MSP430 devices have up to eight digital I/O ports implemented, P1 to P8. Number of ports varies with device type and packaging. For MSP430G2553IN20, there are two ports, in total 16 I/O pins available.
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
Number Systems

Binary numbers
base, radix = 2

\[ A_{n-1} A_{n-2} A_0 A_{-1} A_{-2} A_{-m} \]

\( A_i \in \{0, 1\} \quad A_i \rightarrow \text{binary digit (bit)} \)

Binary number

\[ 1011011101 = (1011011101)_2 \]

Converting from binary to decimal

\[ (1101011) = 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \]

\[ = 131_{10} \]

Notation: Code Composer Studio

\[ (110101)_2 \rightarrow 110101b \]

\( \text{msd (most significant digit)} \)

\( \text{lsd (least significant digit)} \)
decimal integer to binary

\((41)_{10} = (?,?)_2\)

\[
\begin{align*}
\frac{41}{2} &= 20 + \frac{1}{2} \\
\frac{20}{2} &= 10 + \frac{0}{2} \\
\frac{10}{2} &= 5 + \frac{0}{2} \\
\frac{5}{2} &= 2 + \frac{1}{2} \\
\frac{2}{2} &= 1 + \frac{0}{2} \\
\frac{1}{2} &= 0 + \frac{1}{2}
\end{align*}
\]

\((101001)_2\)
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)

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

16 hexadecimal digits

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

In Code Composer Studio:

\[(B65F)_{16} \rightarrow 0\text{XB65F} \text{ (C style)}\]

\[\rightarrow \text{B65F} \text{ (TI style)}\]
Important table for converting bases among binary, octal and hexadecimal number representation

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Binary</th>
<th>Octal</th>
<th>Hex</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0000</td>
<td>00</td>
<td>0</td>
</tr>
<tr>
<td>01</td>
<td>0001</td>
<td>01</td>
<td>1</td>
</tr>
<tr>
<td>02</td>
<td>0010</td>
<td>02</td>
<td>2</td>
</tr>
<tr>
<td>03</td>
<td>0011</td>
<td>03</td>
<td>3</td>
</tr>
<tr>
<td>04</td>
<td>0100</td>
<td>04</td>
<td>4</td>
</tr>
<tr>
<td>05</td>
<td>0101</td>
<td>05</td>
<td>5</td>
</tr>
<tr>
<td>06</td>
<td>0110</td>
<td>06</td>
<td>6</td>
</tr>
<tr>
<td>07</td>
<td>0111</td>
<td>07</td>
<td>7</td>
</tr>
<tr>
<td>08</td>
<td>1000</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>09</td>
<td>1001</td>
<td>11</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>1010</td>
<td>12</td>
<td>A</td>
</tr>
<tr>
<td>11</td>
<td>1011</td>
<td>13</td>
<td>B</td>
</tr>
<tr>
<td>12</td>
<td>1100</td>
<td>14</td>
<td>C</td>
</tr>
<tr>
<td>13</td>
<td>1101</td>
<td>15</td>
<td>D</td>
</tr>
<tr>
<td>14</td>
<td>1110</td>
<td>16</td>
<td>E</td>
</tr>
<tr>
<td>15</td>
<td>1111</td>
<td>17</td>
<td>F</td>
</tr>
</tbody>
</table>
Converting from binary to hexadecimal

\[(10110001101011)_2 = (?)_{16}\]

(use Table)

\[
\begin{array}{cccc}
0010 & 1100 & 0110 & 1011 \\
\hline \\
2 & C & 6 & B
\end{array}
\]

pad zeros if required

\[= (2C6B)_{16}\]

Converting from hexadecimal to binary

\[(3A6)_{16} = (?)_2\]

(use Table)

\[
\begin{array}{cccccc}
0011 & 1010 & 0110 \\
\hline \\
3 & A & 6
\end{array}
\]

discard

\[= (1110100110)_2\]
Complements of Binary Numbers.

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

$n$ bit register

$00 \ldots 01101$

$n$

8 bit register

$000001101$

$1101$ stored in an 8 bit register

$N = 1101$

$n = 8$ (word length)

1's Complement of Binary Numbers

1's complement of $N$ \( \Rightarrow (2^n - 1) - N \)

Change all the 1's to 0, and all the 0's to 1 in $N$ to get 1's complement of $N$
\[ \begin{align*}
10110010 & \quad \text{1's comp.} \quad 01001101 \\
111 & \quad \text{1's comp.} \quad 11111000
\end{align*} \]

2's complement of \( N \) is given by:

\[ 2^{-N} \quad \text{for} \quad N \neq 0 \]
\[ 0 \quad \text{for} \quad N = 0 \]

2's comp. of \( N \) = 1's comp. of \( N + 1 \)

\[ \begin{align*}
101100 & \quad \text{2's comp.} \\
010011 & \quad \text{1's comp.} \\
+ 1 & \\
010100 & \quad \text{2's comp.}
\end{align*} \]

"Trick" to find 2's complement 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 \[ \begin{align*}101100 & \quad \text{2's comp.} \\
010100 & \quad \text{010100}
\end{align*} \]

