# Tutorial 3: ADC and Lab 3 Preparation

TNE097 Microcomputer Systems

Naveen Venkategowda

2025-11-24

LINKÖPING UNIVERSITY

# MSP 430 Code Examples

# MSP430 ADC Operation

- ADC on/off: ADC10ON bit in ADC10CTL0 control register 0
- ENC=0 if we want to modify ADC10CTL registers
- Set ENC=1 to enable conversion
- ADC conversion initiation: rising edge of SHI
  - SHI source: ADC10SC bit or timer output units
- Write into ADC10SA to enable DTC transfer
  - ADC10SA= Start address of destination
- ENC enable and trigger for initiation must be repeated for each conversion

# ADC Control Register 0: ADC10CTL0

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| SREFx | | | ADC10SHTx | | ADC10SR | REFOUT | REFBURST |
| rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MSC | REF2_5V | REFON | ADC10ON | ADC10IE | ADC10IFG | ENC | ADC10SC |
| rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) |

Can be modified only when ENC = 0

| MSC | Bit 7 | Multiple sample and conversion. Valid only for sequence or repeated modes. |
|---|---|---|
| | 0 | The sampling requires a rising edge of the SHI signal to trigger each sample-and-conversion. |
| | 1 | The first rising edge of the SHI signal triggers the sampling timer, but further sample-and-conversions are performed automatically as soon as the prior conversion is completed |

# MSP430 ADC Trigger Options

| Trigger Source | CONSEQ = 00/01<br>MSC = x | CONSEQ = 10/11<br>MSC = 0 | CONSEQ = 10/11<br>MSC = 1 |
|---|---|---|---|
| Hardware | Single conversions, individually triggered | Repeated conversions, individually triggered | Multiple conversions, only first triggered |
| | ENC: toggle ENC every time | ENC: toggle ENC only before start | |
| Software | Repeated conversions, individually triggered | | Multiple conversions, only first triggered |
| | toggle ENC only before start or with ADC10SC in all cases | | |

# ADC10 Interrupts

- One interrupt and one interrupt vector are associated with ADC10
- When DTC is not used (ADC10DTC1 = 0)
  - ADC10IFG is set when conversion results are saved to ADC10MEM.
- When DTC is used (ADC10DTC1 > 0)
  - ADC10IFG is set when a block transfer completes and transfer counter n = 0.
- If both ADC10IE and GIE bits are set, then ADC10IFG flag generates an interrupt request.
  - ADC10IFG flag is automatically reset when interrupt request is serviced,
  - Flag may be reset by software

# MSP 430 Code Flow for ADC

1. Stop watchdog timer
2. Configure ADC control register 0 (ADC10CTL0)
   - ADC on/off
   - Reference voltage sources
   - Sample-hold time
3. Configure ADC control register 1 (ADC10CTL1)
   - Input channels
   - Conversion modes
   - Sample-hold trigger source
   - Data format
   - Clock sources
4. Configure I/O ports for ADC inputs in register ADC10AE0

# MSP 430 Code Flow for ADC

5. To check if ADC is busy read bit 0 (ADCBUSY) in control register ADC10CTL1
   - `ADC10CTL1 & ADC10BUSY`
6. To start ADC conversion set enable and start conversion bits in control register ADC10CTL0
   - ADC10CTL0 |= ENC + ADC10SC
7. Read ADC digital value from ADC10MEM register

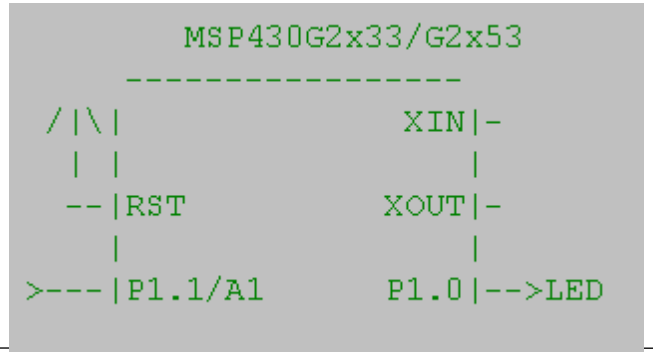
- If using interrupt service routine
  - ISR vector = ADC10_VECTOR
  - Depending on conversion modes: modify ENC, ADC10SC bits
- If using DTC
  - Set number of transfers per block in ADC10DTC1 register
  - Set memory location for samples in ADC10SA address register

# MSP 430 Code Flow for ADC

- If using interrupt service routine
  - ISR vector = ADC10_VECTOR
  - Depending on conversion modes: modify ENC, ADC10SC bits
- If using DTC
  - Set number of transfers per block in ADC10DTC1 register
  - Set memory location for samples in ADC10SA address register

# Example 1: Single-Channel Single-Conversion Mode, ADC clock: Internal ADC oscillator, 5 MHz

```
void main(void)
    {  WDTCTL = WDTPW + WDTHOLD;                    // Stop WDT
      ADC10CTL0 = ADC10SHT_2 + ADC10ON;     // Sample and hold time=16 ADC clocks, ADC ON
      ADC10CTL1 = INCH_1;                           // input A1
      ADC10AE0 |= 0x02;                             // P1.1 ADC function selected
      P1DIR |= 0x01;                                // Set P1.0 to output direction
      ADC10CTL0 |= ENC + ADC10SC;                // Enable and start conversion
      while(ADC10CTL1 & ADC10BUSY) {} //Wait until the conversion is ready (Testing ADC10BUSY
bit in ADC10CTL1)
      if (ADC10MEM < 0x1FF)                 //
    {P1OUT &= ~0x01;} // Clear P1.0 LED off
      else
        {P1OUT |= 0x01;} // Set P1.0 LED on
    }
```



