Section 22. 12-bit High-Speed Successive Approximation Register (SAR) Analog-to-Digital Converter (ADC)

This section of the manual contains the following major topics:

22.1 Introduction .................................................................................................................. 22-2
22.2 Control Registers .........................................................................................................22-6
22.3 ADC Operation............................................................................................................ 22-61
22.4 ADC Module Configuration ....................................................................................... 22-65
22.5 Additional ADC Functions ....................................................................................... 22-85
22.6 Interrupts .................................................................................................................... 22-108
22.7 Operation During Power-Saving Modes .................................................................. 22-114
22.8 Effects of Reset ......................................................................................................... 22-116
22.9 Transfer Function ...................................................................................................... 22-116
22.10 ADC Sampling Requirements ................................................................................ 22-117
22.11 Connection Considerations ................................................................................... 22-117
22.12 Related Application Notes ..................................................................................... 22-118
22.13 Revision History ..................................................................................................... 22-119
22.1  INTRODUCTION

The PIC32 12-bit High-Speed Successive Approximation Register (SAR) Analog-to-Digital Converter (ADC) includes the following features:

- 12-bit resolution
- Up to eight ADC modules with dedicated Sample and Hold (S&H) circuits (see Note 1)
- Two dedicated ADC modules can be combined in Turbo mode to provide double conversion rate
- Single-ended and/or differential inputs
- Can operate during Sleep mode
- Supports touch sense applications
- Up to six digital comparators
- Up to six digital filters supporting two modes:
  - Oversampling mode
  - Averaging mode
- FIFO and DMA engine for dedicated ADC modules (see Note 2)
- Early interrupt generation resulting in faster processing of converted data
- Designed for motor control, power conversion, and general purpose applications

The dedicated ADC modules use a single input (or its alternate) and is intended for high-speed and precise sampling of time-sensitive or transient inputs, whereas the shared ADC module incorporates a multiplexer on the input to facilitate a larger group of inputs, with slower sampling, and provides flexible automated scanning option through the input scan logic.

For each ADC module, the analog inputs are connected to the S&H capacitor. The clock, sampling time, and output data resolution for each ADC module can be set independently. The ADC module performs the conversion of the input analog signal based on the configurations set in the registers. When conversion is complete, the final result is stored in the result buffer for the specific analog input and is passed to the digital filter and digital comparator if configured to use data from this particular sample.

A simplified block diagram of the ADC module is illustrated in Figure 22-1.
Figure 22-1: ADC Block Diagram

Note: The number of ADC modules, analog inputs, ANa, ANb, ANc, and ANd, and the FIFO and DMA features are shown as an example. Refer to the "ADC" chapter in the specific device data sheet to determine the actual ANx selections, ADC module availability, and the specific FIFO and DMA features.
Figure 22-2: FIFO Block Diagram

Note: The number of ADC modules, analog inputs, ANa, ANb, ANc, and ANd, and the FIFO and DMA features are shown as an example. Refer to the “ADC” chapter in the specific device data sheet to determine the actual ANx selections, ADC module availability, and the specific FIFO and DMA features.
Section 22. 12-bit High-Speed SAR ADC

Note: The number of ADC modules, analog inputs, ANa, ANb, ANc, and ANd, and the FIFO and DMA features are shown as an example. Refer to the “ADC” chapter in the specific device data sheet to determine the actual ANx selections, ADC module availability, and the specific FIFO and DMA features.
CONTROL REGISTERS

The PIC32 12-bit High-Speed SAR ADC module has the following Special Function Registers (SFRs):

- **ADCCON1: ADC Control Register 1**
  This register controls the basic operation of all ADC modules, including behavior in Sleep and Idle modes, and data formatting. This register also specifies the vector shift amounts for the Interrupt Controller. Additional ADCCON1 functions include controlling the Turbo feature of the ADC, the RAM buffer length in DMA mode, and Capacitive Voltage Division (CVD).

