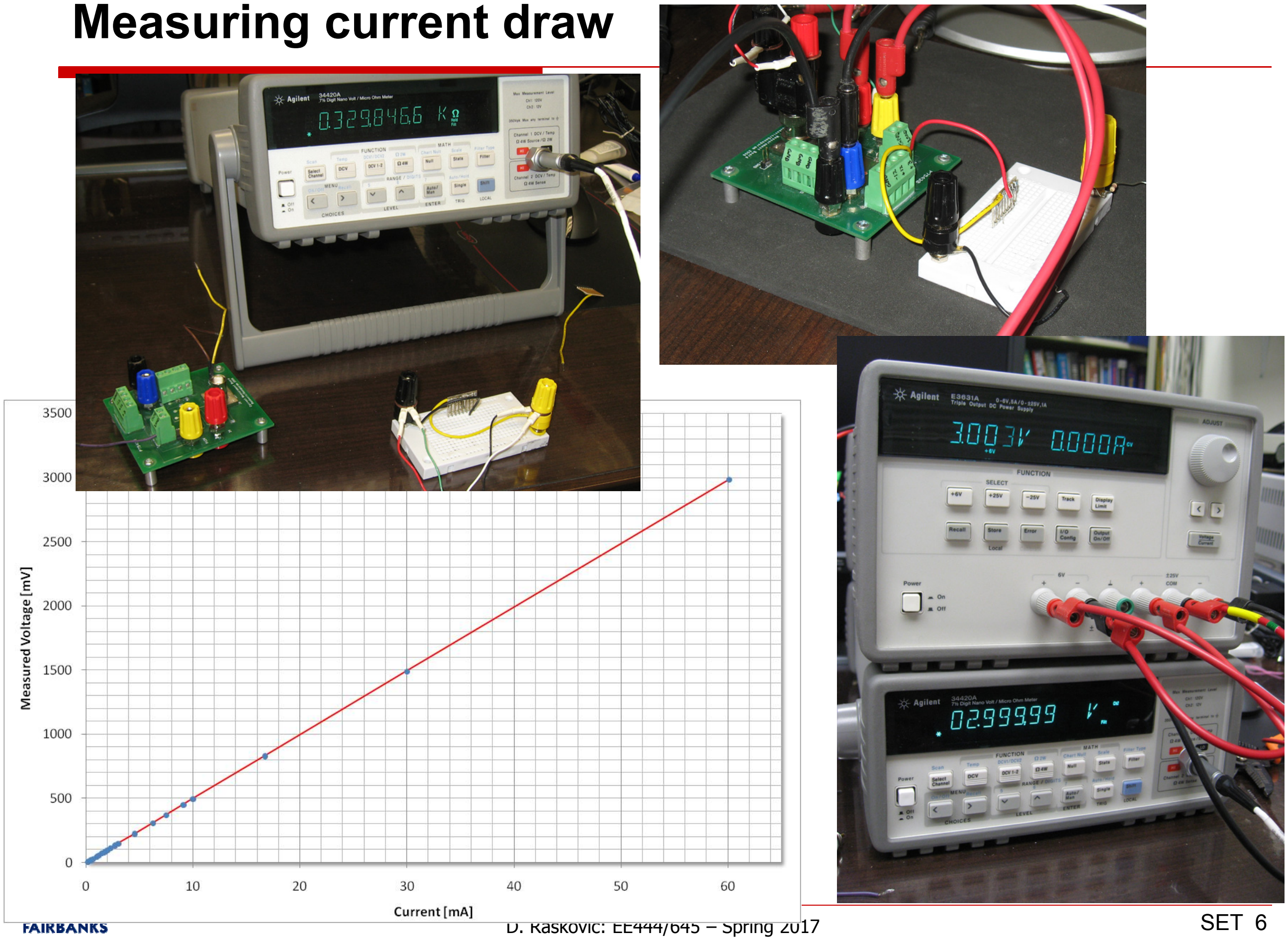
Oscillator 32768Hz active,
Real-Time Clock active via Basic Timer1,
LCD drive active via LCD module,

A typical activity profile

- □ Precisely measure current draw



Measuring current draw



Low-Power Concept: Intelligent Supply Usage

- □ Reduce the product of energy consumption:

$$\boxed{E = V * I * t}$$

t_{sleep} 👍

- ■ Reduce the operating voltage when possible
- ■ Reduce the current consumption,
  - □ the DC and AC current
  - □ the operating frequency (if appropriate)
- ■ Reduce the operational time of each module,
  - □ CPU
  - □ System
  - □ Peripheral Modules

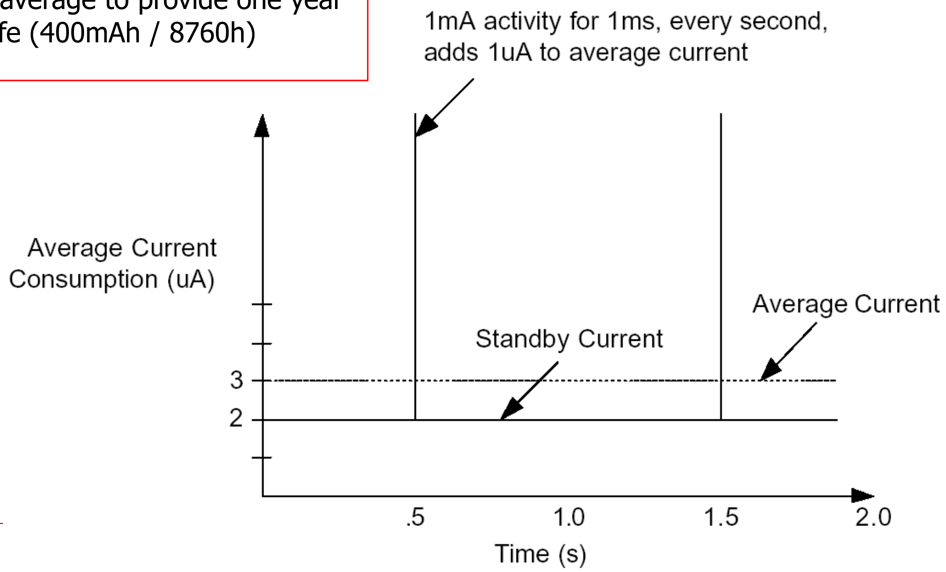
t_{active} 👎

- □ Support interrupt driven operating modes

$$\boxed{Q = \sum_{i=1}^{n} I_i * t_i}$$

Low-Power Concept: Basic Conditions for Burst Mode

- In low-power designs, the *average* current consumption determines battery life
- Example:
  - Battery capacity 400 mAh
  - The application must draw less than 45.7 uA on average to provide one year of battery life (400mAh / 8760h)



Low power / low energy issues in microcontrollers

- Average current (burst mode, etc.)
- Power-down modes
  - enabling and disabling clocks and peripherals
- The clocking system
  - instant-on clocks (are fast clocks also instant-on?)
  - when the fast clocks becomes reliable?
- Interrupts
  - event-driven rather than polling
  - interrupts for all internal peripherals plus some for external components
- Peripherals
  - ability to individually enable/disable peripherals (or on-demand)
  - interoperability of peripherals; DMA
  - peripherals controlling the clock
- Pin leakage
- Processing efficiency (8 vs. 16 bit, etc.)

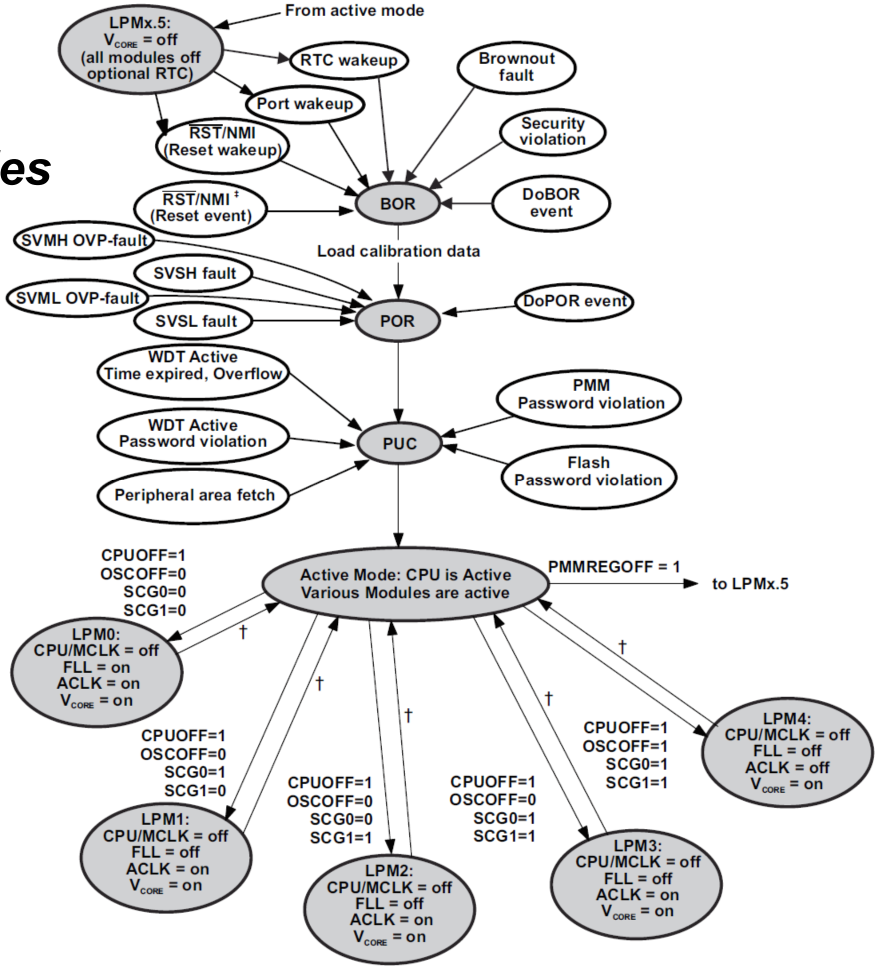
Low power mode control

- Four bits in the Status Register control the CPU and the operation of the system clock generator:
  - CPUOff,
  - OscOff,
  - SCG0, and
  - SCG1.

15 9	8	7							0
Reserved	V	SCG1	SCG0	OSC OFF	CPU OFF	GIE	N	Z	C

- These four bits support discontinuous active mode (AM) requests, to limit the time period of the full operating mode
- Located in the status register
- The present state of the operating condition is saved onto the stack during an interrupt service request.
- As long as the stored status register information is not altered, the processor continues (after RETI) with the same operating mode as before the interrupt event.

Operating Modes



LPMx.5: V_{CORE} = off (all modules off optional RTC)

From active mode

RTC wakeup

Port wakeup

Brownout fault

Security violation

RST/NMI (Reset wakeup)

RST/NMI ‡ (Reset event)

