Texas Instruments MSP430FR6989 Launchpad

MCU:
MSP430FR6989IPZ

MSP-EXP430FR6989 LaunchPad Development Kit

MSP430FR6989 Microcontroller

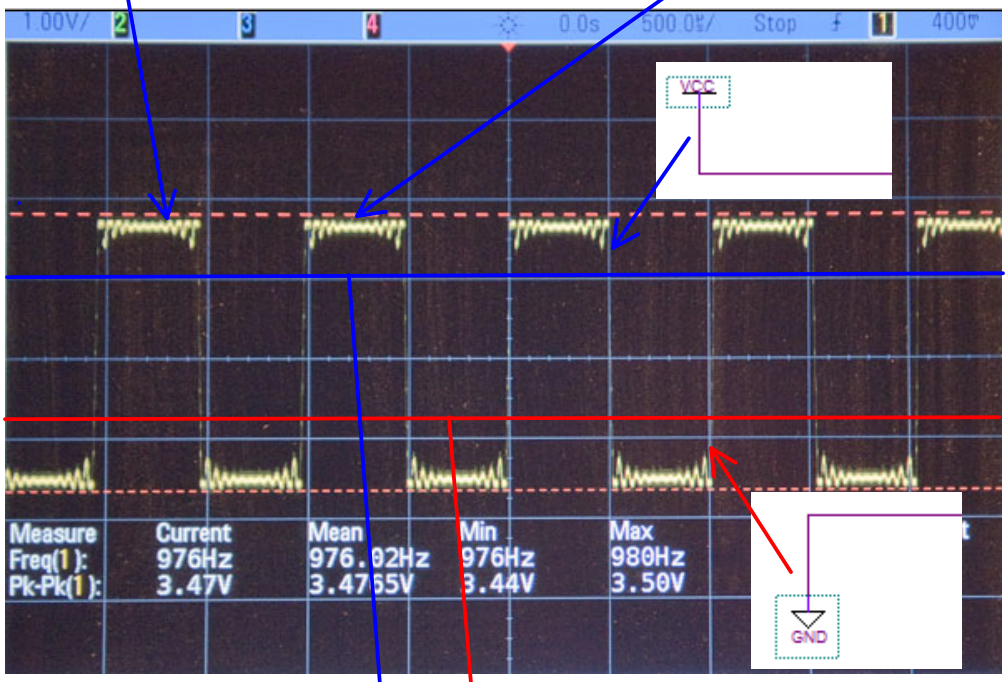
- 1.8-V to 3.6-V operation
- 16-bit RISC architecture up to 16-MHz system clock and 8-MHz FRAM access
- 128KB of nonvolatile FRAM
- 83 GPIOs

Mixed Signal Oscilloscope (MOS)

