# UNIT IV

# ADVANCED MICROCONTROLLERS

## APPLICATION DEVELOPMENT ENVIRONMENT IN MSP430

The ADC featured on the MSP430 line of microcontrollers is an eight channel, ten bit analog-to-digital converter. This means that the MSP430 can perform analog-to-digital conversions on up to eight input signals with ten bit resolution. This enables the MCU to measure analog input signals, respond to these inputs, and subsequently perform different tasks using these inputs, which serves as a rather powerful tool in many applications. For example, in a dual-parachute rocket system, the MSP430 can trigger the ejection parachutes given voltage readings from the on-board electronics (accelerometer, altimeter, etc).

The ADC present on the MSP430 has four modes of operation: single channel single-conversion, sequence-of-channels, repeat single-channel, and repeat sequence-of-channels. Below is a table giving a brief description of each mode.

| CONSEQx | Mode | Operation |
|---|---|---|
| 00 | Single channel single-conversion | A single channel is converted once. |
| 01 | Sequence-of-channels | A sequence of channels is converted once. |
| 10 | Repeat single channel | A single channel is converted repeatedly. |
| 11 | Repeat sequence-of-channels | A sequence of channels is converted repeatedly. |

**ADC Conversion Process**

The general order of actions that the ADC performs for each conversion is exhibited in the single channel single-conversion mode:

1. The conversion mode is selected
2. The ADC is turned on
3. The input channel is selected
4. The ADC waits to be enabled or instructed to start conversion
5. When a conversion is triggered, the voltage at the selected input is sampled
6. The voltage is then converted to a digital value
7. This value is written to the ADC memory register
8. An ADC interrupt is triggered

9. The ADC waits again to be instructed to start a conversion

## ADC Registers

The MSP430 has several control registers that are used to configure various operating parameters of the peripheral that they correspond to. The registers that are used by the ADC are listed in the table

| 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 |

## Configuring the ADC

The ADC can be configured with only a few lines of code. An example of simple configuration, written in the C++ programming language, is implemented in the block of code below.

```
void ADC_capture(void)
{
  /* Configure ADC  Channel */
  ADC10CTL0 = ADC10SHT_3 + ADC10ON + ADC10IE; //64 clk ticks, ADC on, enable interrupt
  ADC10CTL1 = ADC10SSEL_0 + INCH_5; //SMCLK, channel 5
  ADC10CTL0 |= ENC + ADC10SC;       //Enable and start conversion
  while ((ADC10CTL1 & ADC10BUSY) == 0x01); //Wait for conversion to end
}
```