BOR

DoBOR event

SVMH OVP-fault

Load calibration data

SVSH fault

SVML OVP-fault

SVSL fault

POR

DoPOR event

WDT Active Time expired, Overflow

PMM Password violation

WDT Active Password violation

PUC

Flash Password violation

Peripheral area fetch

Active Mode: CPU is Active Various Modules are active

PMMREGOFF = 1

to LPMx.5

CPUOFF=1 OSCOFF=0 SCG0=0 SCG1=0

LPM0: CPU/MCLK = off FLL = on ACLK = on V_{CORE} = on

CPUOFF=1 OSCOFF=0 SCG0=1 SCG1=0

LPM1: CPU/MCLK = off FLL = off ACLK = on V_{CORE} = on

CPUOFF=1 OSCOFF=0 SCG0=0 SCG1=1

LPM2: CPU/MCLK = off FLL = off ACLK = on V_{CORE} = on

CPUOFF=1 OSCOFF=0 SCG0=1 SCG1=1

LPM3: CPU/MCLK = off FLL = off ACLK = on V_{CORE} = on

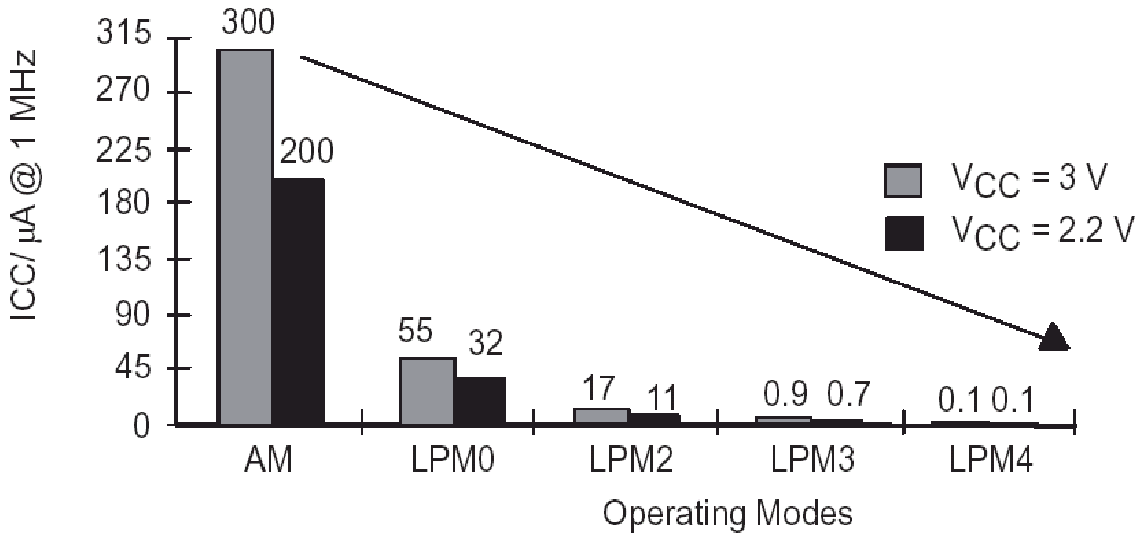
CPUOFF=1 OSCOFF=1 SCG0=1 SCG1=1

LPM4: CPU/MCLK = off FLL = off ACLK = off V_{CORE} = on

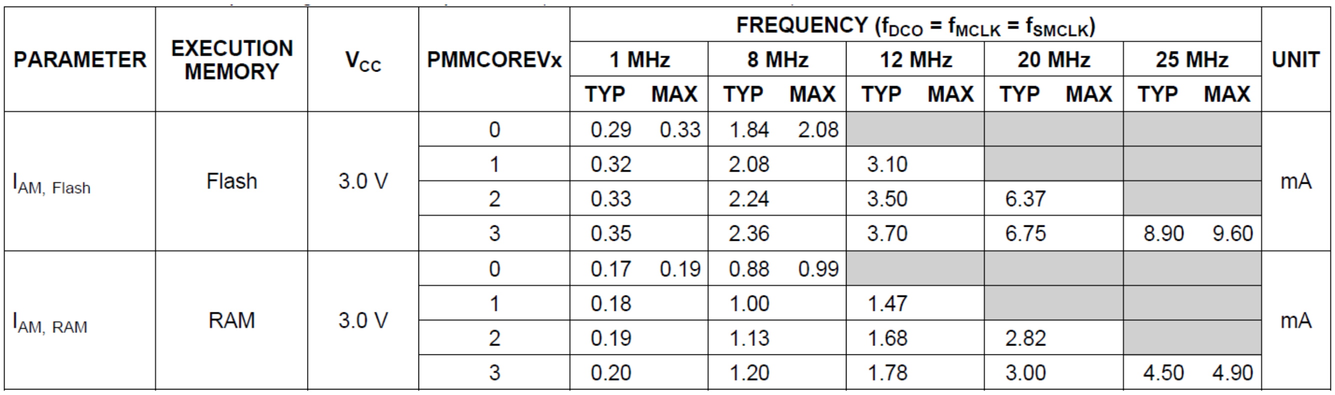
† † † † † †

Low power modes current consumption

41x devices

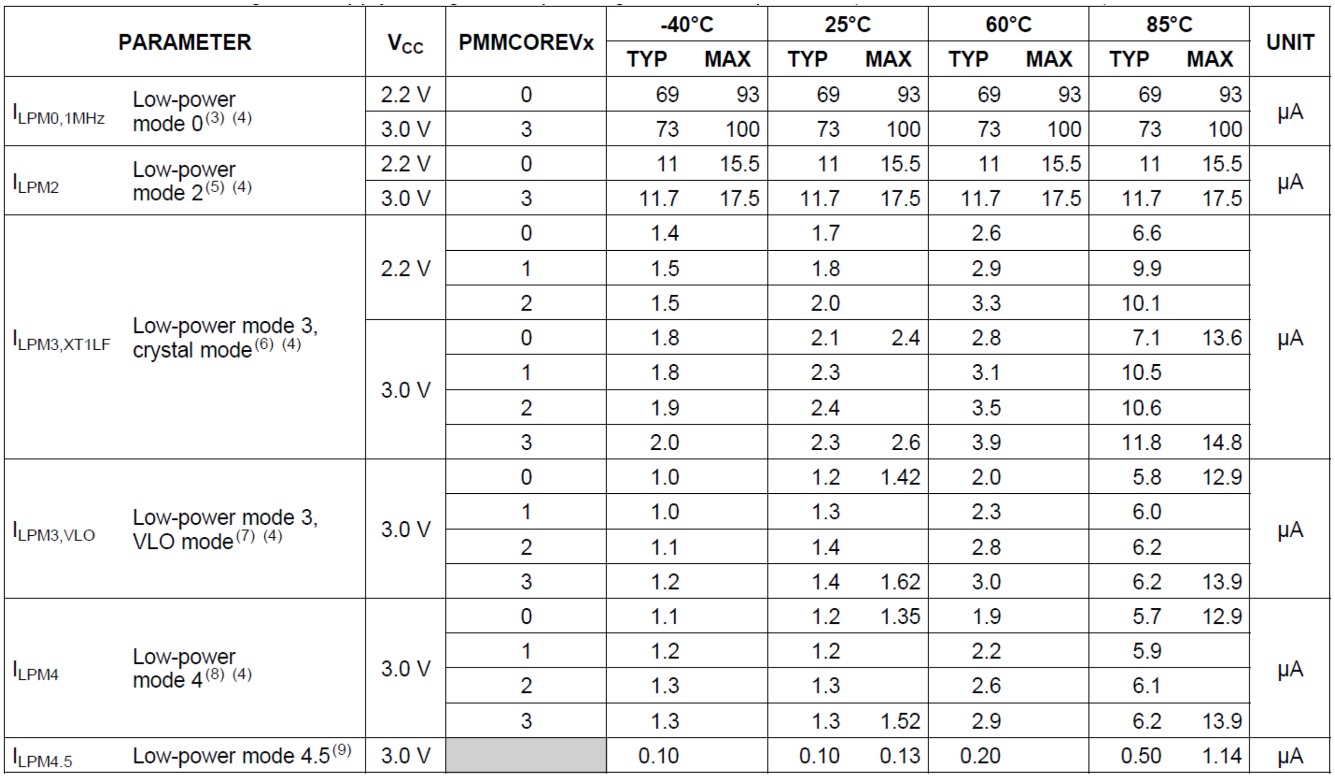


Active mode – x543x



PARAMETER	EXECUTION MEMORY	V_{CC}	PMMCOREVx	FREQUENCY ($f_{DCO} = f_{MCLK} = f_{SMCLK}$)										UNIT
				1 MHz		8 MHz		12 MHz		20 MHz		25 MHz		
				TYP	MAX	TYP	MAX	TYP	MAX	TYP	MAX	TYP	MAX	
$I_{AM,\ Flash}$	Flash	3.0 V	0	0.29	0.33	1.84	2.08							mA
			1	0.32		2.08		3.10						
			2	0.33		2.24		3.50		6.37				
			3	0.35		2.36		3.70		6.75		8.90	9.60	
$I_{AM,\ RAM}$	RAM	3.0 V	0	0.17	0.19	0.88	0.99							mA
			1	0.18		1.00		1.47						
			2	0.19		1.13		1.68		2.82				
			3	0.20		1.20		1.78		3.00		4.50	4.90	

Low power modes – x543x



PARAMETER		V_{CC}	PMMCOREVx	-40°C		25°C		60°C		85°C		UNIT
				TYP	MAX	TYP	MAX	TYP	MAX	TYP	MAX	
$I_{LPM0,1MHz}$	Low-power mode 0[3] [4]	2.2 V	0	69	93	69	93	69	93	69	93	μA
		3.0 V	3	73	100	73	100	73	100	73	100	
I_{LPM2}	Low-power mode 2[5] [4]	2.2 V	0	11	15.5	11	15.5	11	15.5	11	15.5	μA
		3.0 V	3	11.7	17.5	11.7	17.5	11.7	17.5	11.7	17.5	
$I_{LPM3,XT1LF}$	Low-power mode 3, crystal mode[6] [4]	2.2 V	0	1.4		1.7		2.6		6.6		μA
			1	1.5		1.8		2.9		9.9		
			2	1.5		2.0		3.3		10.1		
		3.0 V	0	1.8		2.1	2.4	2.8		7.1	13.6	
			1	1.8		2.3		3.1		10.5		
			2	1.9		2.4		3.5		10.6		
			3	2.0		2.3	2.6	3.9		11.8	14.8	
$I_{LPM3,VLO}$	Low-power mode 3, VLO mode[7] [4]	3.0 V	0	1.0		1.2	1.42	2.0		5.8	12.9	μA
			1	1.0		1.3		2.3		6.0		
			2	1.1		1.4		2.8		6.2		
			3	1.2		1.4	1.62	3.0		6.2	13.9	
I_{LPM4}	Low-power mode 4[8] [4]	3.0 V	0	1.1		1.2	1.35	1.9		5.7	12.9	μA
			1	1.2		1.2		2.2		5.9		
			2	1.3		1.3		2.6		6.1		
			3	1.3		1.3	1.52	2.9		6.2	13.9	
$I_{LPM4.5}$	Low-power mode 4.5[9]	3.0 V		0.10		0.10	0.13	0.20		0.50	1.14	μA

Operating Modes – Software Configurable

- Active mode (AM): SCG1=0, SCG0=0, OscOff=0, CPUOff=0
  - CPU, MCLK are active.
  - ACLK is active. SMCLK optionally active (SMCLKOFF = 0).
  - DCO is enabled if sources ACLK, MCLK, or SMCLK (SMCLKOFF = 0).
  - FLL is enabled if DCO is enabled.
- Low power mode 0 (LPM0): SCG1=0, SCG0=0, OscOff=0, CPUOff=1
  - CPU, MCLK are disabled.
  - ACLK is active. SMCLK optionally active (SMCLKOFF = 0).
  - DCO is enabled if sources ACLK or SMCLK (SMCLKOFF = 0).
  - FLL is enabled if DCO is enabled
- Low power mode 1 (LPM1): SCG1=0, SCG0=1, OscOff=0, CPUOff=1
  - CPU, MCLK are disabled.
  - ACLK is active. SMCLK optionally active (SMCLKOFF = 0).
  - DCO is enabled if sources ACLK or SMCLK (SMCLKOFF = 0).
  - FLL is disabled.

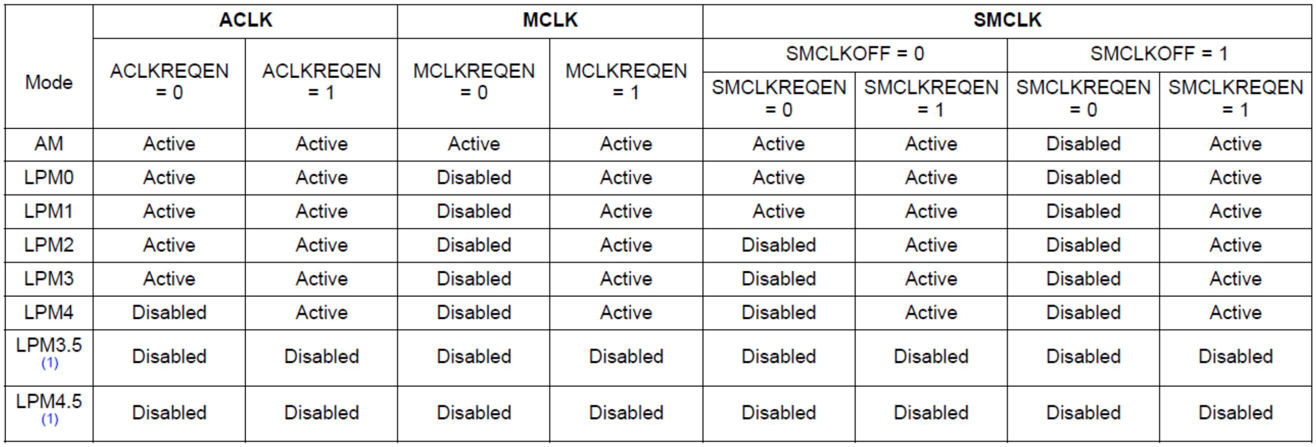
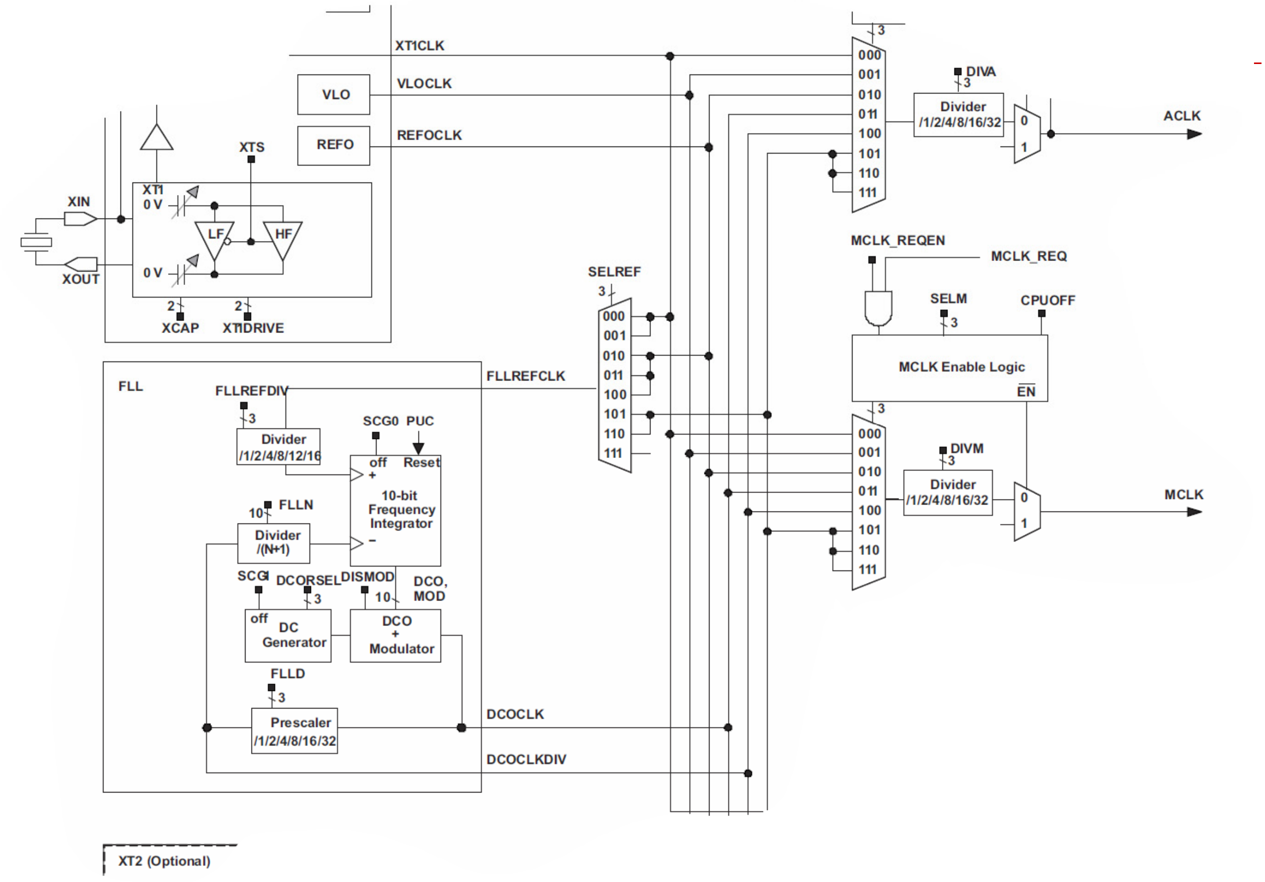