- **ADCCON2: ADC Control Register 2**
  This register controls the reference selection for all ADC modules, the sample time for the shared ADC module, interrupt enable for reference, early interrupt selection, and clock division selection for the shared ADC.

- **ADCCON3: ADC Control Register 3**
  This register enables ADC clock selection, enables/disables the digital feature for the dedicated and shared ADC modules and controls the manual (software) sampling and conversion.

- **ADCTRGMODE: ADC Triggering Mode for Dedicated ADC Register**
  This register has selections for alternate analog inputs and includes trigger settings for the dedicated ADC modules.

- **ADCMCON1: ADC Input Mode Control Register 1** through **ADCMCON4: ADC Input Mode Control Register 4**
  These registers enable the user to select between single-ended and differential operation as well as select between signed and unsigned data format.

- **ADCGIRQEN1: ADC Global Interrupt Enable Register 1** and **ADCGIRQEN2: ADC Global Interrupt Enable Register 2**
  These registers specify which of the individual input conversion interrupts can generate the global ADC interrupt.

- **ADCCSS1: ADC Common Scan Select Register 1** and **ADCCSS2: ADC Common Scan Select Register 2**
  These registers specify the analog inputs to be scanned by the common scan trigger.

- **ADCDSTAT1: ADC Data Ready Status Register 1** and **ADCDSTAT2: ADC Data Ready Status Register 2**
  These registers contain the interrupt status of the individual analog input conversions. Each bit represents the data-ready status for its associated conversion result.

- **ADCCMPENx: ADC Digital Comparator 'x' Enable Register ('x' = 1 through 6)**
  These registers select which analog input conversion results will be processed by the digital comparator.

- **ADCCMPx: ADC Digital Comparator 'x' Limit Value Register ('x' = 1 through 6)**
  These registers contain the high and low digital comparison values for use by the digital comparator.

- **ADCFLTRx: ADC Digital Filter 'x' Register ('x' = 1 through 6)**
  These registers provide control and status bits for the oversampling filter accumulator, and also includes the 16-bit filter output data.

- **ADCTR1: ADC Trigger Source 1 Register**
  This register controls the trigger source selection for AN0 through AN3 analog inputs.

- **ADCTR2: ADC Trigger Source 2 Register**
  This register controls the trigger source selection for AN4 through AN7 analog inputs.

- **ADCTR3: ADC Trigger Source 3 Register**
  This register controls the trigger source selection for AN8 through AN11 analog inputs.

- **ADCTR4: ADC Trigger Source 4 Register**
  This register controls the trigger source selection for AN12 through AN15 analog inputs.

- **ADCTR5: ADC Trigger Source 5 Register**
  This register controls the trigger source selection for AN16 through AN19 analog inputs.
Section 22. 12-bit High-Speed SAR ADC

- **ADCTRG6**: ADC Trigger Source 6 Register
  This register controls the trigger source selection for AN20 through AN23 analog inputs.

- **ADCTRG7**: ADC Trigger Source 7 Register
  This register controls the trigger source selection for AN24 through AN27 analog inputs.

- **ADCTRG8**: ADC Trigger Source 8 Register
  This register controls the trigger source selection for AN28 through AN31 analog inputs.

- **ADCCMPCON1**: ADC Digital Comparator 1 Control Register
  This register controls the operation of Digital Comparator 1, including the generation of interrupts, comparison criteria to be used, and provides status when a comparator event occurs. Additionally, this register provides the output data of CVD.

- **ADCCMPCONx**: ADC Digital Comparator ‘x’ Control Register (‘x’ = 2 through 6)
  These registers control the operation of Digital Comparators 2 through 6, including the generation of interrupts and the comparison criteria to be used. This register also provides status when a comparator event occurs.

- **ADCFSTAT**: ADC FIFO Status Register
  This register specifies the status of the dedicated ADC module FIFO.