Analog (voltage) Signal

Actual Voltages

Analog



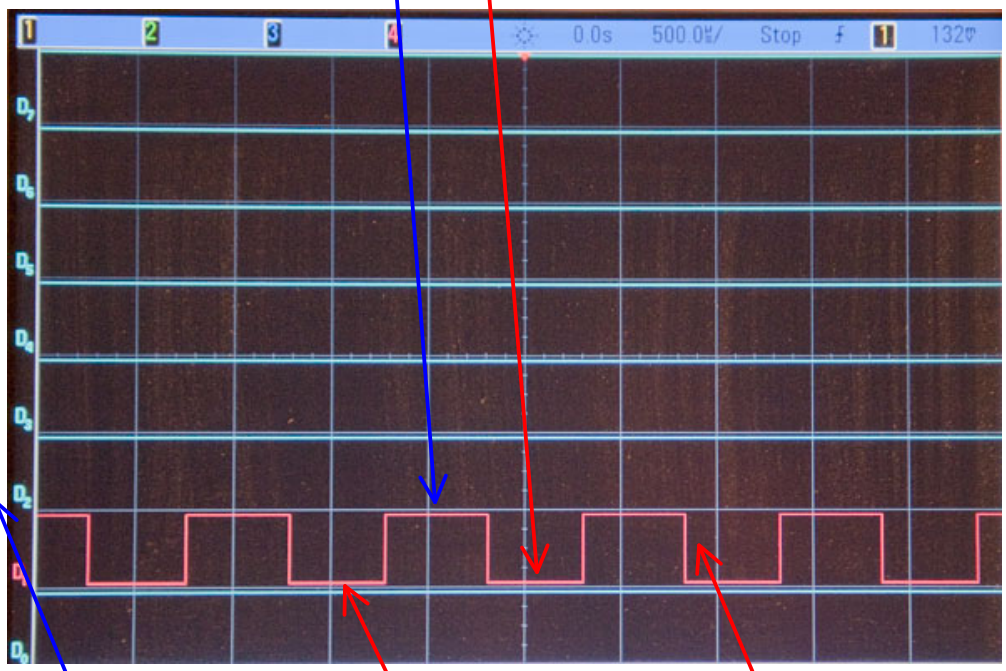
V_H

V_L

Interpretation

Digital signal is Hi or 1 if analog signal voltage > blue voltage

Digital signal is Lo or 0 if analog signal voltage < red voltage



Hi, 1

Lo, 0

Logical Values
not actual Voltages

1 Bit (Binary Digit)

Digital Signal

Digital

Number Systems

Binary numbers
base, radix = 2

$A_{n-1} \quad A_1 A_0 \quad A_{-1} A_{-2} \quad A_{-m}$

$A_i \in \{0, 1\}$ A_i → binary digit (bit)

Binary number
↳ 101101.1101 = $(101101.1101)_2$
bit bit binary point

Converting from binary to decimal

$(110101.11) = 1\times2^5 + 1\times2^4 + 0\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0$
$+ 1\times2^{-1} + 1\times2^{-2} = (53.75)_{10}$
5 4 3 2 1 0 -1 -2

Notation: Code Composer Studio

$(110101)_2 \longrightarrow$ 110101b (b: binary)

msd (most significant digit) → 1 1 0 1 0 1 ← lsd (least significant digit)

decimal integer to binary

$(41)_{10} = (?)_2$

$$\frac{41}{2} = 20 + \frac{1}{2}$$

$$\frac{20}{2} = 10 + \frac{0}{2}$$

$$\frac{10}{2} = 5 + \frac{0}{2}$$

$$\frac{5}{2} = 2 + \frac{1}{2}$$

$$\frac{2}{2} = 1 + \frac{0}{2}$$

$$\frac{1}{2} = 0 + \frac{1}{2}$$

$(1\ 0\ 1\ 0\ 0\ 1)_2$

Hexadecimal numbers

Hexadecimal radix = 16 $16 = 2^4$ radix is a power of 2

useful to represent binary numbers easily in compact form (easier on human beings than binary representation)

10 11 12 13 14 15

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

16 hexadecimal digits

$$(B65F)_{16} = 11\times16^3 + 6\times16^2 + 5\times16^1 + 15\times16^0$$
$$= (46687)_{10}$$

In Code Composer Studio:

zero

$(B65F)_{16} \longrightarrow$ 0XB65F (C style)

$\longrightarrow$ B65Fh (TI style)

Important table for converting bases among binary, octal and hexadecimal number representation

Decimal	Binary	Octal	Hex
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Converting from binary to hexadecimal

$(1011\,0011\,0110\,1011)_2 = (?)_{16}$

(use Table)

0010 1100 0110 1011

2 C 6 B

pad zeros if required

start from here

$= (2C6B)_{16}$

Converting from hexadecimal to binary

$(3A6)_{16} = (?)_2$

(use Table)