Table 5-1. Clock Request System and Power Modes

Mode	ACLK		MCLK		SMCLK			
	ACLKREQEN = 0	ACLKREQEN = 1	MCLKREQEN = 0	MCLKREQEN = 1	SMCLKOFF = 0		SMCLKOFF = 1	
					SMCLKREQEN = 0	SMCLKREQEN = 1	SMCLKREQEN = 0	SMCLKREQEN = 1
AM	Active	Active	Active	Active	Active	Active	Disabled	Active
LPM0	Active	Active	Disabled	Active	Active	Active	Disabled	Active
LPM1	Active	Active	Disabled	Active	Active	Active	Disabled	Active
LPM2	Active	Active	Disabled	Active	Disabled	Active	Disabled	Active
LPM3	Active	Active	Disabled	Active	Disabled	Active	Disabled	Active
LPM4	Disabled	Active	Disabled	Active	Disabled	Active	Disabled	Active
LPM3.5 (1)	Disabled	Disabled	Disabled	Disabled	Disabled	Disabled	Disabled	Disabled
LPM4.5 (1)	Disabled	Disabled	Disabled	Disabled	Disabled	Disabled	Disabled	Disabled

Operating Modes – Software Configurable #2

- Low power mode 2 (LPM2): SCG1=1, SCG0=0, OscOff=0, CPUOff=1
  - CPU, MCLK are disabled.
  - ACLK is active. SMCLK is disabled.
  - DCO is enabled if sources ACLK.
  - FLL is disabled
- Low power mode 3 (LPM3): SCG1=1, SCG0=1, OscOff=0, CPUOff=1
  - CPU, MCLK are disabled.
  - ACLK is active. SMCLK is disabled.
  - DCO is enabled if sources ACLK.
  - FLL is disabled.
- Low power mode 4 (LPM4): SCG1=1, SCG0=1, OscOff=1, CPUOff=1
  - CPU and all clocks are disabled.



Low power modes – x449 Example

PARAMETER		TEST CONDITIONS	V_{CC}	MIN	TYP	MAX	UNIT
$I_{(AM)}$	Active mode (see Note 1), $f_{(MCLK)} = f_{(SMCLK)}$ = 1 MHz, $f_{(ACLK)}$ = 32768 Hz XTS_FLL=0, SELM=(0,1)	T_A = –40°C to 85°C	2.2 V		280	350	µA
			3 V		420	560	
$I_{(LPM0)}$	Low-power mode, (LPM0) (see Note 1 and Note 4)	T_A = –40°C to 85°C	2.2 V		32	45	µA
			3 V		55	70	
$I_{(LPM2)}$	Low-power mode, (LPM2), f(MCLK) = f (SMCLK) = 0 MHz, f(ACLK) = 32768 Hz, SCG0 = 0 (see Note 2 and Note 4)	T_A = –40°C to 85°C	2.2 V		11	14	µA
			3 V		17	22	
$I_{(LPM3)}$	Low-power mode, (LPM3) $f_{(MCLK)} = f_{(SMCLK)}$ = 0 MHz, $f_{(ACLK)}$ = 32,768 Hz, SCG0 = 1 (see Note 3 and Note 4)	T_A = –40°C	2.2 V		1	1.5	µA
		T_A = 25°C			1.1	1.5	
		T_A = 60°C			2	3	
		T_A = 85°C			3.5	6	
		T_A = –40°C	3 V		1.8	2.2	µA
		T_A = 25°C			1.6	1.9	
		T_A = 60°C			2.5	3.5	
		T_A = 85°C			4.2	7.5	
$I_{(LPM4)}$	Low-power mode, (LPM4) $f_{(MCLK)}$ = 0 MHz, $f_{(SMCLK)}$ = 0 MHz, $f_{(ACLK)}$ = 0 Hz, SCG0 = 1 (see Note 2 and Note 4)	T_A = –40°C	2.2 V		0.1	0.5	µA
		T_A = 25°C			0.1	0.5	
		T_A = 60°C			0.7	1.1	
		T_A = 85°C			1.7	3	
		T_A = –40°C	3 V		0.1	0.5	µA
		T_A = 25°C			0.1	0.5	
		T_A = 60°C			0.8	1.2	
		T_A = 85°C			1.9	3.5	

Low power / low energy issues in microcontrollers

4xx series:

Current consumption of active mode versus system frequency

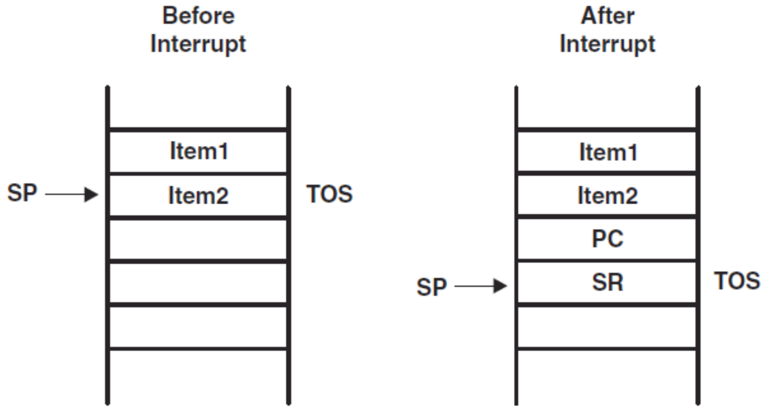
$$I_{(AM)} = I_{(AM)}\,[1\text{ MHz}] \times f_{(System)}\,[\text{MHz}]$$

Current consumption of active mode versus supply voltage

$$I_{(AM)} = I_{(AM)\,[3\text{ V}]} + 175\ \mu\text{A/V} \times (V_{CC} - 3\text{ V})$$

Operating Modes and Interrupts

- Current instruction is completed.
- The PC is pushed onto the stack; The SR is pushed onto the stack.
- The highest priority interrupt is selected
- The IR flag resets automatically on single-source flags, remains set for multiple source flags.
- All bits of SR are cleared except SCG0
  - GIE bit is also cleared, further interrupts are disabled.
- The execution continues with the interrupt service routine



Operating Modes and Interrupts

Selecting a program flow by manipulating the data stored on the stack

- □ Return from interrupt
  - Two different modes are available:

1. Return with low-power mode bits set.
   - When returning from the interrupt, the program counter points to the next instruction.
   - The instruction pointed to is not executed, since the restored low power mode stops CPU activity.
2. Return with low-power mode bits reset.
   - When returning from the interrupt, the program continues at the address following the instruction that set the OscOff or CPUOff-bit in the status register.
   - To use this mode, the interrupt service routine must reset the OscOff, CPUOff, SCG0, and SCG1 bits on the stack.
   - Then, when the SR contents are popped from the stack upon RETI, the operating mode will be active mode (AM).

How to wake up and stay awake?

- □ Use assembly language:

```
; Exit LPM3 Interrupt Service Routine
BIC #CPUOFF+SCG1+SCG0,0(SP) ; Exit LPM3 on RETI
RETI
```

- □ Use vendor-supplied libraries

```
unsigned __bic_SR_register_on_exit(unsigned mask);
```

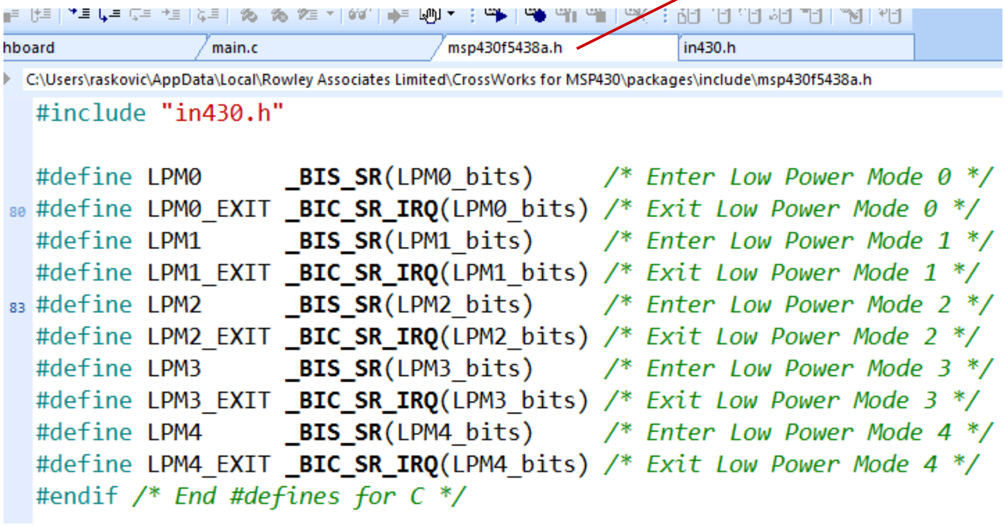
or it’s synonym

```
_BIC_SR_IRQ(X)
```

- □ Use predefined LPMx_EXIT #defines, eg., we have:

```
#define LPM0_EXIT _BIC_SR_IRQ(LPM0_bits)
```

Implementation in CrossStudio



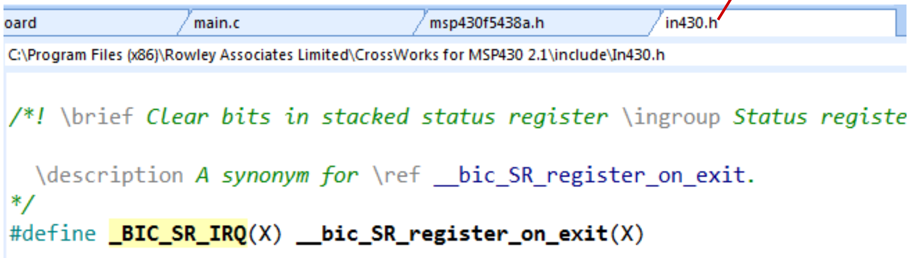
C:\Users\raskovic\AppData\Local\Rowley Associates Limited\CrossWorks for MSP430\packages\include\msp430f5438a.h

```
#include "in430.h"

#define LPM0      _BIS_SR(LPM0_bits)     /* Enter Low Power Mode 0 */
#define LPM0_EXIT _BIC_SR_IRQ(LPM0_bits) /* Exit Low Power Mode 0 */
#define LPM1      _BIS_SR(LPM1_bits)     /* Enter Low Power Mode 1 */
#define LPM1_EXIT _BIC_SR_IRQ(LPM1_bits) /* Exit Low Power Mode 1 */
#define LPM2      _BIS_SR(LPM2_bits)     /* Enter Low Power Mode 2 */
#define LPM2_EXIT _BIC_SR_IRQ(LPM2_bits) /* Exit Low Power Mode 2 */
#define LPM3      _BIS_SR(LPM3_bits)     /* Enter Low Power Mode 3 */
#define LPM3_EXIT _BIC_SR_IRQ(LPM3_bits) /* Exit Low Power Mode 3 */
#define LPM4      _BIS_SR(LPM4_bits)     /* Enter Low Power Mode 4 */
#define LPM4_EXIT _BIC_SR_IRQ(LPM4_bits) /* Exit Low Power Mode 4 */
#endif /* End #defines for C */
```