- **ADCFIFO**: ADC FIFO Data Register
  This register specifies the output value of the dedicated ADC module FIFO.

- **ADCBASE**: ADC Base Register
  These registers specify the base address of the user ADC Interrupt Service Routine (ISR) jump table.

- **ADCDMASTAT**: ADC DMA Status Register
  This register contains the DMA status bits.

- **ADCCNTB**: ADC Sample Count Base Address Register
  This register contains the base address of the sample count in RAM. In addition to storing the converted data of each dedicated ADC module in RAM, DMA also stores the converted sample count.

- **ADCDMAB**: ADC DMA Base Address Register
  This register contains the base address of RAM for the DMA engine.

- **ADCSRGSNS**: ADC Trigger Level/Edge Sensitivity Register
  This register contains the setting for trigger level for each ADC analog input.

- **ADxTIME**: Dedicated ADCx Timing Register ‘x’ (‘x’ = 0 through 6)
  These registers contain the time and clock setting for dedicated analog input.

- **ADCEIEN1**: ADC Early Interrupt Enable Register 1 and
  **ADCEIEN2**: ADC Early Interrupt Enable Register 2
  These registers contain bits to enable or disable early interrupt for individual analog inputs.

- **ADCEISTAT1**: ADC Early Interrupt Status Register 1 and
  **ADCEISTAT2**: ADC Early Interrupt Status Register 2
  These registers contain status bits for early interrupt for individual analog inputs.

- **ADCCANCON**: ADC Analog Warm-up Control Register
  This register contains the warm-up control settings for the analog and bias circuit of the ADC module.

- **ADCDATAx**: ADC Output Data Register (‘x’ = 0 through 63)
  These registers are the analog-to-digital conversion output data registers. The ADCDATAx register is associated with each analog input, 0-63.

- **ADCxCFG**: ADCx Configuration Register ‘x’ (‘x’ = 0 through 7)
  These registers specify the ADC module configuration data.

- **ADCSYSXCFG0**: ADC System Configuration Register 0 and
  **ADCSYSXCFG1**: ADC System Configuration Register 1
  These registers contain read-only bits corresponding to the analog input.
Table 22-1 provides a summary of all ADC Special Function Registers (SFRs). Corresponding registers appear after the summaries, which include a detailed description of each bit. Depending on the device, functionality will vary. Refer to the “ADC” chapter in the specific device data sheet to determine which registers are available for your device.