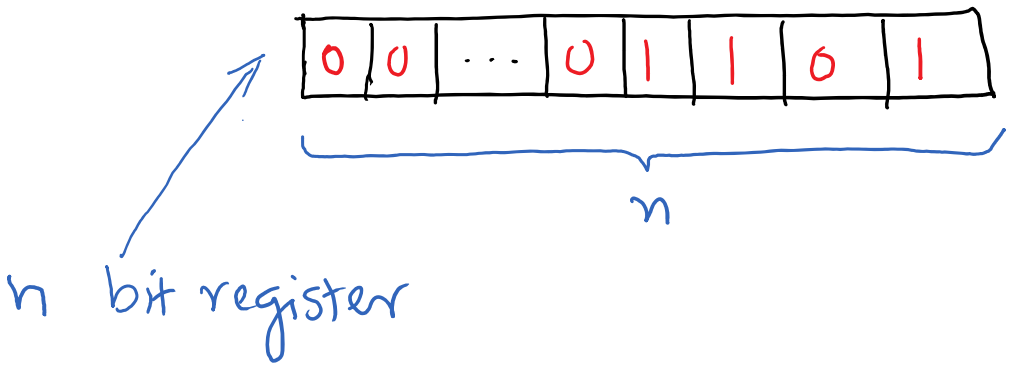
discard

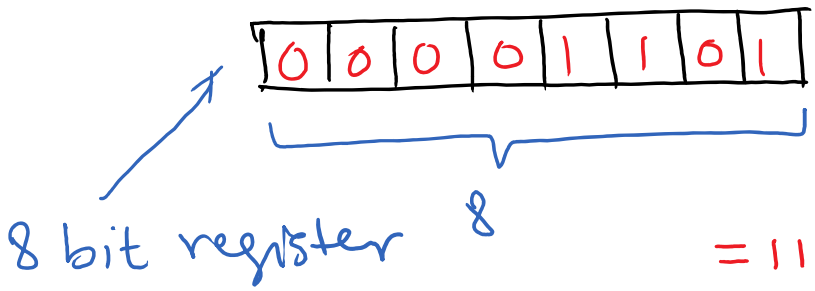
$(3A6)_{16} = 0011\,1010\,0110$

$= (1110100110)_2$

Compliments of Binary numbers.

Binary Number N = 1101 stored in an n bit register.





= 1101 stored in an 8 bit register

N = 1101

n = 8 (word length)

1's Complement of Binary Numbers

register size → n, number → N

1's compliment of N → $(2^n - 1) - N$

Change all the 1's to 0, and all the 0's to 1 in N to get 1's compliment of N

$10110010 \xrightarrow{1\text{'s comp.}} 01001101$

$111 \xrightarrow{1\text{'s comp}} 11111000$

2's compliment of N $\rightarrow$ $\begin{cases} 2^n - N & \text{for } N \neq 0 \\ 0 & \text{for } N = 0 \end{cases}$

2's comp. of N = 1's comp. of N + 1

$101100 \xrightarrow{2\text{'s comp}} 010011 \leftarrow$ 1's comp

$+ \quad 1$

$010100 \leftarrow$ 2's comp.

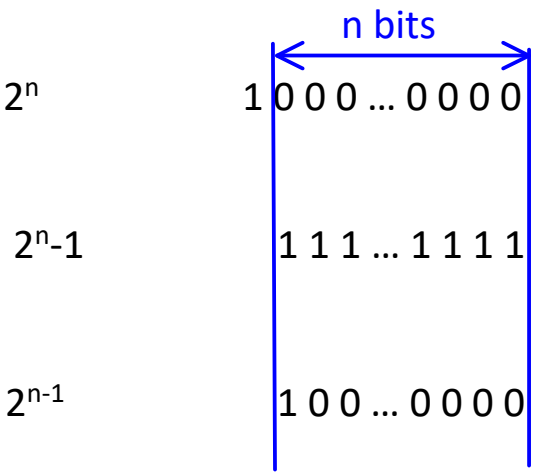
"Trick" to find 2's compliment of N:

Leave all the least significant zeros and the first 1 unchanged – replace all the remaining 0's by 1's, and 1's by 0's.