msp430f5438a.h

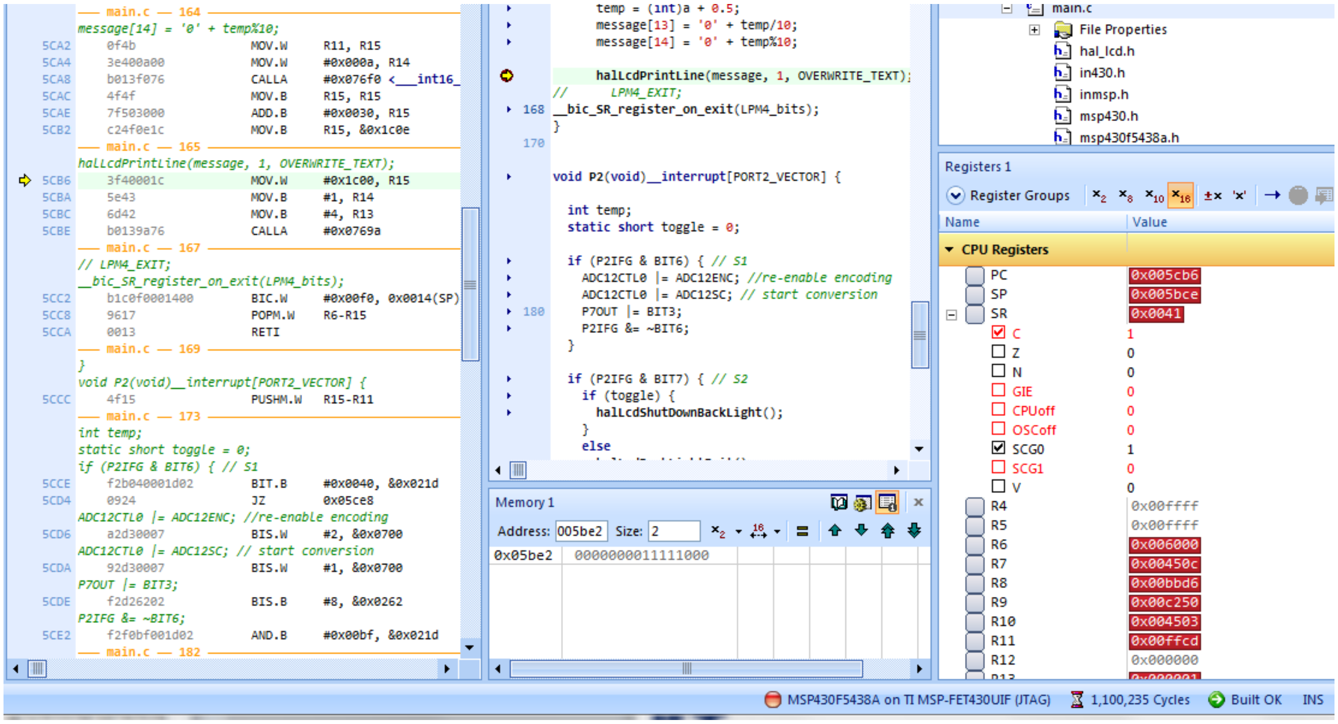
in430.h



C:\Program Files (x86)\Rowley Associates Limited\CrossWorks for MSP430 2.1\include\In430.h

```
/*! \brief Clear bits in stacked status register \ingroup Status registe

  \description A synonym for \ref __bic_SR_register_on_exit.
*/
#define _BIC_SR_IRQ(X) __bic_SR_register_on_exit(X)
```



```
—— main.c —— 164 ——————————————
message[14] = '0' + temp%10;
5CA2    0f4b            MOV.W    R11, R15
5CA4    3e400a00        MOV.W    #0x000a, R14
5CA8    b013f076        CALLA    #0x076f0 <___int16_
5CAC    4f4f            MOV.B    R15, R15
5CAE    7f503000        ADD.B    #0x0030, R15
5CB2    c24f0e1c        MOV.B    R15, &0x1c0e
—— main.c —— 165 ——————————————
halLcdPrintLine(message, 1, OVERWRITE_TEXT);
5CB6    3f40001c        MOV.W    #0x1c00, R15
5CBA    5e43            MOV.B    #1, R14
5CBC    6d42            MOV.B    #4, R13
5CBE    b0139a76        CALLA    #0x0769a
—— main.c —— 167 ——————————————
// LPM4_EXIT;
__bic_SR_register_on_exit(LPM4_bits);
5CC2    b1c0f0001400    BIC.W    #0x00f0, 0x0014(SP)
5CC8    9617            POPM.W   R6-R15
5CCA    0013            RETI
—— main.c —— 169 ——————————————
}
void P2(void)__interrupt[PORT2_VECTOR] {
5CCC    4f15            PUSHM.W  R15-R11
—— main.c —— 173 ——————————————
int temp;
static short toggle = 0;
if (P2IFG & BIT6) { // S1
5CCE    f2b040001d02    BIT.B    #0x0040, &0x021d
5CD4    0924            JZ       0x05ce8
ADC12CTL0 |= ADC12ENC; //re-enable encoding
5CD6    a2d30007        BIS.W    #2, &0x0700
ADC12CTL0 |= ADC12SC; // start conversion
5CDA    92d30007        BIS.W    #1, &0x0700
P7OUT |= BIT3;
5CDE    f2d26202        BIS.B    #8, &0x0262
P2IFG &= ~BIT6;
5CE2    f2f0bf001d02    AND.B    #0x00bf, &0x021d
—— main.c —— 182 ——————————————
```

```
    temp = (int)a + 0.5;
    message[13] = '0' + temp/10;
    message[14] = '0' + temp%10;

    halLcdPrintLine(message, 1, OVERWRITE_TEXT);
//      LPM4_EXIT;
__bic_SR_register_on_exit(LPM4_bits);
}


void P2(void)__interrupt[PORT2_VECTOR] {

  int temp;
  static short toggle = 0;

  if (P2IFG & BIT6) { // S1
    ADC12CTL0 |= ADC12ENC; //re-enable encoding
    ADC12CTL0 |= ADC12SC; // start conversion
    P7OUT |= BIT3;
    P2IFG &= ~BIT6;
  }

  if (P2IFG & BIT7) { // S2
    if (toggle) {
      halLcdShutDownBackLight();
    }
    else
```

Memory 1

Address: 005be2 Size: 2

```
0x05be2   0000000011111000
```

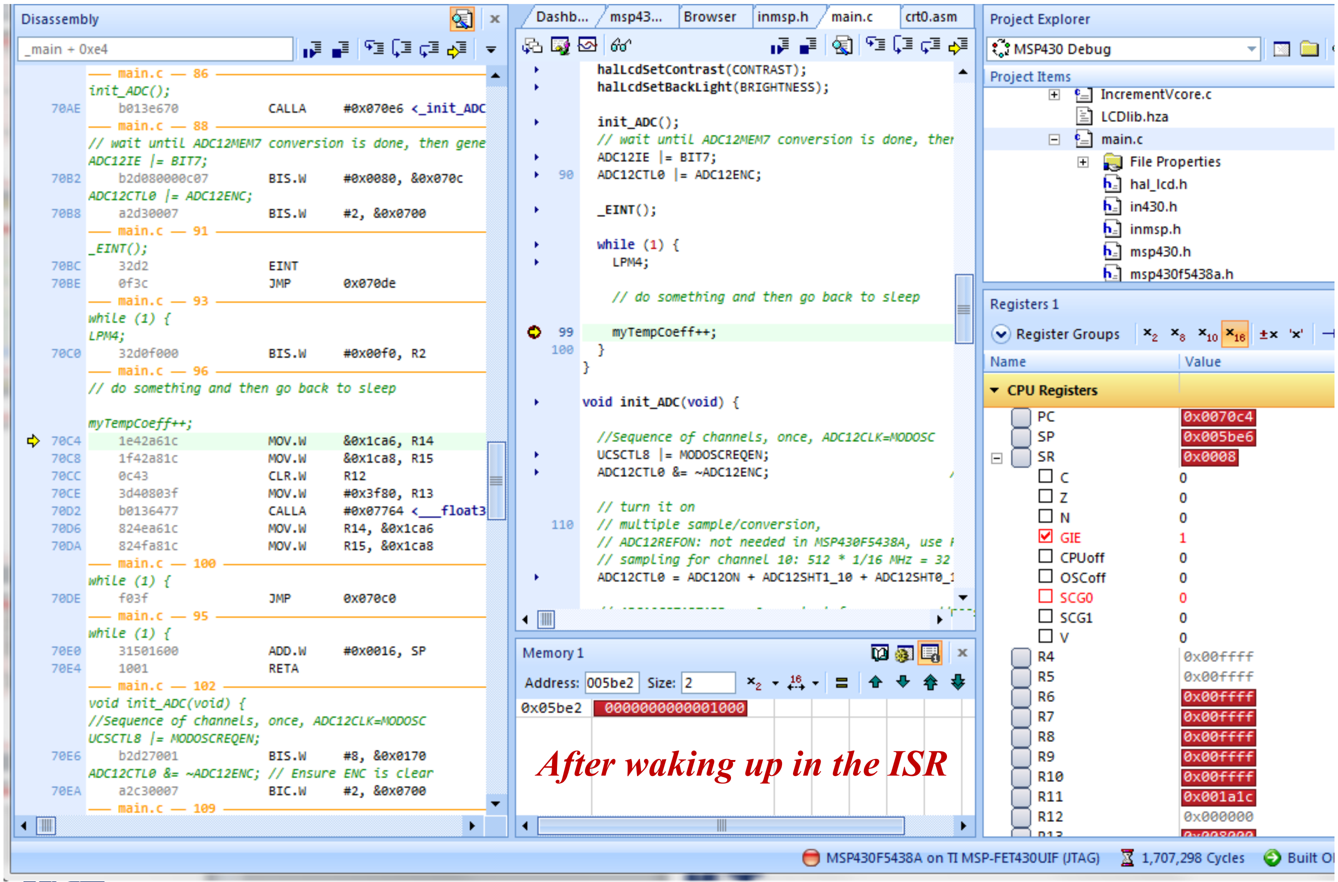
Name	Value
CPU Registers	
PC	0x005cb6
SP	0x005bce
SR	0x0041
☑ C	1
☐ Z	0
☐ N	0
☐ GIE	0
☐ CPUoff	0
☐ OSCoff	0
☑ SCG0	1
☐ SCG1	0
☐ V	0
R4	0x00ffff
R5	0x00ffff
R6	0x006000
R7	0x00450c
R8	0x00bbd6
R9	0x00c250
R10	0x004503
R11	0x00ffcd
R12	0x000000

MSP430F5438A on TI MSP-FET430UIF (JTAG)   1,100,235 Cycles   Built OK   INS

Stopped in the ISR

Operating Modes – Software Configurable

- Active mode (AM): SCG1=0, SCG0=0, OscOff=0, CPUOff=0
  - CPU, MCLK are active.
  - ACLK is active. SMCLK optionally active (SMCLKOFF = 0).
  - DCO is enabled if sources ACLK, MCLK, or SMCLK (SMCLKOFF = 0).
  - FLL is enabled if DCO is enabled.
- Low power mode 0 (LPM0): SCG1=0, SCG0=0, OscOff=0, CPUOff=1
  - CPU, MCLK are disabled.
  - ACLK is active. SMCLK optionally active (SMCLKOFF = 0).
  - DCO is enabled if sources ACLK or SMCLK (SMCLKOFF = 0).
  - FLL is enabled if DCO is enabled
- Low power mode 1 (LPM1): SCG1=0, SCG0=1, OscOff=0, CPUOff=1
  - CPU, MCLK are disabled.
  - ACLK is active. SMCLK optionally active (SMCLKOFF = 0).
  - DCO is enabled if sources ACLK or SMCLK (SMCLKOFF = 0).
  - FLL is disabled.



Disassembly

_main + 0xe4

```
    —— main.c — 86 ————————————————
    init_ADC();
70AE    b013e670        CALLA   #0x070e6 <_init_ADC
    —— main.c — 88 ————————————————
    // wait until ADC12MEM7 conversion is done, then gene
    ADC12IE |= BIT7;
70B2    b2d080000c07    BIS.W   #0x0080, &0x070c
    ADC12CTL0 |= ADC12ENC;
70B8    a2d30007        BIS.W   #2, &0x0700
    —— main.c — 91 ————————————————
    _EINT();
70BC    32d2            EINT
70BE    0f3c            JMP     0x070de
    —— main.c — 93 ————————————————
    while (1) {
    LPM4;
70C0    32d0f000        BIS.W   #0x00f0, R2
    —— main.c — 96 ————————————————
    // do something and then go back to sleep

    myTempCoeff++;
70C4    1e42a61c        MOV.W   &0x1ca6, R14
70C8    1f42a81c        MOV.W   &0x1ca8, R15
70CC    0c43            CLR.W   R12
70CE    3d40803f        MOV.W   #0x3f80, R13
70D2    b0136477        CALLA   #0x07764 <___float3
70D6    824ea61c        MOV.W   R14, &0x1ca6
70DA    824fa81c        MOV.W   R15, &0x1ca8
    —— main.c — 100 ———————————————
    while (1) {
70DE    f03f            JMP     0x070c0
    —— main.c — 95 ————————————————
    while (1) {
70E0    31501600        ADD.W   #0x0016, SP
70E4    1001            RETA
    —— main.c — 102 ———————————————
    void init_ADC(void) {
    //Sequence of channels, once, ADC12CLK=MODOSC
    UCSCTL8 |= MODOSCREQEN;
70E6    b2d27001        BIS.W   #8, &0x0170
    ADC12CTL0 &= ~ADC12ENC; // Ensure ENC is clear
70EA    a2c30007        BIC.W   #2, &0x0700
    —— main.c — 109 ———————————————
```

```
  halLcdSetContrast(CONTRAST);
  halLcdSetBackLight(BRIGHTNESS);

  init_ADC();
  // wait until ADC12MEM7 conversion is done, ther
  ADC12IE |= BIT7;
  ADC12CTL0 |= ADC12ENC;

  _EINT();

  while (1) {
    LPM4;

    // do something and then go back to sleep

    myTempCoeff++;
  }
}

void init_ADC(void) {

  //Sequence of channels, once, ADC12CLK=MODOSC
  UCSCTL8 |= MODOSCREQEN;
  ADC12CTL0 &= ~ADC12ENC;

  // turn it on
  // multiple sample/conversion,
  // ADC12REFON: not needed in MSP430F5438A, use
  // sampling for channel 10: 512 * 1/16 MHz = 32
  ADC12CTL0 = ADC12ON + ADC12SHT1_10 + ADC12SHT0_1
```