<table>
<thead>
<tr>
<th>Register Name</th>
<th>Bit Range</th>
<th>Bit 31/15</th>
<th>Bit 30/14</th>
<th>Bit 29/13</th>
<th>Bit 28/12</th>
<th>Bit 27/21</th>
<th>Bit 26/20</th>
<th>Bit 25/19</th>
<th>Bit 24/8</th>
<th>Bit 23/7</th>
<th>Bit 22/6</th>
<th>Bit 21/5</th>
<th>Bit 20/4</th>
<th>Bit 19/3</th>
<th>Bit 18/2</th>
<th>Bit 17/1</th>
<th>Bit 16/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADCCON1</td>
<td>31:16</td>
<td>TRIBEN</td>
<td>TRIBERR</td>
<td>TRMST&lt;2:0&gt;</td>
<td>TRSELV&lt;2:0&gt;</td>
<td>FRACT</td>
<td>SELRES&lt;1:0&gt;</td>
<td>STRSSEL&lt;4:0&gt;</td>
<td>ON</td>
<td>—</td>
<td>SIDL</td>
<td>AICPMPEN</td>
<td>CVDEN</td>
<td>FSSCLKEN</td>
<td>FSSBCLKEN</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>ADCCON2</td>
<td>31:16</td>
<td>BVRORDY</td>
<td>REFPLL</td>
<td>EORRDY</td>
<td>CVDCLP&lt;2:0&gt;</td>
<td>SAMC&lt;6:0&gt;</td>
<td>IRQV&lt;2:0&gt;</td>
<td>STRGLVL</td>
<td>DMALB&lt;2:0&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>ADCCON3</td>
<td>31:16</td>
<td>ADCSEL&lt;1:0&gt;</td>
<td>CONCLKDIV&lt;6:0&gt;</td>
<td>DGEN7</td>
<td>DGEN6</td>
<td>DGEN5</td>
<td>DGEN4</td>
<td>DGEN3</td>
<td>DGEN2</td>
<td>DGEN1</td>
<td>DGEN0</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>ADCTRGMODE</td>
<td>31:16</td>
<td>SHALT&lt;1:0&gt;</td>
<td>SHALT&lt;1:0&gt;</td>
<td>SHALT&lt;1:0&gt;</td>
<td>SHALT&lt;1:0&gt;</td>
<td>SHALT&lt;1:0&gt;</td>
<td>SHALT&lt;1:0&gt;</td>
<td>SHALT&lt;1:0&gt;</td>
<td>SHALT&lt;1:0&gt;</td>
<td>SHALT&lt;1:0&gt;</td>
<td>SHALT&lt;1:0&gt;</td>
<td>SHALT&lt;1:0&gt;</td>
<td>SHALT&lt;1:0&gt;</td>
<td>SHALT&lt;1:0&gt;</td>
<td>SHALT&lt;1:0&gt;</td>
<td>SHALT&lt;1:0&gt;</td>
<td></td>
</tr>
<tr>
<td>ADCEIMCON1</td>
<td>31:16</td>
<td>DIFF15</td>
<td>SIGN15</td>
<td>DIFF14</td>
<td>SIGN14</td>
<td>DIFF13</td>
<td>SIGN13</td>
<td>DIFF12</td>
<td>SIGN12</td>
<td>DIFF11</td>
<td>SIGN11</td>
<td>DIFF10</td>
<td>SIGN10</td>
<td>DIFF9</td>
<td>SIGN9</td>
<td>DIFF8</td>
<td>SIGN8</td>
</tr>
<tr>
<td>ADCEIMCON2</td>
<td>31:16</td>
<td>DIFF31</td>
<td>SIGN31</td>
<td>DIFF30</td>
<td>SIGN30</td>
<td>DIFF29</td>
<td>SIGN29</td>
<td>DIFF28</td>
<td>SIGN28</td>
<td>DIFF27</td>
<td>SIGN27</td>
<td>DIFF26</td>