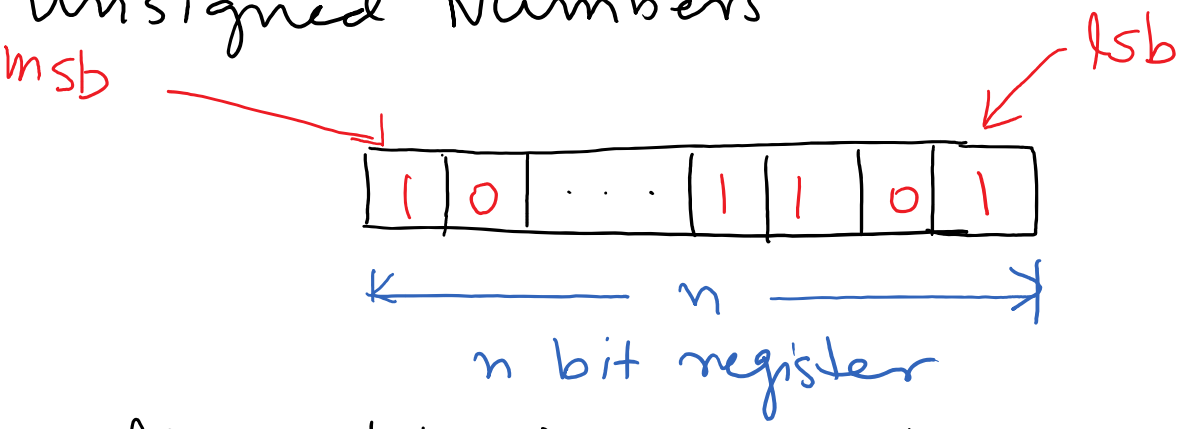
2's comp. of $101100 \longrightarrow 010100$

invert these

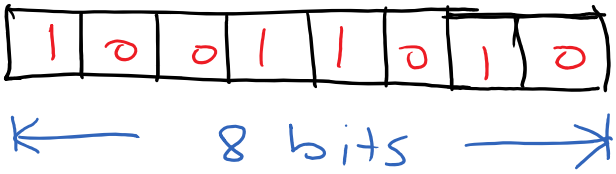
leave these unchanged


n bits
2^n
1 0 0 0 … 0 0 0 0
2^n-1
1 1 1 … 1 1 1 1
2^{n-1}
1 0 0 … 0 0 0 0

Unsigned Numbers

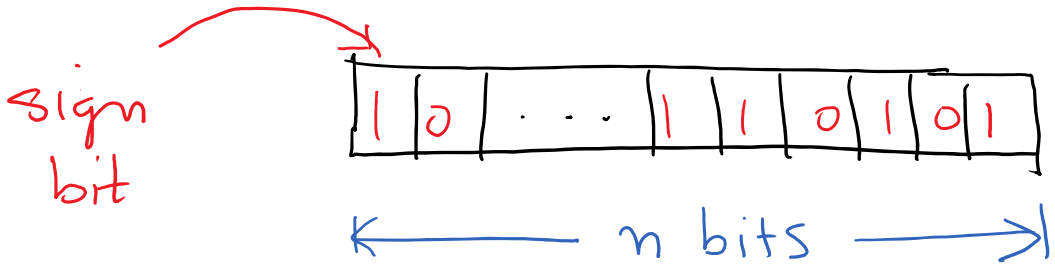


All the bits in the register represent the magnitude of the number.