Memory 1

Address: 005be2 Size: 2

```
0x05be2   0000000000001000
```

After waking up in the ISR

Project Explorer

MSP430 Debug

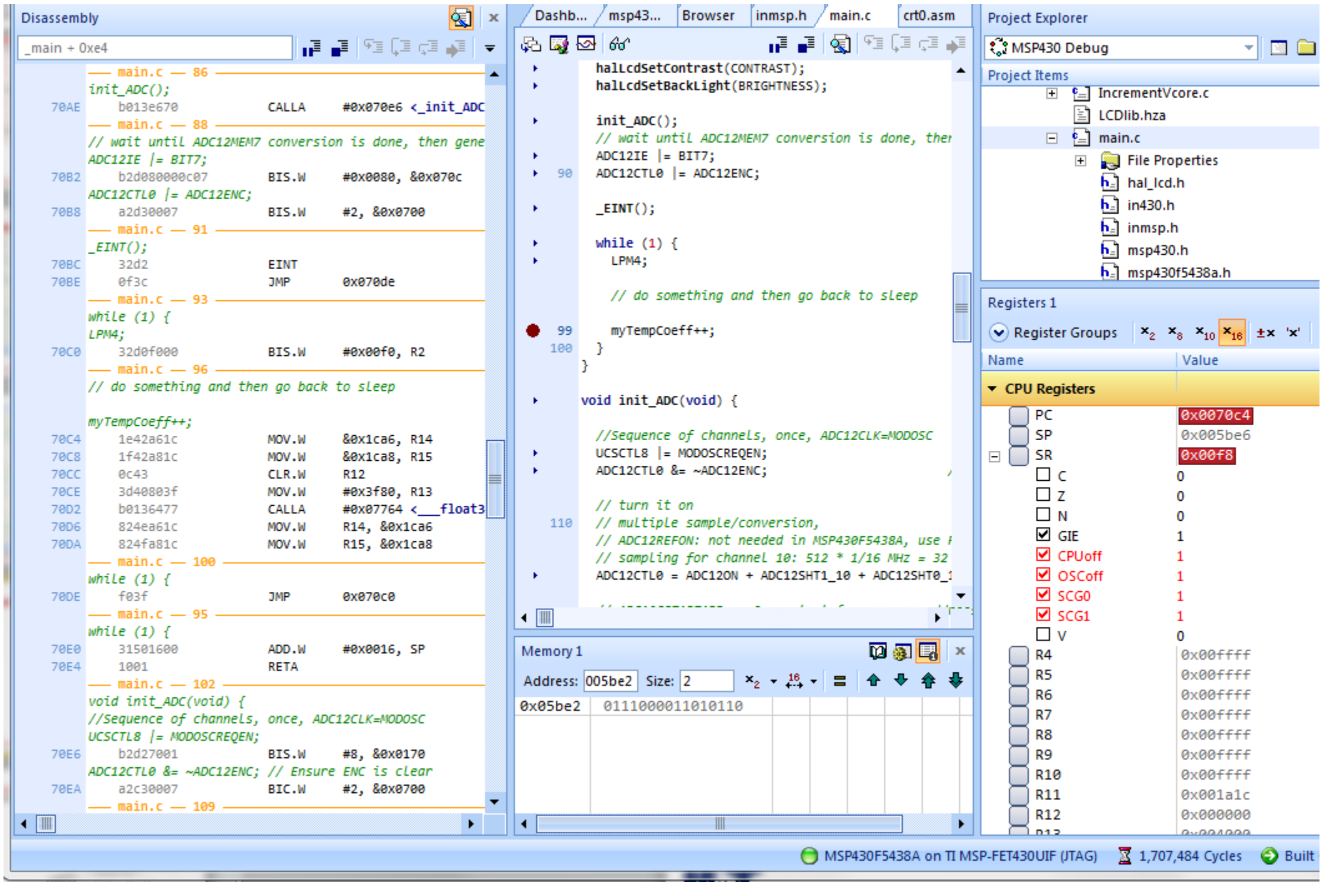
Project Items

Registers 1

Register Groups

```
Name            Value
▾ CPU Registers
  PC            0x0070c4
  SP            0x005be6
  SR            0x0008
    ☐ C         0
    ☐ Z         0
    ☐ N         0
    ☑ GIE       1
    ☐ CPUoff    0
    ☐ OSCoff    0
    ☐ SCG0      0
    ☐ SCG1      0
    ☐ V         0
  R4            0x00ffff
  R5            0x00ffff
  R6            0x00ffff
  R7            0x00ffff
  R8            0x00ffff
  R9            0x00ffff
  R10           0x00ffff
  R11           0x001a1c
  R12           0x000000
```

MSP430F5438A on TI MSP-FET430UIF (JTAG) 1,707,298 Cycles Built OI



```
        —— main.c — 86 ——————————————————————
        init_ADC();
 70AE     b013e670            CALLA    #0x070e6 <_init_ADC
        —— main.c — 88 ——————————————————————
        // wait until ADC12MEM7 conversion is done, then gene
        ADC12IE |= BIT7;
 70B2     b2d080000c07        BIS.W    #0x0080, &0x070c
        ADC12CTL0 |= ADC12ENC;
 70B8     a2d30007            BIS.W    #2, &0x0700
        —— main.c — 91 ——————————————————————
        _EINT();
 70BC     32d2                EINT
 70BE     0f3c                JMP      0x070de
        —— main.c — 93 ——————————————————————
        while (1) {
        LPM4;
 70C0     32d0f000            BIS.W    #0x00f0, R2
        —— main.c — 96 ——————————————————————
        // do something and then go back to sleep

        myTempCoeff++;
 70C4     1e42a61c            MOV.W    &0x1ca6, R14
 70C8     1f42a81c            MOV.W    &0x1ca8, R15
 70CC     0c43                CLR.W    R12
 70CE     3d40803f            MOV.W    #0x3f80, R13
 70D2     b0136477            CALLA    #0x07764 <___float3
 70D6     824ea61c            MOV.W    R14, &0x1ca6
 70DA     824fa81c            MOV.W    R15, &0x1ca8
        —— main.c — 100 —————————————————————
        while (1) {
 70DE     f03f                JMP      0x070c0
        —— main.c — 95 ——————————————————————
        while (1) {
 70E0     31501600            ADD.W    #0x0016, SP
 70E4     1001                RETA
        —— main.c — 102 —————————————————————
        void init_ADC(void) {
        //Sequence of channels, once, ADC12CLK=MODOSC
        UCSCTL8 |= MODOSCREQEN;
 70E6     b2d27001            BIS.W    #8, &0x0170
        ADC12CTL0 &= ~ADC12ENC; // Ensure ENC is clear
 70EA     a2c30007            BIC.W    #2, &0x0700
        —— main.c — 109 —————————————————————
```

```
  halLcdSetContrast(CONTRAST);
  halLcdSetBackLight(BRIGHTNESS);

  init_ADC();
  // wait until ADC12MEM7 conversion is done, then
  ADC12IE |= BIT7;
  ADC12CTL0 |= ADC12ENC;

  _EINT();

  while (1) {
    LPM4;

    // do something and then go back to sleep

    myTempCoeff++;
  }
}

void init_ADC(void) {

  //Sequence of channels, once, ADC12CLK=MODOSC
  UCSCTL8 |= MODOSCREQEN;
  ADC12CTL0 &= ~ADC12ENC;

  // turn it on
  // multiple sample/conversion,
  // ADC12REFON: not needed in MSP430F5438A, use
  // sampling for channel 10: 512 * 1/16 MHz = 32
  ADC12CTL0 = ADC12ON + ADC12SHT1_10 + ADC12SHT0_1
```

Address: 005be2 Size: 2

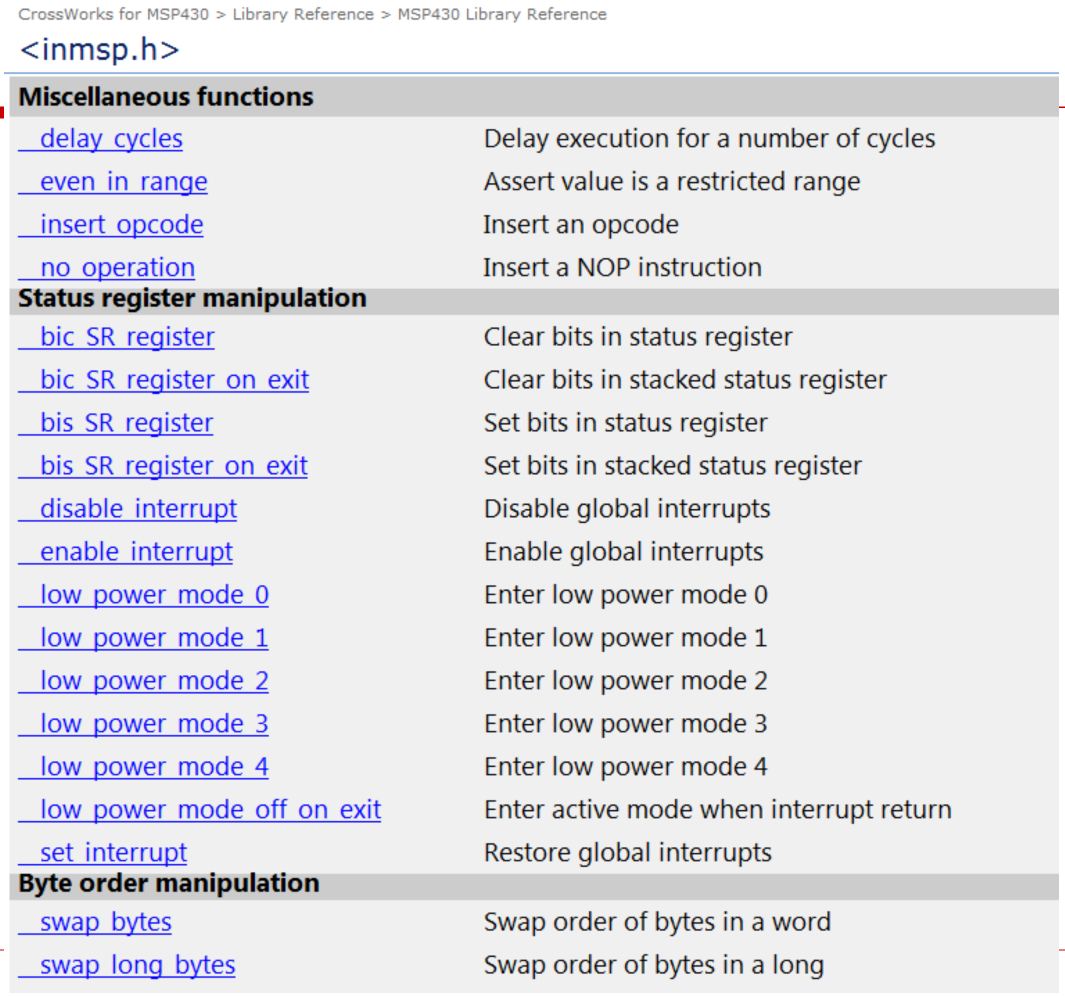
```
0x05be2   0111000011010110
```

- IncrementVcore.c
- LCDlib.hza
- main.c
  - File Properties
  - hal_lcd.h
  - in430.h
  - inmsp.h
  - msp430.h
  - msp430f5438a.h

Name	Value
CPU Registers	
PC	0x0070c4
SP	0x005be6
SR	0x00f8
□ C	0
□ Z	0
□ N	0
☑ GIE	1
☑ CPUoff	1
☑ OSCoff	1
☑ SCG0	1
☑ SCG1	1
□ V	0
R4	0x00ffff
R5	0x00ffff
R6	0x00ffff
R7	0x00ffff
R8	0x00ffff
R9	0x00ffff
R10	0x00ffff
R11	0x001a1c
R12	0x000000

MSP430F5438A on TI MSP-FET430UIF (JTAG) 1,707,484 Cycles Built

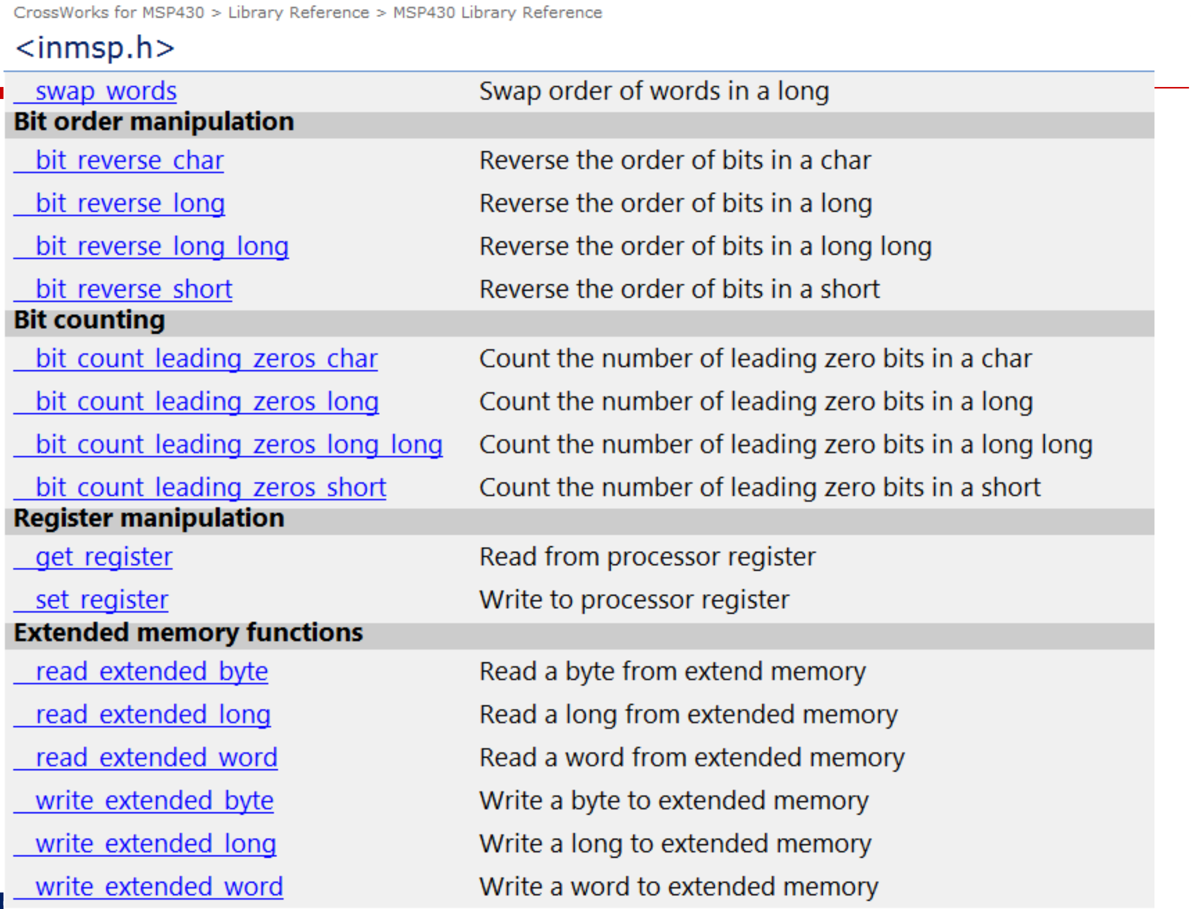
If LPM4_EXIT is commented out in the ISR