# Example 2: Single ADC triggered in an infinite for loop, ADC clock: SMCLK 1 MHz

```
void main(void)
    { WDTCTL = WDTPW + WDTHOLD;                    // Stop WDT
      ADC10CTL0 = ADC10SHT_2 + ADC10ON;      // Sample and hold time = 16 ADC clocks, ADC ON
      ADC10CTL1 = INCH_1 + ADC10SSEL_3 ;          // input A1, SMCLK 1 MHz
      ADC10AE0 |= 0x02;                           // P1.1 ADC function selected
      P1DIR |= 0x01;                              // Set P1.0 to output direction
for(;;){
      ADC10CTL0 |= ENC + ADC10SC;                // Enable and start conversion
      while(ADC10CTL1 & ADC10BUSY) {} //wait until the conversion is ready (Testing ADC10BUSY
bit in ADC10CTL1)
      if (ADC10MEM < 0x1FF)                  //Voltage = 3.3V * 0x1FF / 0x3FF
        {P1OUT &= ~0x01;}                         // Clear P1.0 LED off
      else
        {P1OUT |= 0x01;}                          // Set P1.0 LED on
    }
}
```

# Example 3: Single ADC triggered in an infinite for loop, Temperature sensor

```
void main(void)
    {  WDTCTL = WDTPW + WDTHOLD;                    // Stop WDT
      ADC10CTL0 = ADC10SHT_3 + ADC10ON; // Sample and hold time = 64 ADC clocks, ADC ON
      ADC10CTL1 = INCH_10 + ADC10SSEL_3 ;          // Temperature sensor, SMCLK 1 MHz
      P1DIR |= 0x01;                              // Set P1.0 to output direction
for(;;){
      ADC10CTL0 |= ENC + ADC10SC;                // Enable and start conversion
      while(ADC10CTL1 & ADC10BUSY) {} //wait until the conversion is ready (Testing
ADC10BUSY bit in ADC10CTL1)
      if (ADC10MEM < 0x1FF)  //Select a proper value to test the temperature sensor
    {P1OUT &= ~0x01;}                            // Clear P1.0 LED off
      else
        {P1OUT |= 0x01;}                         // Set P1.0 LED on
    }
}
```

Temperature sensor requires sample-and-hold time > 30µs.

# Example 4: Single-Channel Single-Conversion Mode with Interrupt

```
void main(void)
{  WDTCTL = WDTPW + WDTHOLD;                    // Stop WDT
  ADC10CTL0 = ADC10SHT_2 + ADC10ON + ADC10IE; // ADC10ON, interrupt enabled
  ADC10CTL1 = INCH_1 + ADC10SSEL_3 ;           // input A1, SMCLK
  ADC10AE0 |= 0x02;                           // P1.1 ADC function selected
  P1DIR |= 0x01;                              // Set P1.0 to output direction
  ADC10CTL0 |= ENC + ADC10SC;                // Enable and start conversion
  __bis_SR_register(LPM0_bits + GIE);       // LPM0, general interrupt enable
}
#pragma vector=ADC10_VECTOR
__interrupt void ADC10_ISR(void)
{
    if (ADC10MEM < 0x1FF)                   //Voltage = 3.3V * 0x1FF / 0x3FF
      {P1OUT &= ~0x01; }                       // Clear P1.0 LED off
    else
    {  P1OUT |= 0x01;  }           // Set P1.0 LED on
    ADC10CTL0 |= ENC + ADC10SC;     // Enable and start conversion
}
```

# Example 5, Repeat single channel, Multiple sample and conversion with interrupt

```
void main(void)
    { WDTCTL = WDTPW + WDTHOLD;                    // Stop WDT
      // Sample and hold time = 16 ADC clocks, ADC10ON, Enable interrupt, Multiple sample and
conversion
      ADC10CTL0 = ADC10SHT_2 + ADC10ON + ADC10IE + MSC;
      ADC10CTL1 = CONSEQ_2 + INCH_1 + ADC10SSEL_3;  // Repeat single channel, input A1, ADC clock:
SMCLK
      ADC10AE0 |= 0x02;                            // P1.1 ADC function selected
      P1DIR |= 0x01;                               // Set P1.0 to output direction
      ADC10CTL0 |= ENC + ADC10SC;                 // Sampling and conversion start
      __bis_SR_register(LPM0_bits + GIE); //DO NOT forget the global interrupt enable
}
#pragma vector=ADC10_VECTOR
__interrupt void ADC10_ISR(void)
{
    if (ADC10MEM < 0x1FF)
      {P1OUT &= ~0x01; }                     // Clear P1.0 LED off
    else
    {  P1OUT |= 0x01;  }                    // Set P1.0 LED on
}
```

# Sample code – DTC (one block)

```
//create a result buffer
/*the number of element is same as number
of ADC reading using DTC*/

int a[0x10] = {0};

/*enable the DTC by setting the ADC10DTC1
with a number other than 0*/
ADC10DTC1 = 0x10;

/*Set the address of the ADC buffer with
the address of the Array */

ADC10SA = (int)a;
// or
ADC10SA = (unsigned short)a;
```

Watch 1

| Expression | Value | Location | Type |
|---|---|---|---|
| av | Error (c... | | |
| a | <array> | Memory:0x200 | int[16] |
| [0] | 1022 | Memory:0x200 | int |
| [1] | 1022 | Memory:0x202 | int |
| [2] | 1022 | Memory:0x204 | int |
| [3] | 1022 | Memory:0x206 | int |
| [4] | 1022 | Memory:0x208 | int |
| [5] | 1022 | Memory:0x20A | int |
| [6] | 1021 | Memory:0x20C | int |
| [7] | 1023 | Memory:0x20E | int |
| [8] | 1022 | Memory:0x210 | int |
| [9] | 1022 | Memory:0x212 | int |
| [10] | 1017 | Memory:0x214 | int |
| [11] | 1022 | Memory:0x216 | int |
| [12] | 1022 | Memory:0x218 | int |
| [13] | 1022 | Memory:0x21A | int |
| [14] | 1022 | Memory:0x21C | int |
| [15] | 1023 | Memory:0x21E | int |
| ADC10M... | 1022 | Memory:0x1B4 | unsigned short |
| <click to ... | | | |