Number $N = +(10011010)_2$

Signed Numbers
(2's compliment representation)



example 8 bit registers

$N = (9)_{10} \longrightarrow 00001001$

Sign bit
0 = +

magnitude
$= 0001001 = (9)_{10}$

$N = (-9)_{10} \longrightarrow$ 2's comp of 00001001

$= 11110111$

Sign bit
1 = −

Example

11101010

signed bit is 1 ∴ number is −ve

2's comp. of 11101010 is $00010110 = (22)_{10}$

∴ 11101010 is $(-22)_{10}$

00011011

signed bit 0 ∴ number is +ve

00011011 is $(27)_{10}$

ECE2560 Lesson01 Page_13

Signed Binary Addition and subtraction

Addition

Add the two numbers (including the signed bit)
discard the carry out

X = +6	00000110	
Y = +13	00001101	
Z = X+Y = +19	00010011	↖ sign bit
X = -6	11111010	
Y = +13	00001101	
Z = X+Y = +7	(1) 00000111	discard ↖ sign bit
X = +6	00000110	
Y = -13	11110011	
Z = X+Y = -7	11111001	↖ sign bit

2's comp of 11111001 is 00000111 = +7

therefore 11111001 is -7

ECE2560 Lesson01 Page_14

X = -6 1 1 1 1 1 0 1 0

Y = -13 1 1 1 1 0 0 1 1

Z = X + Y = -19 (1) 1 1 1 0 1 1 0 1

discard ↑ sign bit

2's comp of 1 1 1 0 1 1 0 1 is 0 0 0 1 0 0 1 1 = +19

+ 19

therefore 1 1 1 0 1 1 0 1 is -19

Subtraction (2's comp representation)