<td>SIGN26</td>
<td>DIFF25</td>
<td>SIGN25</td>
<td>DIFF24</td>
<td>SIGN24</td>
</tr>
<tr>
<td>ADCEIMCON3</td>
<td>31:16</td>
<td>DIFF59</td>
<td>SIGN59</td>
<td>DIFF58</td>
<td>SIGN58</td>
<td>DIFF57</td>
<td>SIGN57</td>
<td>DIFF56</td>
<td>SIGN56</td>
<td>DIFF55</td>
<td>SIGN55</td>
<td>DIFF54</td>
<td>SIGN54</td>
<td>DIFF53</td>
<td>SIGN53</td>
<td>DIFF52</td>
<td>SIGN52</td>
</tr>
<tr>
<td>ADCEIMCON4</td>
<td>31:16</td>
<td>DIFF79</td>
<td>SIGN79</td>
<td>DIFF78</td>
<td>SIGN78</td>
<td>DIFF77</td>
<td>SIGN77</td>
<td>DIFF76</td>
<td>SIGN76</td>
<td>DIFF75</td>
<td>SIGN75</td>
<td>DIFF74</td>
<td>SIGN74</td>
<td>DIFF73</td>
<td>SIGN73</td>
<td>DIFF72</td>
<td>SIGN72</td>
</tr>
<tr>
<td>ADCCSS1</td>
<td>31:16</td>
<td>CSS31</td>
<td>CSS30</td>
<td>CSS29</td>
<td>CSS28</td>
<td>CSS27</td>
<td>CSS26</td>
<td>CSS25</td>
<td>CSS24</td>
<td>CSS23</td>
<td>CSS22</td>
<td>CSS21</td>
<td>CSS20</td>
<td>CSS19</td>
<td>CSS18</td>
<td>CSS17</td>
<td>CSS16</td>
</tr>
<tr>
<td>ADCCSS2</td>
<td>31:16</td>
<td>CSS61</td>
<td>CSS60</td>
<td>CSS59</td>
<td>CSS58</td>
<td>CSS57</td>
<td>CSS56</td>
<td>CSS55</td>
<td>CSS54</td>
<td>CSS53</td>
<td>CSS52</td>
<td>CSS51</td>
<td>CSS50</td>
<td>CSS49</td>
<td>CSS48</td>
<td>CSS47</td>
<td>CSS46</td>
</tr>
<tr>
<td>ADCDSAT1</td>
<td>31:16</td>
<td>ARDY31</td>
<td>ARDY30</td>
<td>ARDY29</td>
<td>ARDY28</td>
<td>ARDY27</td>
<td>ARDY26</td>
<td>ARDY25</td>
<td>ARDY24</td>
<td>ARDY23</td>
<td>ARDY22</td>
<td>ARDY21</td>
<td>ARDY20</td>
<td>ARDY19</td>
<td>ARDY18</td>
<td>ARDY17</td>
<td>ARDY16</td>
</tr>
<tr>
<td>ADCDSAT2</td>
<td>31:16</td>
<td>ARDY61</td>
<td>ARDY60</td>
<td>ARDY59</td>
<td>ARDY58</td>
<td>ARDY57</td>
<td>ARDY56</td>
<td>ARDY55</td>
<td>ARDY54</td>
<td>ARDY53</td>
<td>ARDY52</td>
<td>ARDY51</td>
<td>ARDY50</td>
<td>ARDY49</td>
<td>ARDY48</td>
<td>ARDY47</td>
<td>ARDY46</td>
</tr>
<tr>
<td>ADCCOMPCx</td>
<td>'x' = 1-6</td>
<td>AFEN</td>
<td>DATA16EN</td>
<td>DFMODE</td>
<td>OVRSM&lt;2:0&gt;</td>
<td>AFGEN</td>
<td>AFRDY</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>ADCLKT1x</td>
<td>'x' = 1-6</td>
<td>DCPH&lt;16:0&gt;</td>
<td>DCMPL&lt;15:0&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