# Lab 3: ADC

# MSP 430 ADC Registers

# ADC Registers summary

**Table 22-3. ADC10 Registers**

| Register | Short Form | Register Type | Address | Initial State |
|---|---|---|---|---|
| ADC10 input enable register 0 | ADC10AE0 | Read/write | 04Ah | Reset with POR |
| ADC10 input enable register 1 | ADC10AE1 | Read/write | 04Bh | Reset with POR |
| ADC10 control register 0 | ADC10CTL0 | Read/write | 01B0h | Reset with POR |
| ADC10 control register 1 | ADC10CTL1 | Read/write | 01B2h | Reset with POR |
| ADC10 memory | ADC10MEM | Read | 01B4h | Unchanged |
| ADC10 data transfer control register 0 | ADC10DTC0 | Read/write | 048h | Reset with POR |
| ADC10 data transfer control register 1 | ADC10DTC1 | Read/write | 049h | Reset with POR |
| ADC10 data transfer start address | ADC10SA | Read/write | 01BCh | 0200h with POR |

# MSP ADC Programming

| Function | Parameter | Name | Register |
|---|---|---|---|
| I/O management | Select input channels | INCHx | ADC10CTL1 |
| | Input enable/ disable | ADC10AE0 | ADC10AE0 |
| Clock | Clock source | ADC10SSELx | ADC10CTL1 |
| | Clock divider | ADC10DIVx | ADC10CTL1 |
| Reference voltages | Reference source | SREFx | ADC10CTL0 |
| | Internal reference generator on/off | REFON | ADC10CTL0 |
| | Internal reference gen. voltage | REF2_5V | ADC10CTL0 |
| Start/ enable | ADC on/ off | ADC10ON | ADC10CTL0 |
| | Enable/disable conversion | ENC | ADC10CTL0 |
| | Start conversion | ADC10SC | ADC10CTL0 |
| Sampling | Sample-Hold select/trigger | SHSx | ADC10CTL1 |
| | Sample-Hold time | ADC10SHTx | ADC10CTL0 |
| | Sampling rate | ADC10SR | ADC10CTL0 |

# MSP ADC Programming

| Function | Parameter | Name | Register |
|---|---|---|---|
| Status | Busy | ADC10BUSY | ADC10CTL1 |
| Interrupts | ADC interrupt enable | ADC10IE | ADC10CTL0 |
| | ADC interrupt flag | ADC10IFG | ADC10CTL0 |
| Data acquisition | Conversion modes | CONSEQx | ADC10CTL1 |
| | Multiple sample and conversion | MSC | ADC10CTL0 |
| | Data format: Binary/ 2's compl. | ADC10DF | ADC10CTL1 |
| Memory management | One/two block mode | ADC10TB | ADC10DTC0 |
| | Continuous transfer: Single/multiple | ADC10CT | ADC10DTC0 |
| | Block order (Block1 or 2) | ADC10B1 | ADC10DTC0 |
| | DTC Number of transfers | | ADC10DTC1 |
| | DTC start address | | ADC10SA |

# Task 1 Code

```
    void main(void)
    {
1   WDTCTL = WDTPW + WDTHOLD; // Stop WDT
2   ADC10CTL0 = SREF_1 + ADC10SHT_2 + REFON + ADC10ON;
3   ADC10CTL1 = INCH_1; // input A1, pin P1.1
4   ADC10AE0 |= 0x02; // Select ADC function for pin P1.1
5   P1DIR |= 0x01; // Set P1.0 to output direction
6   for (;;)
    {
8   ADC10CTL0 |= ENC + ADC10SC; // Sampling and conversion start
9   while(ADC10CTL1 & ADC10BUSY) {} //wait until the conversion is ready
    // Testing ADC10BUSY bit in ADC10CTL1
10  if (ADC10MEM < 0x88) // ADC10MEM = A1 > V?
11  P1OUT &= ~0x01; // Clear P1.0 LED off
12  else
14  P1OUT |= 0x01; // Set P1.0 LED on
    }
```



Code shows that MSP430 uses internal 1.5V reference voltage to convert voltage from channel A1.

# Task 1.1

- Modify the code so that MSP430:
  - measures analog voltage from channel A5
  - uses reference voltage as internal reference 2.5V
- Channel Select: Configure ADC10CTL1 register to select channel A5 (INCH_5)
- Reference voltage: Configure ADC10CTL0 to change ref voltage
  - Bit 6: REF2_5V

# ADC Control Registers: ADC10CTL0

## ADC10CTL0

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| SREFx | | | ADC10SHTx | | ADC10SR | REFOUT | REFBURST |
| rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MSC | REF2_5V | REFON | ADC10ON | ADC10IE | ADC10IFG | ENC | ADC10SC |
| rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) |

Can be modified only when ENC = 0

## ADC10CTL1

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| INCHx | | | | SHSx | | ADC10DF | ISSH |
| rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ADC10DIVx | | | ADC10SSELx | | CONSEQx | | ADC10BUSY |
| rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | r-0 |

Can be modified only when ENC = 0

# Task 1.2

- Modify the code so that MSP430:
  - Uses interrupt to read the measured voltage.
  - If the measured voltage is higher than 1.5V turn on RED LED by setting P1.0 (to 1). Otherwise, reset P1.0 (to 0).