CrossWorks for MSP430 > Library Reference > MSP430 Library Reference

<inmsp.h>

Miscellaneous functions	
__delay_cycles	Delay execution for a number of cycles
__even_in_range	Assert value is a restricted range
__insert_opcode	Insert an opcode
__no_operation	Insert a NOP instruction
Status register manipulation	
__bic_SR_register	Clear bits in status register
__bic_SR_register_on_exit	Clear bits in stacked status register
__bis_SR_register	Set bits in status register
__bis_SR_register_on_exit	Set bits in stacked status register
__disable_interrupt	Disable global interrupts
__enable_interrupt	Enable global interrupts
__low_power_mode_0	Enter low power mode 0
__low_power_mode_1	Enter low power mode 1
__low_power_mode_2	Enter low power mode 2
__low_power_mode_3	Enter low power mode 3
__low_power_mode_4	Enter low power mode 4
__low_power_mode_off_on_exit	Enter active mode when interrupt return
__set_interrupt	Restore global interrupts
Byte order manipulation	
__swap_bytes	Swap order of bytes in a word
__swap_long_bytes	Swap order of bytes in a long



CrossWorks for MSP430 > Library Reference > MSP430 Library Reference

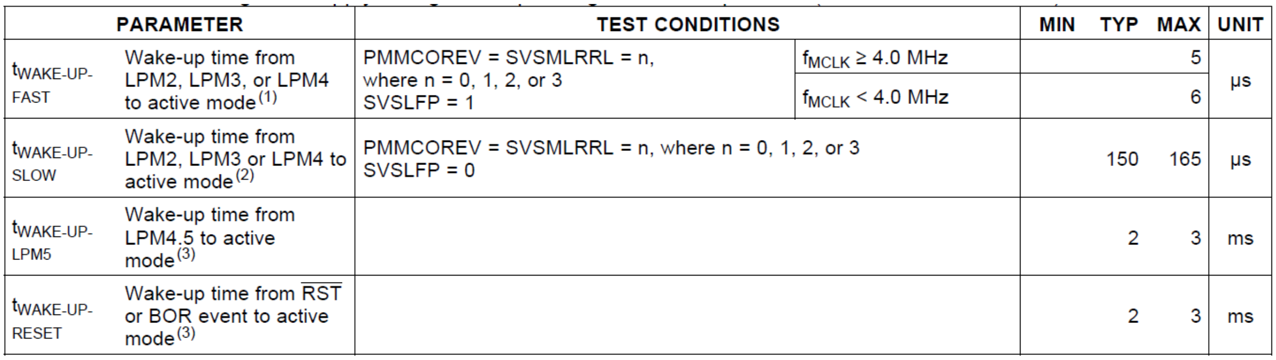
<inmsp.h>

__swap_words	Swap order of words in a long
Bit order manipulation	
__bit_reverse_char	Reverse the order of bits in a char
__bit_reverse_long	Reverse the order of bits in a long
__bit_reverse_long_long	Reverse the order of bits in a long long
__bit_reverse_short	Reverse the order of bits in a short
Bit counting	
__bit_count_leading_zeros_char	Count the number of leading zero bits in a char
__bit_count_leading_zeros_long	Count the number of leading zero bits in a long
__bit_count_leading_zeros_long_long	Count the number of leading zero bits in a long long
__bit_count_leading_zeros_short	Count the number of leading zero bits in a short
Register manipulation	
__get_register	Read from processor register
__set_register	Write to processor register
Extended memory functions	
__read_extended_byte	Read a byte from extend memory
__read_extended_long	Read a long from extended memory
__read_extended_word	Read a word from extended memory
__write_extended_byte	Write a byte to extended memory
__write_extended_long	Write a long to extended memory
__write_extended_word	Write a word to extended memory

Low Power Modes x.5

- x5438 has only LPM4.5
- LPM4.5 wakeup input capability is available for all bits of ports P1 and P2.
- On devices that have it, LPM3.5 power mode allows for RTC mode operation at the lowest power consumption available.
- LPMx.5 disables the voltage regulator that supplies voltage to core
  - Consequently, all registers and SRAM contents are lost
  - Exit from LPMx.5 forces a complete reset
- The wakeup time from LPMx.5 is significantly longer than the wakeup time from the other power modes
  - Primarily because the core voltage needs to be regenerated

Low Power Modes x.5



PARAMETER		TEST CONDITIONS		MIN	TYP	MAX	UNIT
$t_{WAKE\text{-}UP\text{-}FAST}$	Wake-up time from LPM2, LPM3, or LPM4 to active mode [1]	PMMCOREV = SVSMLRRL = n, where n = 0, 1, 2, or 3 SVSLFP = 1	$f_{MCLK} \geq 4.0$ MHz			5	μs
			$f_{MCLK} < 4.0$ MHz			6	
$t_{WAKE\text{-}UP\text{-}SLOW}$	Wake-up time from LPM2, LPM3 or LPM4 to active mode [2]	PMMCOREV = SVSMLRRL = n, where n = 0, 1, 2, or 3 SVSLFP = 0			150	165	μs
$t_{WAKE\text{-}UP\text{-}LPM5}$	Wake-up time from LPM4.5 to active mode [3]				2	3	ms
$t_{WAKE\text{-}UP\text{-}RESET}$	Wake-up time from $\overline{RST}$ or BOR event to active mode [3]				2	3	ms

Low Power Modes x.5

- Events that can wake up the MSP430 from LPMx.5:
  - RST event,
  - a power on cycle, or
  - a specific I/O event
- I/O must be configured properly in advance:
  - Set all ports to general purpose
  - Ensure there are no floating inputs [input high impedance, input with pulldown, input with pullup, output high (low or high drive strength), or output low (low or high drive strength)]
  - If you want I/O port event to wake you up, set one (or more) of them properly (must be P1 or P2) – enable interrupts (PxIE), set the edge (PxIES)
- Enter LPMx.5

```
MOV.B #PMMPW_H, &PMMCTL0_H    ; Open PMM registers for write
BIS.B #PMMREGOFF, &PMMCTL0_L  ;
BIS #GIE+CPUOFF+OSCOFF+SCG1+SCG0,SR ; Enter LPMx.5 when PMMREGOFF is set
```

Low Power Modes x.5

- After exiting LPMx.5, program execution continues from the address stored in the system reset vector location 0FFFEh (after boot code)
- The PMMLPM5IFG bit is set indicating that the device was in LPMx.5
- During LPMx.5, all I/O pin conditions are automatically locked to the current state.
- Upon exit from LPMx.5, the I/O pin conditions remain locked until the application unlocks them

Here’s what TI says about low power

- The most important factor – use clock system to maximize time in LPM3 or 4 (< 2 μA with 32 kHz for ACLK)
  - Use interrupts to wake the processor and control program flow.
  - Peripherals should be switched on only when needed.
  - Use low-power integrated peripheral modules in place of software driven functions.
  - Calculated branching and fast table look-ups should be used in place of flag polling and long software calculations.
  - Avoid frequent subroutine and function calls due to overhead.
  - For longer software routines, single-cycle CPU registers should be used.

Connection of unused pins

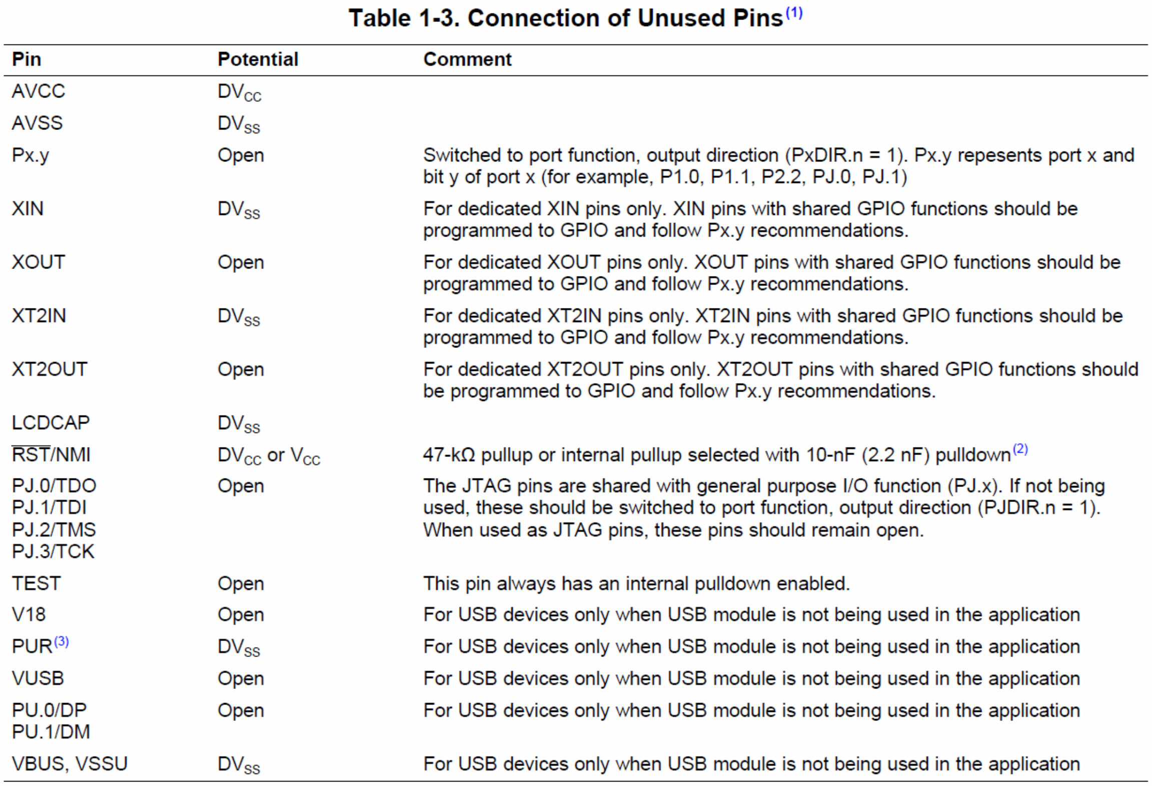


Table 1-3. Connection of Unused Pins[1]

Pin	Potential	Comment
AVCC	DV_{CC}	
AVSS	DV_{SS}	
Px.y	Open	Switched to port function, output direction (PxDIR.n = 1). Px.y repesents port x and bit y of port x (for example, P1.0, P1.1, P2.2, PJ.0, PJ.1)
XIN	DV_{SS}	For dedicated XIN pins only. XIN pins with shared GPIO functions should be programmed to GPIO and follow Px.y recommendations.
XOUT	Open	For dedicated XOUT pins only. XOUT pins with shared GPIO functions should be programmed to GPIO and follow Px.y recommendations.
XT2IN	DV_{SS}	For dedicated XT2IN pins only. XT2IN pins with shared GPIO functions should be programmed to GPIO and follow Px.y recommendations.
XT2OUT	Open	For dedicated XT2OUT pins only. XT2OUT pins with shared GPIO functions should be programmed to GPIO and follow Px.y recommendations.
LCDCAP	DV_{SS}	
$\overline{RST}$/NMI	DV_{CC} or V_{CC}	47-kΩ pullup or internal pullup selected with 10-nF (2.2 nF) pulldown[2]
PJ.0/TDO PJ.1/TDI PJ.2/TMS PJ.3/TCK	Open	The JTAG pins are shared with general purpose I/O function (PJ.x). If not being used, these should be switched to port function, output direction (PJDIR.n = 1). When used as JTAG pins, these pins should remain open.
TEST	Open	This pin always has an internal pulldown enabled.
V18	Open	For USB devices only when USB module is not being used in the application
PUR[3]	DV_{SS}	For USB devices only when USB module is not being used in the application
VUSB	Open	For USB devices only when USB module is not being used in the application
PU.0/DP PU.1/DM	Open	For USB devices only when USB module is not being used in the application
VBUS, VSSU	DV_{SS}	For USB devices only when USB module is not being used in the application

MSP430x4xx is powered directly from a 2.6V 1800mAh battery and is operating at 4 MHz using a 32KHz crystal (i.e. MCLK = 4MHz, ACLK = 32KHz).