Before enabling the ADC, the user application must initialize the ADC calibration values by copying them from the factory-programmed DEVADcx Flash registers into the corresponding ADCCxCFG registers.
### Table 22-1: ADC SFR Summary

<table>
<thead>
<tr>
<th>Register Name</th>
<th>Bit Range</th>
<th>Bit 31/15</th>
<th>Bit 30/14</th>
<th>Bit 29/13</th>
<th>Bit 28/12</th>
<th>Bit 27/11</th>
<th>Bit 26/10</th>
<th>Bit 25/9</th>
<th>Bit 24/8</th>
<th>Bit 23/7</th>
<th>Bit 22/6</th>
<th>Bit 21/5</th>
<th>Bit 20/4</th>
<th>Bit 19/3</th>
<th>Bit 18/2</th>
<th>Bit 17/1</th>
<th>Bit 16/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADCTRG6</td>
<td>31:16</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>15:0</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>ADCTRG7</td>
<td>31:16</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>15:0</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>ADCTRG8</td>
<td>31:16</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>15:0</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>ADCCMPCON1</td>
<td>31:16</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>15:0</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>ADCCMPCONx</td>
<td>x = 2-6</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>15:0</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>ADCFSAT</td>
<td>31:16</td>
<td>FIEN</td>
<td>ADC3EN</td>
<td>ADC2EN</td>
<td>ADC1EN</td>
<td>ADC0EN</td>
<td>FIEN</td>
<td>FRDY</td>
<td>FWRROVRN</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>15:0</td>
<td>FCR&lt;7:0&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>ADCFIFO</td>
<td>31:16</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>15:0</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>ADCMBASE</td>
<td>31:16</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>15:0</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>ADCDMASTAT</td>
<td>31:16</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>15:0</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>ADCDSTAT</td>
<td>31:16</td>
<td>FEN</td>
<td>ADC3EN</td>
<td>ADC2EN</td>
<td>ADC1EN</td>
<td>ADC0EN</td>
<td>FEN</td>
<td>FRDY</td>
<td>FWRROVRN</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>15:0</td>
<td>FCR&lt;7:0&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>ADCDSYSCFG0</td>
<td>31:16</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>15:0</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>ADCDSYSCFG1</td>
<td>31:16</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>15:0</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Note 1:</td>
<td></td>
<td>Before enabling the ADC, the user application must initialize the ADC calibration values by copying them from the factory-programmed DEVADCx Flash registers into the corresponding ADCxCFG registers.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note 1:** Before enabling the ADC, the user application must initialize the ADC calibration values by copying them from the factory-programmed DEVADCx Flash registers into the corresponding ADCxCFG registers.
Register 22-1:  ADCCON1: ADC Control Register 1

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Bit 31/23/15/7</th>
<th>Bit 29/21/13/5</th>
<th>Bit 27/19/11/3</th>
<th>Bit 25/17/9/1</th>
<th>Bit 24/16/8/0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td></td>
<td>TRBEN</td>
<td>TRBERR</td>
<td>TRBMST&lt;2:0&gt;</td>
<td>TRBSLV&lt;2:0&gt;</td>
<td></td>
</tr>
<tr>
<td>23:16</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td></td>
<td>FRACT</td>
<td>SELRES&lt;1:0&gt;</td>
<td>STRGSRC&lt;4:0&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:8</td>
<td>R/W-0</td>
<td>U-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>U-0</td>
</tr>
<tr>
<td></td>
<td>ON</td>
<td>SIDL</td>
<td>AICPMEN</td>
<td>CVDEN</td>
<td>FSSCLKEN</td>
</tr>
<tr>
<td>7:0</td>
<td>U-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td></td>
<td>— IRQVS&lt;2:0&gt;</td>
<td>STRGLVL</td>
<td>DMABL&lt;2:0&gt;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend:
- HC = Hardware Set
- HS = Hardware Cleared
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as '0'
- -n = Value at POR
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- x = Bit is unknown

bit 31  TRBEN: Turbo Channel Enable bit
1 = Enable the Turbo channel
0 = Disable the Turbo channel

bit 30  TRBERR: Turbo Channel Error Status bit
1 = An error occurred while setting the Turbo channel and Turbo channel function to be disabled regardless of the TRBEN bit being set to '1'.
0 = Turbo channel error did not occur

Note: The status of this bit is valid only after the TRBEN bit is set.

bit 29-27  TRBMST<2:0>: Turbo Master ADCx bits
111 = Reserved
110 = ADC6 is selected as the Turbo Master
...  
000 = ADC0 is selected as the Turbo Master

bit 26-24  TRBSLV<2:0>: Turbo Slave ADCx bits
111 = Reserved
110 = ADC6 is selected as the Turbo Slave
...  
000 = ADC0 is selected as the Turbo Slave

bit 23  FRACT: Fractional Data Output Format bit
1 = Fractional
0 = Integer

bit 22-21  SELRES<1:0>: Shared ADC Resolution bits
11 = 12 bits (default)
10 = 10 bits
01 = 8 bits
00 = 6 bits

bit 20-16  STRGSRC<4:0>: Scan Trigger Source Select bits
11111 - 00100 = Refer to the “ADC” chapter in the specific device data sheet for trigger source selections
00011 = Reserved
00010 = Global level software trigger (GLSWTRG) is not self-cleared
00001 = Global software trigger (GSWTRG) is self-cleared on the next clock cycle
00000 = No trigger

bit 15  ON: ADC Module Enable bit
1 = ADC module is enabled
0 = ADC module is disabled