# Task 1.2

- Follow preliminary ADC configuration as in Task 1.1
- Enable ADC10 interrupt
  - Configure ADC10CTL0 register
  - Use ADC10IE interrupt enable bit
- Enable general interrupt
  - GIE in status register using intrinsic function
- Write ISR
  - Get ADC10 interrupt vector with pragma directive (ADC10_VECTOR)
  - Data processing
  - Restart sampling and conversion

# Task 2: Use Timer 0 to Trigger ADC

- Following code use Timer0_A TACCR0 and TACCR1 to generate a pulse (PWM)
- This pulse can be utilized as the signal to start sampling and conversion for ADC10 of MSP430

```
//============================================
TACCR0 = 2048-1; // PWM Period
TACCTL1 = OUTMOD_3; // Output Mode 3: set/reset
TACCR1 = 1024-1; // TACCR1 PWM Duty Cycle
TACTL = TASSEL_2 + MC_1; // SMCLK, up mode
//============================================
```

# Task 2: Use Timer 0 to Trigger ADC

- ADC sampling signal is controlled by the Timer0_A CCR1 output unit instead of ADC10SC
- Configure the ADC working mode to “Repeat single channel mode” from A1 so that ADC10 can be continuously triggered by Timer0_A output
- Read the ADC10 value from ADC interrupts.
- When input voltage from A1 is higher than 1.5V, set P1.0, otherwise, reset P1.0.

# Task 2.1: Code Flow to Use ADC with Timer

- Modify the algorithm based on step below

1. Select sample and hold trigger source in ADC10CTL1
   - SHSx = 01 (for Timer 1)
2. Configure ADC in “Repeat Single Channel” conversion mode for input A1
   - INCH_3 using ADC10AE0
   - Mode using CONSEQ_1 in ADC10CTL1
   - P1SEL for general purpose I/O configuration
3. Enable ADC
   - Only ENC bit.
   - Not using ADC10SC bit but using Timer 1 instead
4. Configure Timer 1 for desired output to trigger ADC

# Task 2.1: Code Flow to Use ADC with Timer

5. Create ISR for ADC
   - Get ISR vector #pragma vector=ADC10_VECTOR
   - Read ADC output value from ADC10MEM register
   - Compare ADC output with given value

if( A1 >1.5V) LED On
else LED off

Conversion from given voltage to binary value for comparison:

$$N_{ADC} = 1023 \frac{V_{in} - V_{R-}}{V_{R+} - V_{R-}}$$

   - No need to restart ADC as timer is used for start conversion

# Task 3: Using DTC with ADC

- Implement a function that uses DTC mode to get multiple ADC readings from a single channel
- Key Steps:
  - Configure ADC10 as “repeat single channel mode” via channel A1
  - Use internal reference 2.5V
  - Enable MSC bit to set ADC work continuously.
  - Configure the ADC DTC mode as “one block” mode, 16 readings (n = 0x10)
  - Assigned the read value to an array with 16 elements
  - Handle the ADC DTC reading in an ISR and calculate the average value of the ADC reading
  - Restart the DTC in the ADC interrupt.

# Task 3: Code Flow Using DTC with ADC

- Follow preliminary ADC configuration as in previous tasks and modify the algorithm based on steps below

1. Configure ADC
   - "Repeat single channel" conversion mode
   - Input Channel A1
   - VR+ = 2.5 V
2. Configure ADC to take multiple readings
   - Use MSC bit instead of Timer or ADC10SC bit
3. Enable interrupts

# Task 3: Code Flow Using DTC with ADC

4. Configure DTC operation
   - Enable DTC
   - Set number of transfers per block
   - Set DTC sample location to where the physical memory array is located
5. Create ISR
   - Compute average
   - Restart DTC again: set number of transfer and set DTC sample location ADC10SA = (int)samples

# MSP 430 ADC: ADC Control Register 0: ADC10CTL0

# ADC Control Register 0: ADC10CTL0

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| SREFx | | | ADC10SHTx | | ADC10SR | REFOUT | REFBURST |
| rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MSC | REF2_5V | REFON | ADC10ON | ADC10IE | ADC10IFG | ENC | ADC10SC |
| rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) |

Can be modified only when ENC = 0

| SREFx | Bits 15-13 | Select reference |
|---|---|---|
| | 000 | VR+ = VCC and VR- = VSS |
| | 001 | VR+ = VREF+ and VR- = VSS |
| | 010 | VR+ = VeREF+ and VR- = VSS. |
| | 011 | VR+ = Buffered VeREF+ and VR- = VSS. |

# ADC Control Register 0: ADC10CTL0

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| SREFx | | | ADC10SHTx | | ADC10SR | REFOUT | REFBURST |
| rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MSC | REF2_5V | REFON | ADC10ON | ADC10IE | ADC10IFG | ENC | ADC10SC |
| rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) |

Can be modified only when ENC = 0

| SREFx | Bits 15-13 | Select reference |
|---|---|---|
| | 100 | VR+ = VCC and VR- = VREF-/ VeREF- |
| | 101 | VR+ = VREF+ and VR- = VREF-/ VeREF- |
| | 110 | VR+ = VeREF+ and VR- = VREF-/ VeREF- |
| | 111 | VR+ = Buffered VeREF+ and VR- = VREF-/ VeREF- |

# ADC Control Register 0: ADC10CTL0

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| SREFx | | | ADC10SHTx | | ADC10SR | REFOUT | REFBURST |
| rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MSC | REF2_5V | REFON | ADC10ON | ADC10IE | ADC10IFG | ENC | ADC10SC |
| rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) |