a) Ignoring the other components of the system and assuming that the battery voltage is going to remain constant, **calculate the battery life in days,** if the system performs the following tasks:

Microcontroller wakes up every 1 ms to update the real-time clock, spending on average 16 clock cycles in the active mode every time it wakes up; after that, it goes back to LPM0 mode;

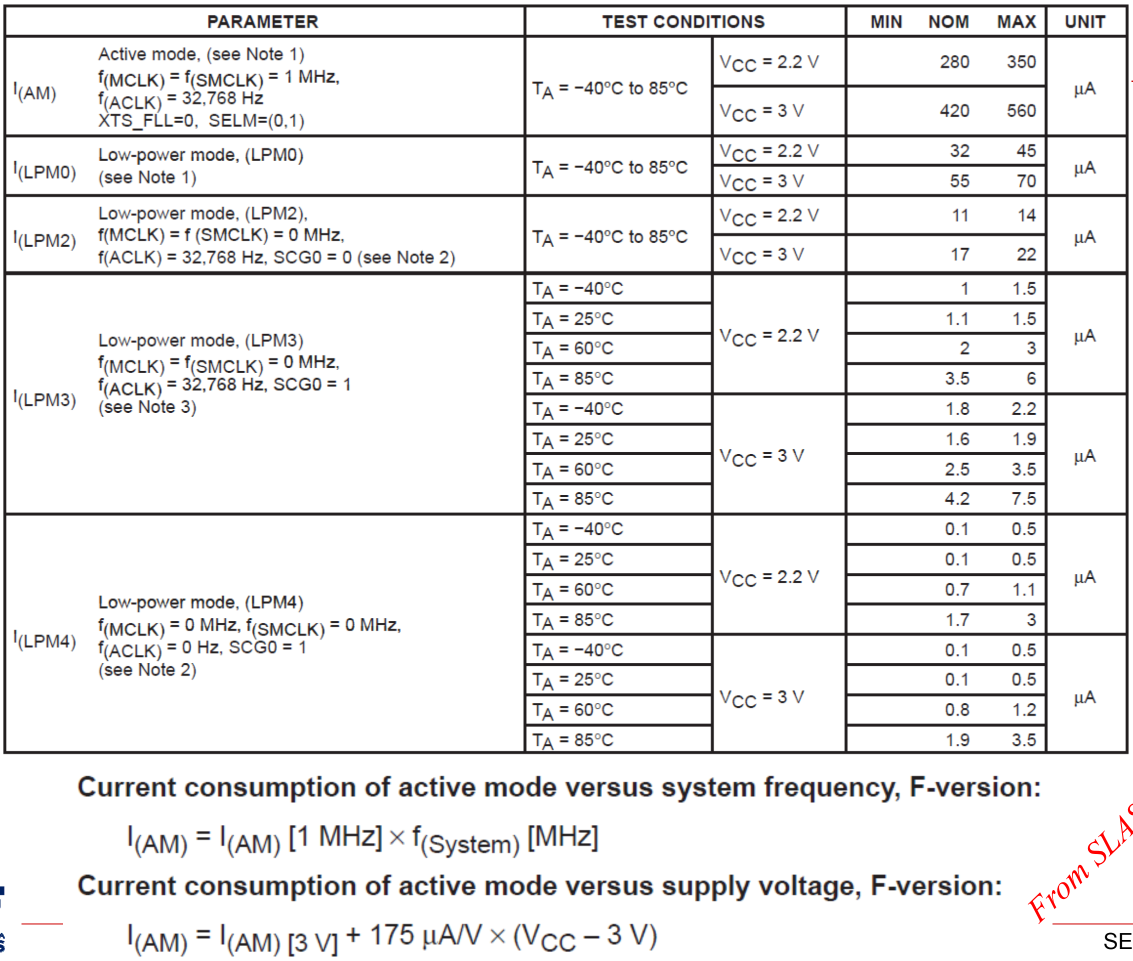
Microcontroller also wakes up every 100 ms (using a lower-priority interrupt than the one for the real-time-clock) and spends 176 clock cycles in the active mode, processing and moving data; after that, it goes back to LPM0 mode;

b) What is the percentage of time the microcontroller spends in LPM0 mode?

c) What would be the battery life without using the low-power mode?

Assume that it takes 6 μs for the microcontroller to wake up and that the current consumption while waking up is equal to the current consumption in the active mode.

(*Note: Use* ***the maximum values*** *for the current consumptions at given frequency/voltage from the appropriate table in the document SLAS344*)



PARAMETER		TEST CONDITIONS		MIN	NOM	MAX	UNIT
$I_{(AM)}$	Active mode, (see Note 1) $f_{(MCLK)} = f_{(SMCLK)}$ = 1 MHz, $f_{(ACLK)}$ = 32,768 Hz XTS_FLL=0, SELM=(0,1)	T_A = −40°C to 85°C	V_{CC} = 2.2 V		280	350	μA
			V_{CC} = 3 V		420	560	
$I_{(LPM0)}$	Low-power mode, (LPM0) (see Note 1)	T_A = −40°C to 85°C	V_{CC} = 2.2 V		32	45	μA
			V_{CC} = 3 V		55	70	
$I_{(LPM2)}$	Low-power mode, (LPM2), f(MCLK) = f (SMCLK) = 0 MHz, f(ACLK) = 32,768 Hz, SCG0 = 0 (see Note 2)	T_A = −40°C to 85°C	V_{CC} = 2.2 V		11	14	μA
			V_{CC} = 3 V		17	22	
$I_{(LPM3)}$	Low-power mode, (LPM3) $f_{(MCLK)} = f_{(SMCLK)}$ = 0 MHz, $f_{(ACLK)}$ = 32,768 Hz, SCG0 = 1 (see Note 3)	T_A = −40°C	V_{CC} = 2.2 V		1	1.5	μA
		T_A = 25°C			1.1	1.5	
		T_A = 60°C			2	3	
		T_A = 85°C			3.5	6	
		T_A = −40°C	V_{CC} = 3 V		1.8	2.2	μA
		T_A = 25°C			1.6	1.9	
		T_A = 60°C			2.5	3.5	
		T_A = 85°C			4.2	7.5	
$I_{(LPM4)}$	Low-power mode, (LPM4) $f_{(MCLK)}$ = 0 MHz, $f_{(SMCLK)}$ = 0 MHz, $f_{(ACLK)}$ = 0 Hz, SCG0 = 1 (see Note 2)	T_A = −40°C	V_{CC} = 2.2 V		0.1	0.5	μA
		T_A = 25°C			0.1	0.5	
		T_A = 60°C			0.7	1.1	
		T_A = 85°C			1.7	3	
		T_A = −40°C	V_{CC} = 3 V		0.1	0.5	μA
		T_A = 25°C			0.1	0.5	
		T_A = 60°C			0.8	1.2	
		T_A = 85°C			1.9	3.5	

Current consumption of active mode versus system frequency, F-version:

$$I_{(AM)} = I_{(AM)}\,[1\text{ MHz}] \times f_{(System)}\,[\text{MHz}]$$

Current consumption of active mode versus supply voltage, F-version:

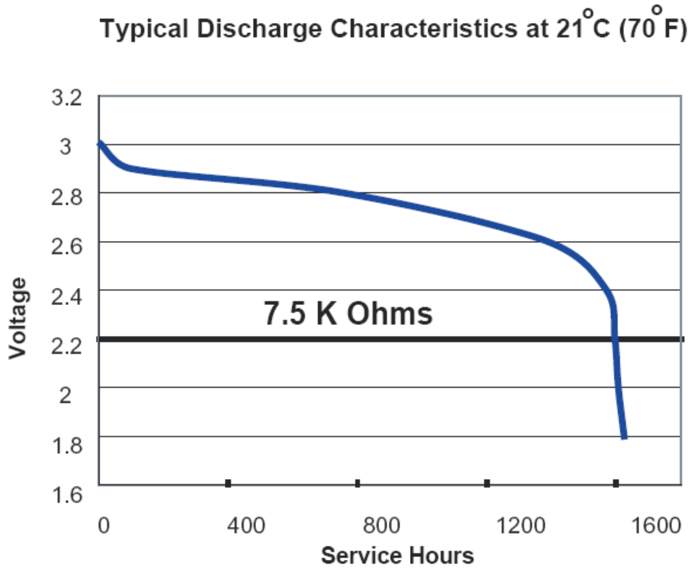
$$I_{(AM)} = I_{(AM)\,[3\text{ V}]} + 175\ \mu\text{A/V} \times (V_{CC} - 3\text{ V})$$

From SLAS344

Duracell 2450 @ 21 C

Manufacturer’s discharge current and your discharge current might differ

Discharge current is between
3 V/7.5 KΩ = 0.4 mA and
2.2 V / 7.5 kΩ = 0.3 mA
⇒Very low discharge current

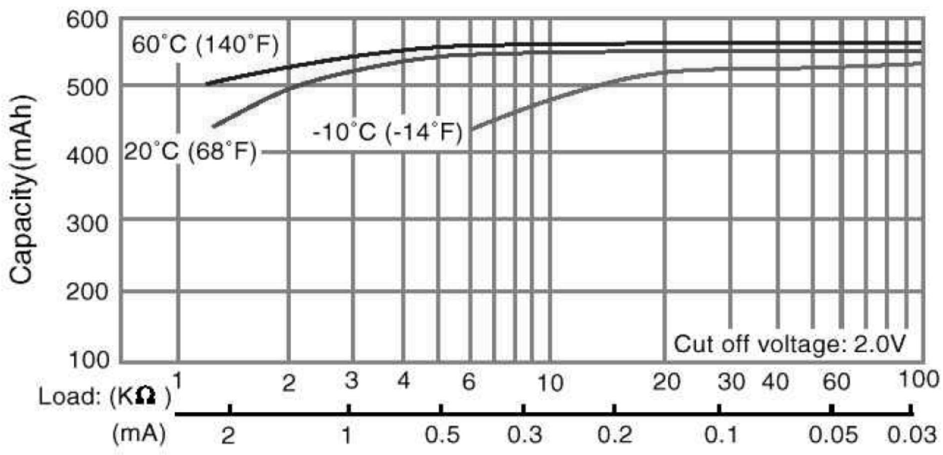


Typical Discharge Characteristics at 21°C (70°F)

Capacity vs. Dicharge Rate for Panasonic lithium CR2354

Max capacity 560 mAh

The higher the discharge current,
less capacity is utilized from the battery



The effects of a switching regulator

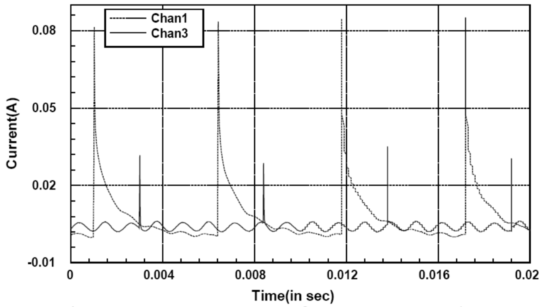


Figure 4. Current Output from CR2354 (Chan1) and DC/DC Converter (Chan3) during Sleep Mode – Snap Shot

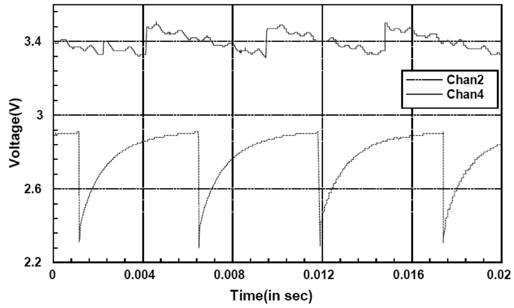


Figure 5. Voltage Output from CR2354 (Chan2) and DC/DC Converter (Chan4) during Sleep Mode – Snap Shot

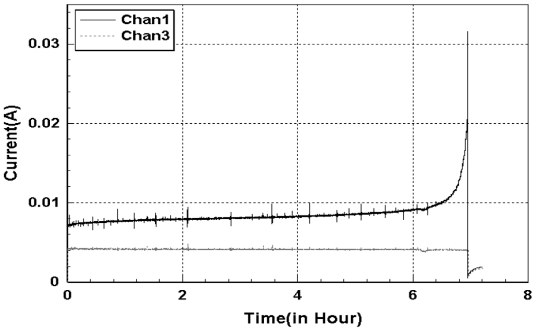


Figure 6. Average Current Output from CR2354 (Chan1) and DC/DC Converter (Chan3) during Sleep Mode - Complete

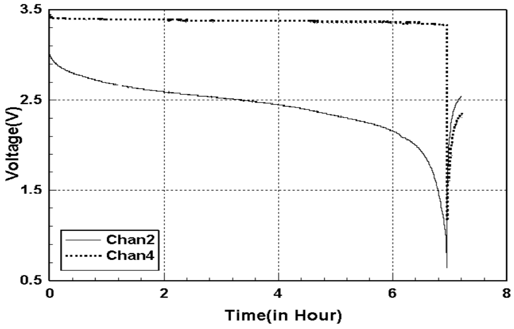
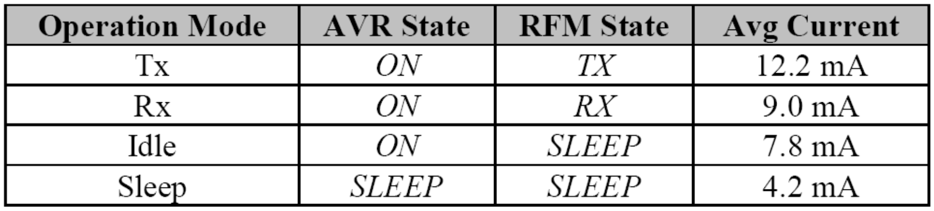