Note: The ON bit should be set only after the ADC module has been configured.

bit 14  Unimplemented: Read as '0'
Register 22-1: ADCCON1: ADC Control Register 1 (Continued)

bit 13  **SIDL:** Stop in Idle Mode bit
   1 = Discontinue module operation when the device enters Idle mode
   0 = Continue module operation in Idle mode

bit 12  **AICPMPEN:** Analog Input Charge Pump Enable bit
   1 = Analog input charge pump is enabled (default)
   0 = Analog input charge pump is disabled

bit 11  **CVDEN:** Capacitive Voltage Division Enable bit
   1 = CVD operation is enabled
   0 = CVD operation is disabled

bit 10  **FSSCLKEN:** Fast Synchronous System Clock to ADC Control Clock bit
   1 = Fast synchronous system clock to ADC control clock is enabled
   0 = Fast synchronous system clock to ADC control clock is disabled

bit 9   **FSPBCLKEN:** Fast Synchronous Peripheral Clock to ADC Control Clock bit
   1 = Fast synchronous peripheral clock to ADC control clock is enabled
   0 = Fast synchronous peripheral clock to ADC control clock is disabled

bit 8-7  Unimplemented: Read as '0'

bit 6-4  **IRQVS<2:0>:** Interrupt Vector Shift bits
   To determine interrupt vector address, this bit specifies the amount of left shift done to the ARDYx status bits in the ADCDSTAT1 and ADCDSTAT2 registers, prior to adding with the ADCBASE register (see 22.6.2 “ADC Base Register (ADCBASE) Usage” for more information).
   
   Interrupt Vector Address = Read Value of ADCBASE = Value written to ADCBASE + x << IRQVS<2:0>,
   where 'x' is the smallest active input ID from the ADCDSTAT1 or ADCDSTAT2 registers (which has highest priority).
   
   111 = Shift x left 7 bit position
   110 = Shift x left 6 bit position
   101 = Shift x left 5 bit position
   100 = Shift x left 4 bit position
   011 = Shift x left 3 bit position
   010 = Shift x left 2 bit position
   001 = Shift x left 1 bit position
   000 = Shift x left 0 bit position

bit 3   **STRGLVL:** Scan Trigger High Level/Positive Edge Sensitivity bit
   1 = Scan trigger is high level sensitive. Once STRIG mode is selected (TRGSRCx<4:0> in the ADCTRGx register), the scan trigger will continue for all selected analog inputs, until the STRIG option is removed.
   0 = Scan trigger is positive edge sensitive. Once STRIG mode is selected (TRGSRCx<4:0> in the ADCTRGx register), only a single scan trigger will be generated, which will complete the scan of all selected analog inputs.

bit 2-0  **DMABL<2:0>:** DMA Buffer Length Size bits
   111 = Allocates 128 locations in RAM to each analog input
   110 = Allocates 64 locations in RAM to each analog input
   101 = Allocates 32 locations in RAM to each analog input
   100 = Allocates 16 locations in RAM to each analog input
   011 = Allocates 8 locations in RAM to each analog input
   010 = Allocates 4 locations in RAM to each analog input
   001 = Allocates 2 locations in RAM to each analog input
   000 = Allocates 1 location in RAM to each analog input

   **Note:** Since each output data is 16-bit wide, one location consists of 2 bytes.
## Register 22-2: ADCCON2: ADC Control Register 2