Can be modified only when ENC = 0

| ADC10SHTx | Bits 12-11 | ADC10 sample-and-hold time |
|---|---|---|
| | 00 | 4 × ADC10CLKs |
| | 01 | 8 × ADC10CLKs |
| | 10 | 16 × ADC10CLKs |
| | 11 | 64 × ADC10CLKs |

# ADC Control Register 0: ADC10CTL0

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| SREFx | | | ADC10SHTx | | ADC10SR | REFOUT | REFBURST |
| rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MSC | REF2_5V | REFON | ADC10ON | ADC10IE | ADC10IFG | ENC | ADC10SC |
| rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) |

Can be modified only when ENC = 0

| ADC10SRs | Bits 10 | ADC10 sampling rate |
|---|---|---|
| | 0 | Reference buffer supports up to ~200 ksps |
| | 1 | Reference buffer supports up to ~50 ksps |
| **REFOUT** | **Bit 9** | **Reference output** |
| | 0 | Reference output off |
| | 1 | Reference output on |

# ADC Control Register 0: ADC10CTL0

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| SREFx | | | ADC10SHTx | | ADC10SR | REFOUT | REFBURST |
| rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MSC | REF2_5V | REFON | ADC10ON | ADC10IE | ADC10IFG | ENC | ADC10SC |
| rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) |

Can be modified only when ENC = 0

| REFBURST | Bits 8 | Reference burst |
|---|---|---|
| | 0 | Reference buffer on continuously |
| | 1 | Reference buffer on only during sample-and-conversion |

# ADC Control Register 0: ADC10CTL0

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| SREFx | | | ADC10SHTx | | ADC10SR | REFOUT | REFBURST |
| rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MSC | REF2_5V | REFON | ADC10ON | ADC10IE | ADC10IFG | ENC | ADC10SC |
| rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) |

Can be modified only when ENC = 0

| MSC | Bit 7 | Multiple sample and conversion. Valid only for sequence or repeated modes. |
|---|---|---|
| | 0 | The sampling requires a rising edge of the SHI signal to trigger each sample-and-conversion. |
| | 1 | The first rising edge of the SHI signal triggers the sampling timer, but further sample-and-conversions are performed automatically as soon as the prior conversion is completed |

# ADC Control Register 0: ADC10CTL0

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| SREFx | | | ADC10SHTx | | ADC10SR | REFOUT | REFBURST |
| rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MSC | REF2_5V | REFON | ADC10ON | ADC10IE | ADC10IFG | ENC | ADC10SC |
| rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) |

Can be modified only when ENC = 0

| REF2_5V | Bits 6 | Reference-generator voltage. REFON must also be set. |
|---|---|---|
| | 0 | 1.5 V |
| | 1 | 2.5 V |
| **REFON** | **Bit 5** | **Reference generator on** |
| | 0 | Reference off |
| | 1 | Reference on |

# ADC Control Register 0: ADC10CTL0

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| SREFx | | | ADC10SHTx | | ADC10SR | REFOUT | REFBURST |
| rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MSC | REF2_5V | REFON | ADC10ON | ADC10IE | ADC10IFG | ENC | ADC10SC |
| rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) |

Can be modified only when ENC = 0

| ADC10ON | Bits 4 | ADC10 on |
|---|---|---|
| | 0 | ADC10 off |
| | 1 | ADC10 on |
| **ADC10IE** | **Bit 3** | **ADC10 interrupt enable** |
| | 0 | Interrupt disabled |
| | 1 | Interrupt enabled |

# ADC Control Register 0: ADC10CTL0

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| SREFx | | | ADC10SHTx | | ADC10SR | REFOUT | REFBURST |
| rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MSC | REF2_5V | REFON | ADC10ON | ADC10IE | ADC10IFG | ENC | ADC10SC |
| rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) |

Can be modified only when ENC = 0

| ADC10IFG | Bit 2 | ADC10 interrupt flag. |
|---|---|---|
| | 0 | No interrupt pending |
| | 1 | Interrupt pending |

- **ADC10IFG** bit is set if ADC10MEM is loaded with a conversion result.
- **ADC10IFG** is automatically reset when the interrupt request is accepted, or it may be reset by software. When using the DTC this flag is set when a block of transfers is completed.

# ADC Control Register 0: ADC10CTL0

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| SREFx | | | ADC10SHTx | | ADC10SR | REFOUT | REFBURST |
| rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MSC | REF2_5V | REFON | ADC10ON | ADC10IE | ADC10IFG | ENC | ADC10SC |
| rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) |

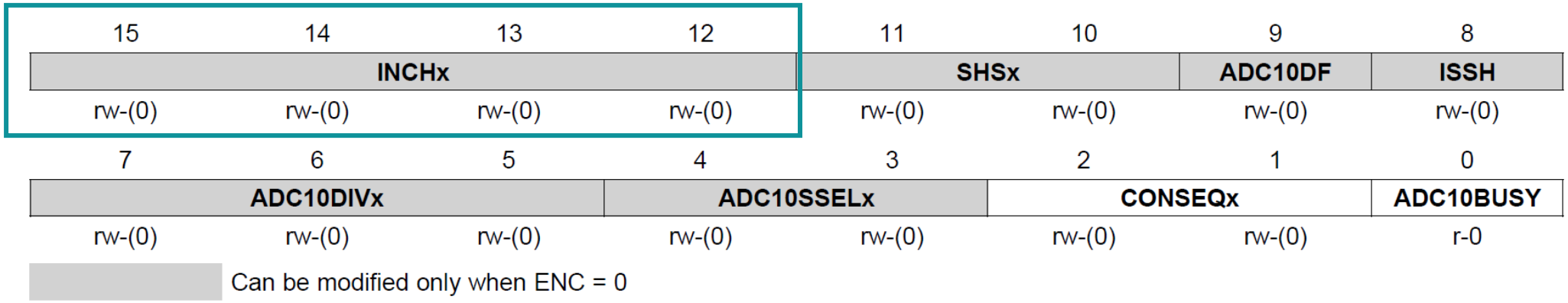
Can be modified only when ENC = 0

| ENC | Bit 1 | Enable conversion |
|---|---|---|
| | 0 | ADC10 disabled |
| | 1 | ADC10 enabled |
| **ADC10SC** | **Bit 0** | **Start conversion. Software-controlled sample-and-conversion start.** |
| | 0 | No sample-and-conversion start |
| | 1 | Start sample-and-conversion |

ADC10SC and ENC may be set together with one instruction. ADC10SC is reset automatically.