Z = X − Y

Z = { X + (2's comp of Y)
discard the carry out

X = -6 1 1 1 1 1 0 1 0

Y = -13 → 2's comp of -13 → 0 0 0 0 1 1 0 1 } add

Z = X − Y = +7 (1) 0 0 0 0 0 1 1 1

−6 − (−13) discard ↑ sign bit

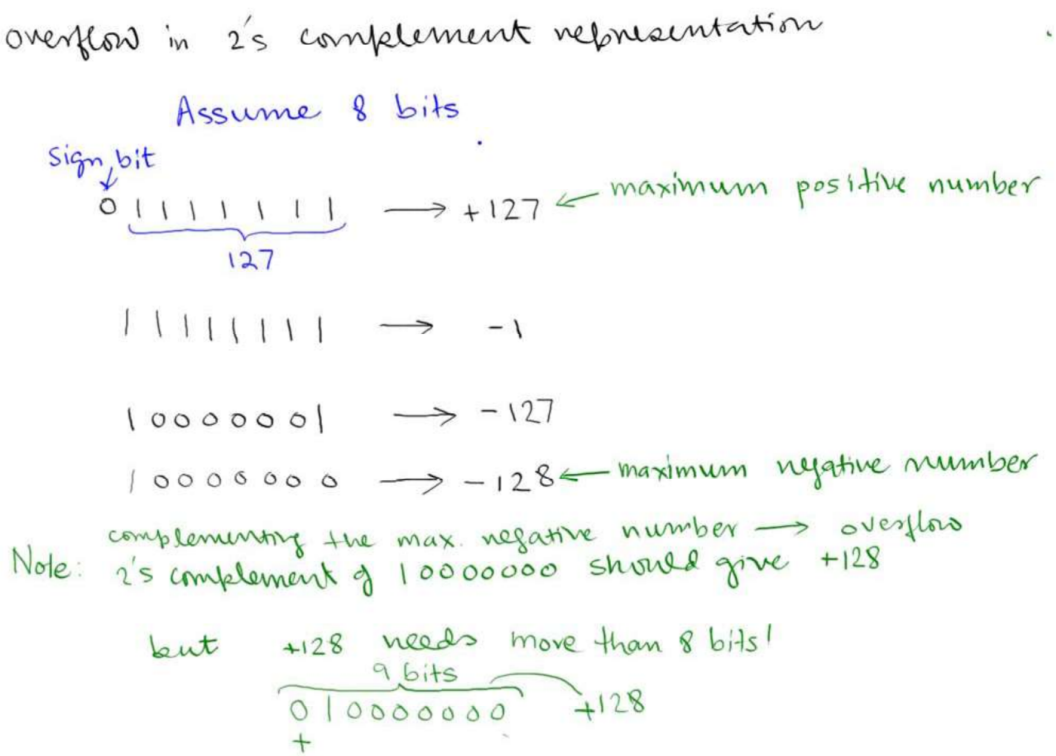
X = +6 0 0 0 0 0 1 1 0

Y = -13 → 2's comp of -13 → 0 0 0 0 1 1 0 1 } add

Z = X − Y = +19 0 0 0 1 0 0 1 1

+6 − (−13) ↑ sign bit

ECE2560 Lesson01 Page_15



ECE2560 Lesson01 Page_16

keep this "special case" in mind when converting the maximum negative number

8 bits can store a number if $-128 \leq \text{number} \leq +127$

otherwise there will be overflow

Range of 2's Comp

$$-2^{n-1} \leq N \leq 2^{n-1} - 1$$

Addition overflow in 2's compliment representation

8 bits 2's compliment representation
-128 <= N <= 127

Discard

addition

```
 10000001 = -127
 10000001 = -127
-------------
100000010  = 2  overflow!!
```

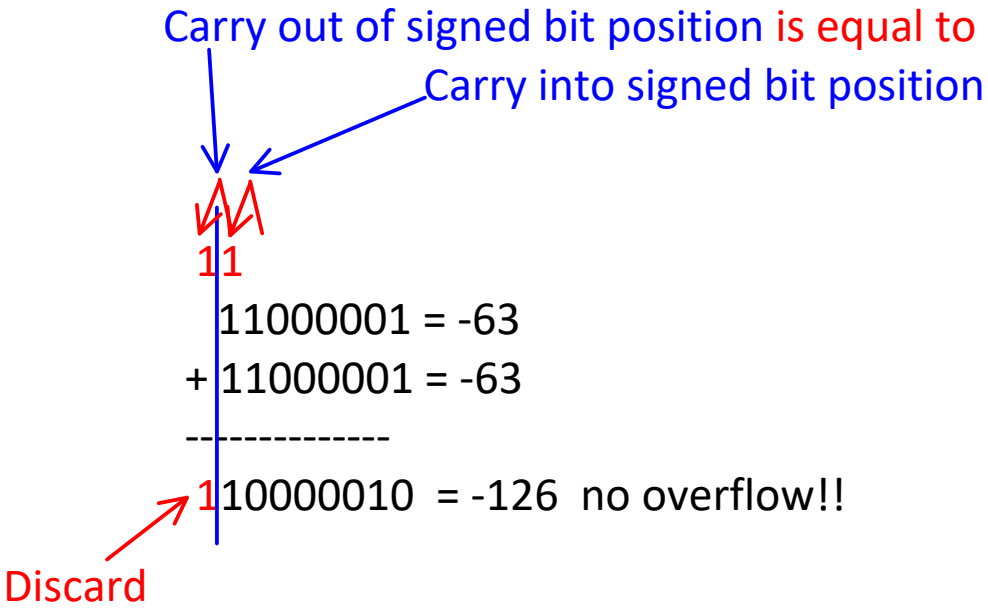
(2's compliment of 10000001 is 01111111 = 127
therefore 10000001 = -127)

```
  11000001    = -63
+ 11000001    =  -63          -63+(-63) = -126
-------------
 110000010 =  -126 no overflow!!
```

(2's compliment of 10000010 is 01111110 = 126
 therefore 10000010 = -126)

Carry out of signed bit position not equal to
Carry into signed bit position

```
 10
  10000001 = -127
+ 10000001 = -127
-------------
 100000010  = 2  overflow!!
```



For addition (in 2's compliment representation),

- Overflow can only happen if both numbers have the same sign
- There is overflow if carry out of the signed bit position is not equal to carry into the signed bit position