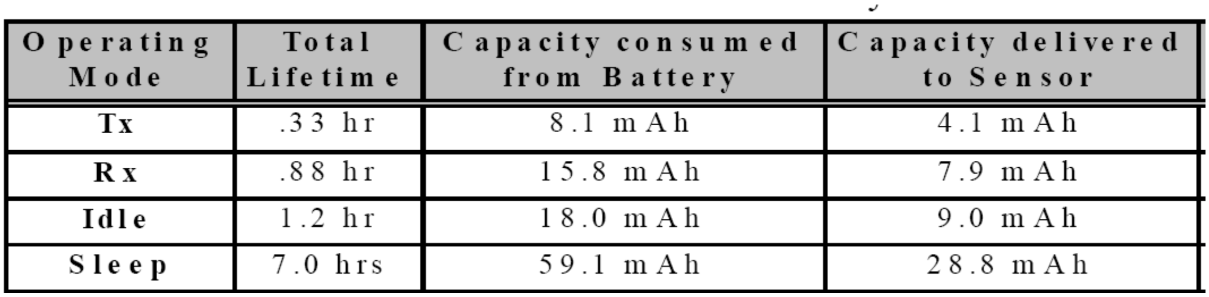
Figure 7. Average Voltage Output from CR2354 (Chan2) and DC/DC Converter (Chan4) during Sleep Mode – Complete

Shocking results

Table 1. Node States And Current Consumption at 3.3V



Operation Mode	AVR State	RFM State	Avg Current
Tx	*ON*	*TX*	12.2 mA
Rx	*ON*	*RX*	9.0 mA
Idle	*ON*	*SLEEP*	7.8 mA
Sleep	*SLEEP*	*SLEEP*	4.2 mA



Operating Mode	Total Lifetime	Capacity consumed from Battery	Capacity delivered to Sensor
Tx	.33 hr	8.1 mAh	4.1 mAh
Rx	.88 hr	15.8 mAh	7.9 mAh
Idle	1.2 hr	18.0 mAh	9.0 mAh
Sleep	7.0 hrs	59.1 mAh	28.8 mAh

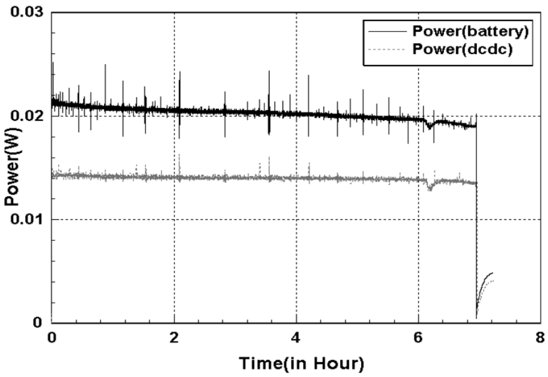


Figure 8. Power Output from CR2354 and DC/DC Converter (Chan3) during Sleep Mode

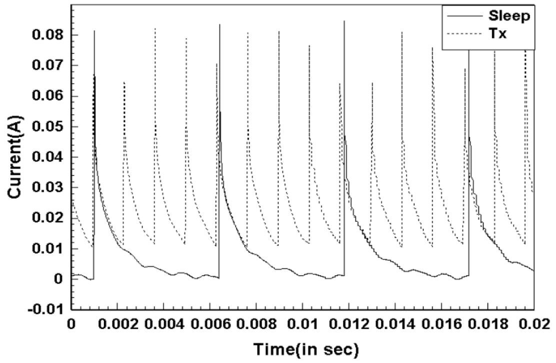


Figure 9. Current drawn from the battery during Sleep Mode and Tx Mode

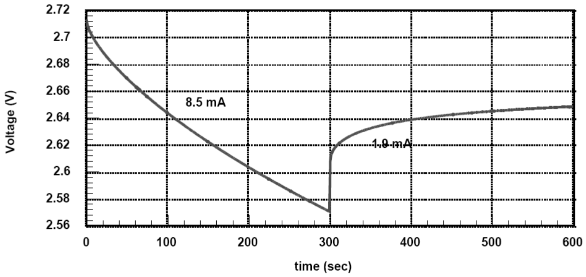


Figure 2. Recovery Effect of Relaxation Model

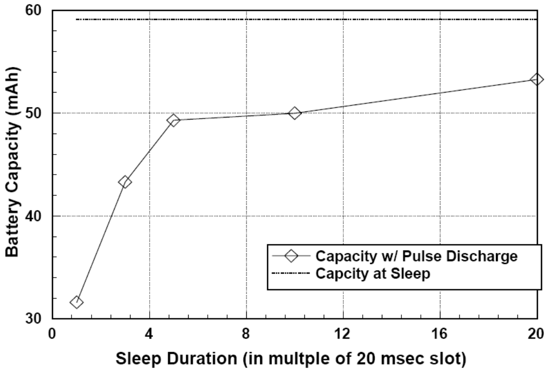
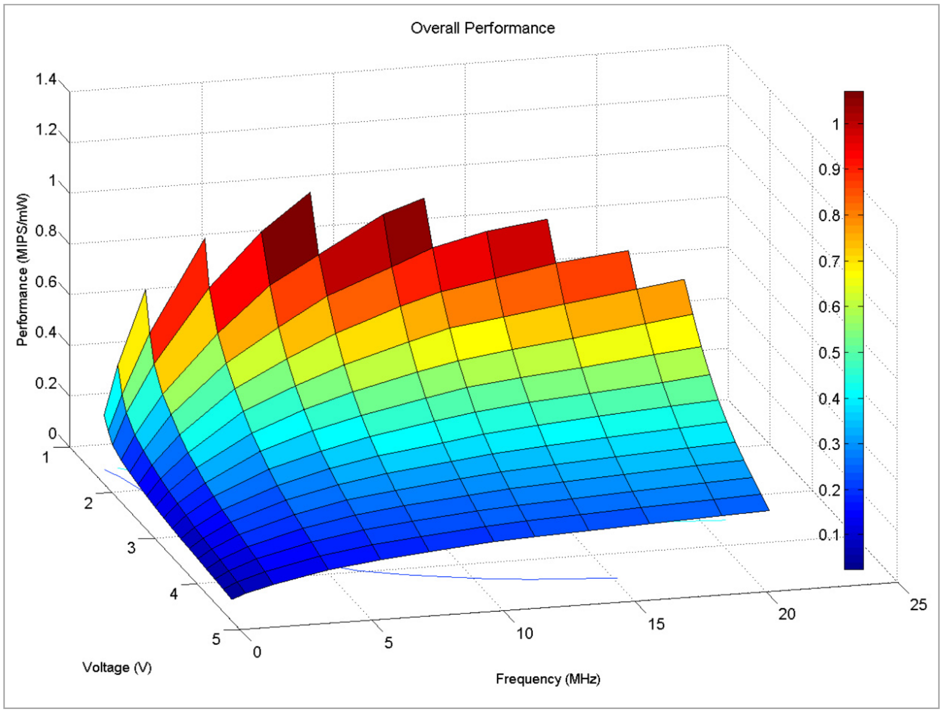
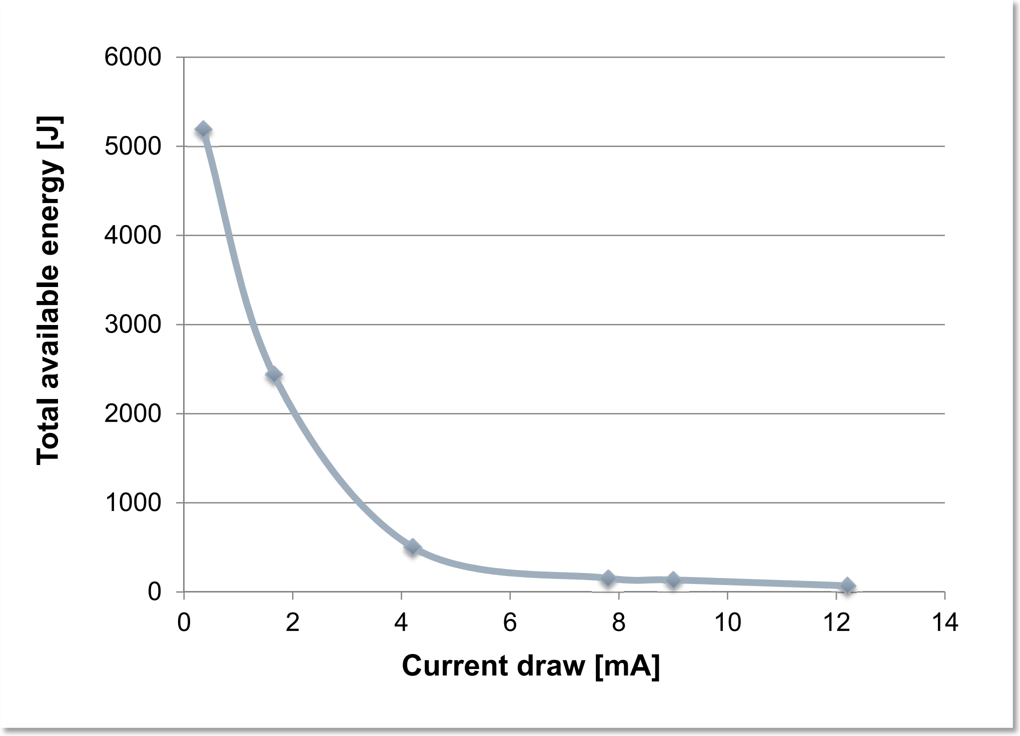


Figure 10. Impact of Pulse Discharging on Battery

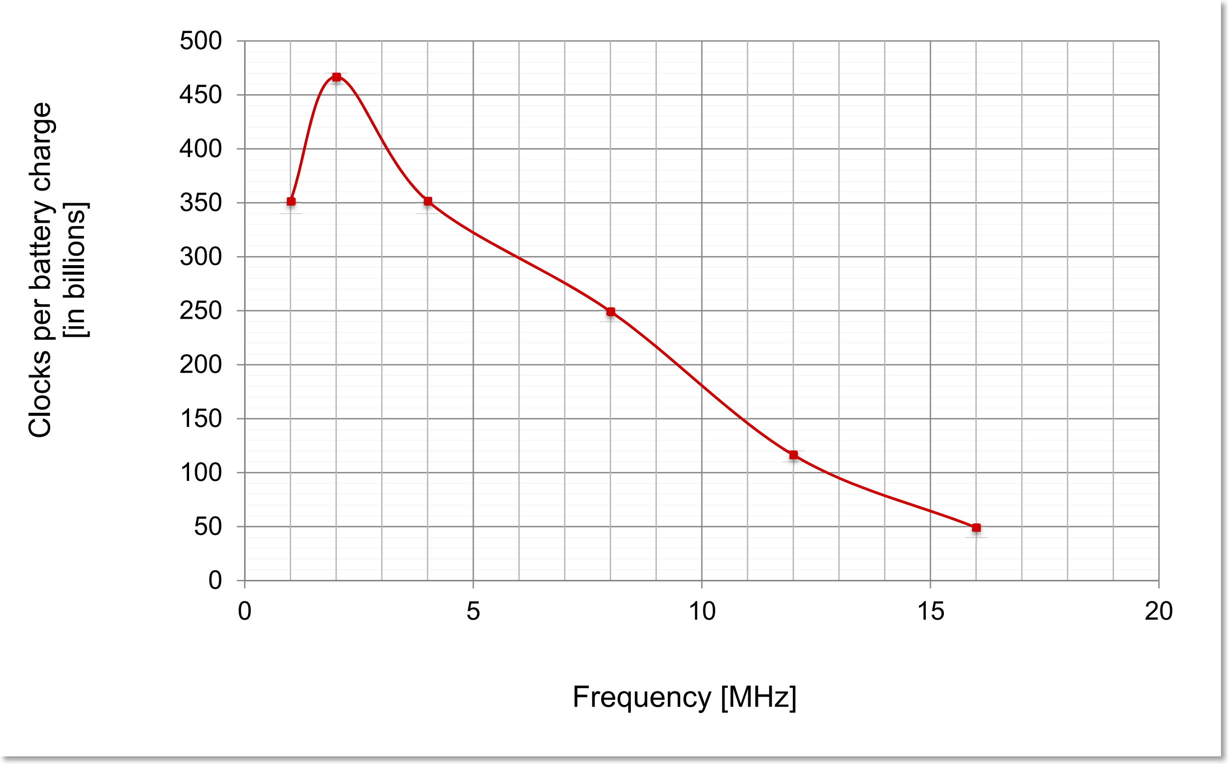
Voltage vs. Frequency vs. Performance for MSP430F2618



Available battery energy vs. average discharge current



Cycles per battery charge vs. clock frequency



References and reading material

- ❑ Trevor Mudge, “Power: A First-Class Architectural Design Constraint,” IEEE Computer, April 2001, pp. 52-58
- ❑ Sung Park, Andreas Savvides, Mani B. Srivastava, “Battery Capacity Measurement And Analysis Using Lithium Coin Cell Battery,” ISLPED ’01, August 6-7, 2001, Huntington Beach, California, USA.

Download from the IEEE Xplore Digital Library (through UAF Rasmuson Library)

- ❑ Mike Mitchel, “Choosing An Ultralow-Power MCU,” TI Application Report, *SLAA207*, June 2004.

Google it

- ❑ MSP430x5xx and MSP430x6xx Family User’s Guide, Texas Instruments (Chapter 1, Sections 1.3 – 1.6)
- ❑ MSP430x543xA, MSP430x541xA Mixed Signal Microcontroller (Datasheet), Texas Instruments, SLAS655

Blackboard