# MSP 430 ADC: ADC Control Register 1: ADC10CTL1

# ADC Control Register 1: ADC10CTL1

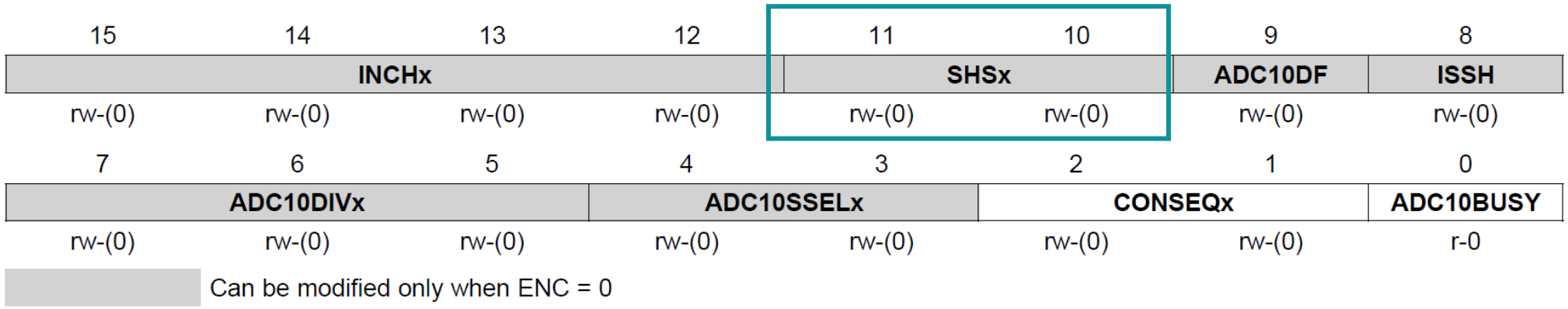



| INCHx | Bits 15-12 | Input channel select |
|---|---|---|
| | 0000 | A0 |
| | 0001 | A1 |
| | 0010 | A2 |
| | . | . |
| | 0111 | A7 |

| Bits 15-12 | Input channel select |
|---|---|
| 1000 | VeREF+ |
| 1001 | VREF-/VeREF- |
| 1010 | Temperature sensor |
| 1011 | (VCC - VSS) / 2 |

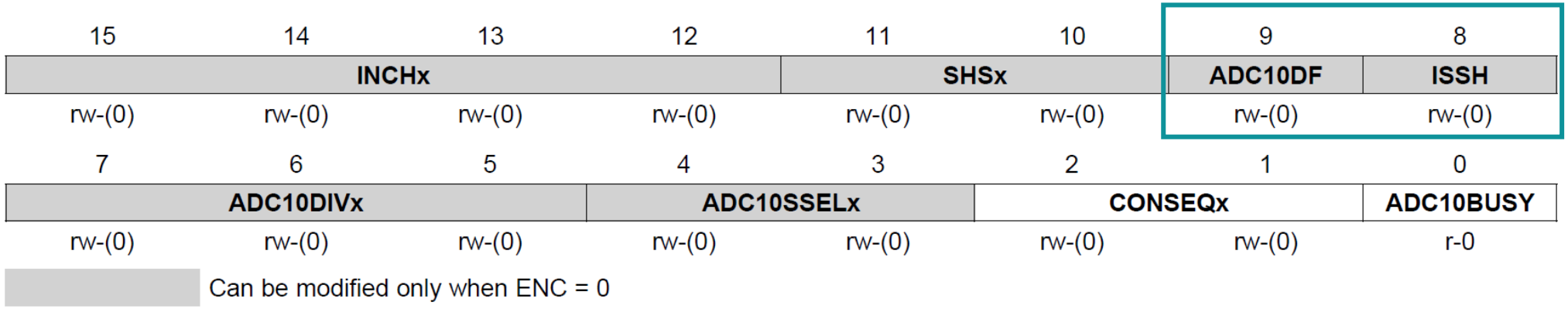
| Bits 15-12 | Input channel select |
|---|---|
| 1100 | (VCC - VSS) / 2, A12 on MSP430F22xx |
| 1101 | (VCC - VSS) / 2, A13 on MSP430F22xx |
| 1110 | (VCC - VSS) / 2, A14 on MSP430F22xx |
| 1111 | (VCC - VSS) / 2, A15 on MSP430F22xx |