Invert these

Leave these Unchanged
<table>
<thead>
<tr>
<th>$2^n$</th>
<th>1 0 0 0 ... 0 0 0 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^{n-1}$</td>
<td>1 1 1 ... 1 1 1</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>$2^{n-1}$</td>
<td>1 0 0 ... 0 0 0</td>
</tr>
</tbody>
</table>
Unsigned Numbers

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

Number $N = + (10011010)_2$

Signed Numbers

(2's Complement Representation)
Example 8 bit registers

\[ N = (9)_{10} \rightarrow 00001001 \]

- Sign bit: 0 = +
- Magnitude: \( N = 001001 = (9)_{10} \)

\[ N = (-9)_{10} \rightarrow \text{2's comp of } 00001001 = 11110111 \]
- Sign bit: 1 = -

Example

\[ 11010100 \]

- Signed bit is 1 \( \therefore \) number is -ve
- 2's comp of \( 11010100 \) is \( 00101100 = (22)_{10} \)
  \( \therefore 11010100 \) is \( (-22)_{10} \)

\[ 00011011 \]

- Signed bit 0 \( \therefore \) number is +ve
  \( 00011011 \) is \( (27)_{10} \)
Signed Binary Addition and Subtraction

Addition

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

\[
\begin{align*}
X &= +6 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
Y &= +13 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\
Z &= X + Y &= +19 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & \text{ (sign bit)} \\
\end{align*}
\]

\[
\begin{align*}
X &= -6 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\
Y &= +13 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\
Z &= X + Y &= +7 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & \text{ (sign bit)} & \text{ (discard sign bit)} \\
\end{align*}
\]

\[
\begin{align*}
X &= +6 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
Y &= -13 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\
Z &= X + Y &= -7 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & \text{ (sign bit)} \\
\end{align*}
\]

2's comp of 1111001 is \[\text{0000111} = +7\]

Therefore, 1111001 is \(-7\).
\[ X = -6 \quad \overline{11111010} \]
\[ Y = -13 \quad \overline{11110011} \]
\[ Z = X + Y = -19 \quad \overline{11101101} \]

Discard the carry out

**2's comp of** \(\overline{11101101}\) **is** \(\overline{00010011} = +19\)

**Therefore** \(\overline{11101101}\) **is** -19

**Subtraction (2's comp representation)**

\[ Z = X - Y \]

\[ Z = \begin{cases} 
X + (2's \text{ comp of } Y) \\
\text{discard the carry out}
\end{cases} \]

\[ X = -6 \quad \overline{11111010} \]
\[ Y = -13 \quad \overline{11110011} \]

Add

\[ Z = X - Y = +7 \quad \overline{00000111} \]

Discard the carry out

\[ -6 - (-13) \quad \text{Sign bit} \]

\[ X = +6 \quad \overline{00000110} \]
\[ Y = -13 \quad \overline{00001101} \]

Add

\[ Z = X - Y = +19 \quad \overline{00010011} \]

Sign bit
overflow in 2’s complement representation

Assume 8 bits.

<table>
<thead>
<tr>
<th>Sign bit</th>
<th>Value</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 1 1 1 1 1</td>
<td>+127</td>
<td>maximum positive number</td>
</tr>
<tr>
<td>1 1 1 1 1 1 1</td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>1 0 0 0 0 0 0</td>
<td>-127</td>
<td></td>
</tr>
<tr>
<td>1 0 0 0 0 0 0 0</td>
<td>-128</td>
<td>maximum negative number</td>
</tr>
</tbody>
</table>

Note: 2’s complement of 100000000 should give +128

but +128 needs more than 8 bits!

\[
\begin{align*}
\underbrace{0 1 0 0 0 0 0 0}_9 + 128
\end{align*}
\]
Keep two "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

\[
\begin{align*}
\text{addition} \\
10000001 &= -127 \\
10000001 &= -127 \\
\hline
100000010 &= 2 \text{ overflow!!}
\end{align*}
\]

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

\[
\begin{align*}
11000001 &= -63 \\
+11000001 &= -63 \\
\hline
110000010 &= -126 \text{ no overflow!!}
\end{align*}
\]

(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

\[
\begin{align*}
10 \\
10000001 &= -127 \\
+10000001 &= -127 \\
\hline
100000010 &= 2 \text{ overflow!!}
\end{align*}
\]
Carry out of signed bit position is equal to
Carry into signed bit position

11000001 = -63
+ 11000001 = -63
-----------
11000010 = -126 no 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
overflow if

carry into sign bit position ≠ carry out of sign bit position

\[
\begin{array}{c}
\text{C}_{n-1} \\
\rightarrow X \\
\rightarrow Y \\
\rightarrow F
\end{array}
\]

\[
F = X \oplus Y
\]

\[
\begin{array}{c|c|c}
X & Y & F \\
\hline
0 & 0 & 0 \\
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 0
\end{array}
\]

\[
\text{overflow}
\]

10000000 \rightarrow -128

2's complement = 1's complement + 1

\[
\begin{array}{c}
0 \rightarrow \text{carry into signed bit position} . \\
\hline
1 \rightarrow 1\text{'s complement} + 1 \\
\hline
10000000
\end{array}
\]

overlow

V = 1 - overflow
V = 0 - no overflow