# ADC Control Register 1: ADC10CTL1



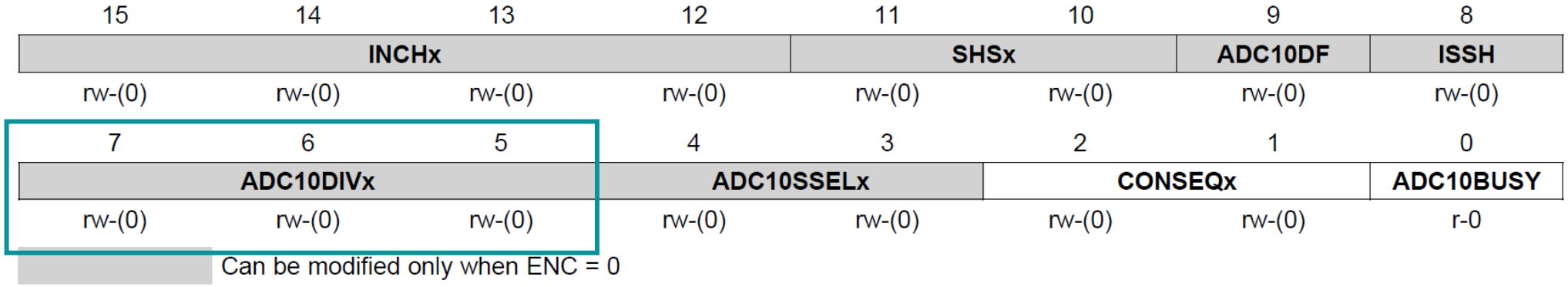

| SHSx | Bits 11-10 | Sample-and-hold source select |
|---|---|---|
| | 00 | ADC10SC bit |
| | 01 | Timer_A.OUT1 |
| | 10 | Timer_A.OUT0 |
| | 11 | Timer_A.OUT2 |

# ADC Control Register 1: ADC10CTL1




| ADC10DF | Bit 9 | ADC10 data format |
|---|---|---|
| | 0 | Straight binary |
| | 1 | 2s complement |
| **ISSH** | **Bit 8** | **Invert signal sample-and-hold** |
| | 0 | sample-input signal is not inverted |
| | 1 | sample-input signal is inverted |

# ADC Control Register 1: ADC10CTL1



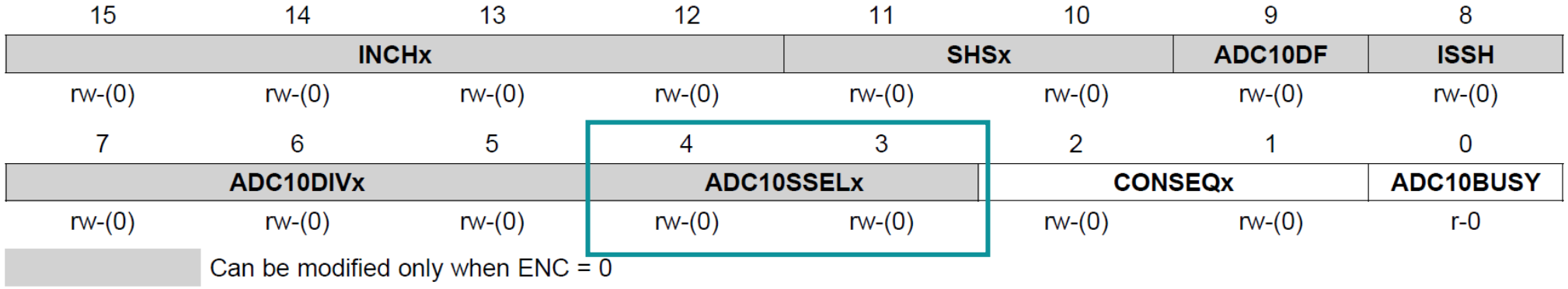
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| INCHx | | | | SHSx | | ADC10DF | ISSH |
| rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| ADC10DIVx | | | ADC10SSELx | | CONSEQx | | ADC10BUSY |
| rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | r-0 |

Can be modified only when ENC = 0

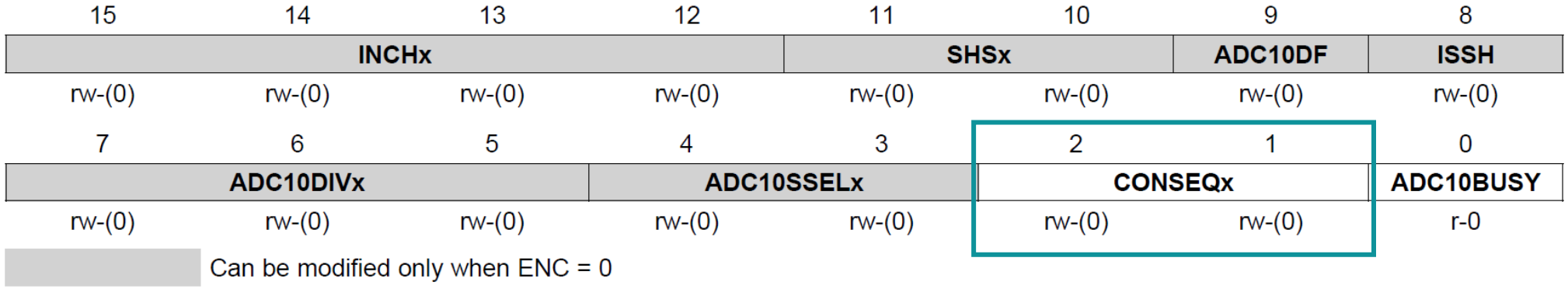
| ADC10DIVx | Bits 7-5 | ADC10 clock divider |
|---|---|---|
| | 000 | /1 |
| | 001 | /2 |
| | 010 | /3 |
| | . | . |
| | 111 | /8 |

# ADC Control Register 1: ADC10CTL1




| ADC10SSELx | Bits 4-3 | ADC10 clock source select |
|---|---|---|
| | 00 | ADC10OSC |
| | 01 | ACLK |
| | 10 | MCLK |
| | 11 | SMCLK |

# ADC Control Register 1: ADC10CTL1




| CONSEQx | Bits 2-1 | Conversion sequence mode select |
|---|---|---|
| | 00 | Single-channel-single-conversion |
| | 01 | Sequence-of-channels |
| | 10 | Repeat-single-channel |
| | 11 | Repeat-sequence-of-channels |

# ADC Control Register 1: ADC10CTL1

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| INCHx | | | | SHSx | | ADC10DF | ISSH |
| rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ADC10DIVx | | | ADC10SSELx | | CONSEQx | | ADC10BUSY |
| rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | r-0 |

Can be modified only when ENC = 0

| ADC10BUSY | Bit 0 | ADC10 busy. This bit indicates an active sample or conversion operation |
|---|---|---|
| | 0 | No operation is active |
| | 1 | A sequence, sample, or conversion is active. |

# MSP 430 ADC Memory Registers

# Analog (Input) Enable Control Register 0: ADC10AE0

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **ADC10AE0x** | | | | | | | |
| rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) |

| | | |
|---|---|---|
| **ADC10AE0x** | Bits 7-0 | ADC10 analog enable. These bits enable the corresponding pin for analog input. BIT0 corresponds to A0, BIT1 corresponds to A1, etc. The analog enable bit of not implemented channels should not be programmed to 1. |
| | | 0 Analog input disabled |
| | | 1 Analog input enabled |

# ADC Conversion-Memory Register: ADC10MEM

### 22.3.5 ADC10MEM, Conversion-Memory Register, Binary Format

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | **Conversion Results** | |
| r0 | r0 | r0 | r0 | r0 | r0 | r | r |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| **Conversion Results** | | | | | | | |
| r | r | r | r | r | r | r | r |

| | | |
|---|---|---|
| **Conversion Results** | Bits 15-0 | The 10-bit conversion results are right justified, straight-binary format. Bit 9 is the MSB. Bits 15-10 are always 0. |

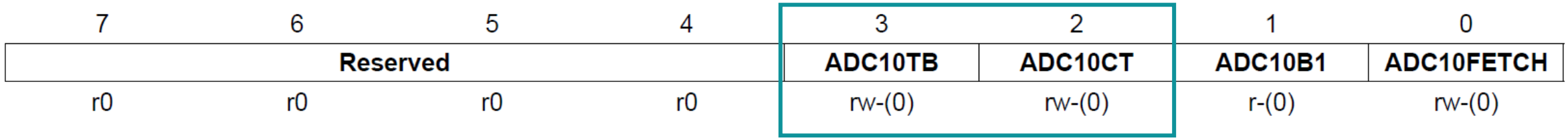
Change data format: ADC10DF in ADC10CTL1

### 22.3.6 ADC10MEM, Conversion-Memory Register, 2s Complement Format

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| **Conversion Results** | | | | | | | |
| r | r | r | r | r | r | r | r |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| **Conversion Results** | | 0 | 0 | 0 | 0 | 0 | 0 |
| r | r | r0 | r0 | r0 | r0 | r0 | r0 |

| | | |
|---|---|---|
| **Conversion Results** | Bits 15-0 | The 10-bit conversion results are left-justified, 2s complement format. Bit 15 is the MSB. Bits 5-0 are always 0. |

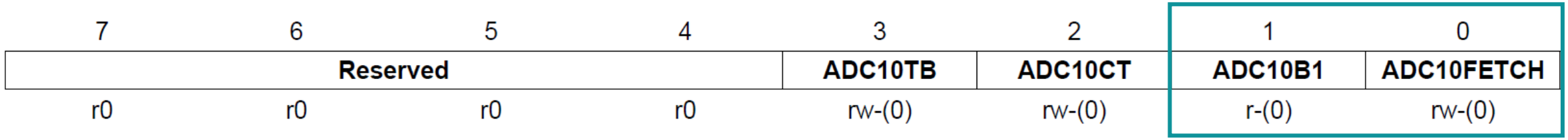
# Data Transfer Control Register 0: ADC10DTC0



| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | ADC10TB | ADC10CT | ADC10B1 | ADC10FETCH |
| r0 | r0 | r0 | r0 | rw-(0) | rw-(0) | r-(0) | rw-(0) |

| ADC10TB | Bit 3 | ADC10 two-block mode |
|---|---|---|
| | 0 | One-block transfer mode |
| | 1 | Two-block transfer mode |
| **ADC10CT** | **Bit 2** | **ADC10 continuous transfer** |
| | 0 | Data transfer stops when one block (one-block mode) or two blocks (two-block mode) have completed. |
| | 1 | Data is transferred continuously. DTC operation is stopped only if ADC10CT cleared, or ADC10SA is written to. |

# Data Transfer Control Register 0: ADC10DTC0



| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | ADC10TB | ADC10CT | ADC10B1 | ADC10FETCH |
| r0 | r0 | r0 | r0 | rw-(0) | rw-(0) | r-(0) | rw-(0) |

| ADC10B1 | Bit 1 | ADC10 block one. This bit indicates for two-block mode which block is filled with ADC10 conversion results. |
|---|---|---|
| | 0 | Block 2 is filled |
| | 1 | Block 1 is filled |

| ADC10FETCH | Bit 0 | This bit should normally be reset. |
|---|---|---|

# Data Transfer Registers

### 22.3.8 ADC10DTC1, Data Transfer Control Register 1

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| DTC Transfers | | | | | | | |
| rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) |

| | | |
|---|---|---|
| **DTC Transfers** | Bits 7-0 | DTC transfers. These bits define the number of transfers in each block.<br>0 DTC is disabled<br>01h-0FFh Number of transfers per block |

### 22.3.9 ADC10SA, Start Address Register for Data Transfer

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| ADC10SAx | | | | | | | |
| rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ADC10SAx | | | | | | | 0 |
| rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | r0 |

| | | |
|---|---|---|
| **ADC10SAx** | Bits 15-1 | ADC10 start address. These bits are the start address for the DTC. A write to register ADC10SA is required to initiate DTC transfers. |
| **Unused** | Bit 0 | Unused, Read only. Always read as 0. |