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Bit 31/23/15/7</th>
<th>Bit 30/22/14/6</th>
<th>Bit 29/21/13/5</th>
<th>Bit 28/20/12/4</th>
<th>Bit 27/19/11/3</th>
<th>Bit 26/18/10/2</th>
<th>Bit 25/17/9/1</th>
<th>Bit 24/16/8/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BCHVRRDY</td>
<td>REFFLT</td>
<td>EOSRDY</td>
<td>CVDCPL&lt;2:0&gt;</td>
<td>SAMC&lt;9:0&gt;</td>
</tr>
<tr>
<td>23:16</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SAMC&lt;7:0&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:8</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td></td>
<td>BGVRIEN</td>
<td>REFFLTIEN</td>
<td>EOSIEN</td>
<td>ADCEIOVR</td>
<td>ECRIEN</td>
<td>ADCEIS&lt;2:0&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td>U-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
</tbody>
</table>

Legend:  
HC = Hardware Set  
HS = Hardware Cleared  
r = Reserved  
R = Readable bit  
W = Writable bit  
U = Unimplemented bit, read as '0'  
-<n> = Value at POR  
‘1’ = Bit is set  
‘0’ = Bit is cleared  
x = Bit is unknown

**bit 31** BGVRRDY: Band Gap Voltage/ADC Reference Voltage Status bit  
1 = Both band gap voltage and ADC reference voltages (VREF) are ready  
0 = Either or both band gap voltage and ADC reference voltages (VREF) are not ready  
Data processing is valid only after the BGVRRDY bit is set by hardware, hence the application code must check that the BGVRRDY bit is set to ensure data validity. This bit is set to ‘0’ when the ON bit (ADCCON1<15>) = 0.

**bit 30** REFFLT: Band Gap/VREF/AVDD BOR Fault Status bit  
1 = Fault in band gap or the VREF voltage while the ON bit (ADCCON1<15>) was set. Most likely a band gap or VREF fault will be caused by a BOR of the analog VDD supply.  
0 = Band gap and VREF voltage are working properly  
This bit is cleared when the ON bit (ADCCON1<15>) = 0 and the BGVRRDY bit = 1.

**bit 29** EOSRDY: End of Scan Interrupt Status bit  
1 = All analog inputs are considered for scanning through the scan trigger (all analog inputs specified in the ADCCSS1 and ADCCSS2 registers) have completed scanning  
0 = Scanning has not completed  
This bit is cleared when ADCCON2<31:24> are read in software.

**bit 28-26** CVDCPL<2:0>: Capacitor Voltage Divider (CVD) Setting bit  
111 = 7 * 2.5 pF = 17.5 pF  
110 = 6 * 2.5 pF = 15 pF  
101 = 5 * 2.5 pF = 12.5 pF  
100 = 4 * 2.5 pF = 10 pF  
011 = 3 * 2.5 pF = 7.5 pF  
010 = 2 * 2.5 pF = 5 pF  
001 = 1 * 2.5 pF = 2.5 pF  
000 = 0 * 2.5 pF = 0 pF

**bit 25-16** SAMC<9:0>: Sample Time for the Shared ADC bits  
1111111111 = 1025 TAD  
0000000000 = 2 TAD  
Where TAD = period of the ADC conversion clock for the Shared ADC controlled by the ADCDIV<6:0> bits.

**bit 15** BGVRIEN: Band Gap/VREF Voltage Ready Interrupt Enable bit  
1 = Interrupt will be generated when the BGVRRDY bit is set  
0 = No interrupt is generated when the BGVRRDY bit is set
Section 22. 12-bit High-Speed SAR ADC

Register 22-2: ADCCON2: ADC Control Register 2 (Continued)

bit 14  REFFLTIE: Band Gap/VREF Voltage Fault Interrupt Enable bit
       1 = Interrupt will be generated when the REFFLT bit is set
       0 = No interrupt is generated when the REFFLT bit is set

bit 13  EOSIEN: End of Scan Interrupt Enable bit
       1 = Interrupt will be generated when EOSRDY bit is set
       0 = No interrupt is generated when the EOSRDY bit is set

bit 12  ADCEIOVR: Early Interrupt Request Override bit
       1 = Early interrupt generation is overridden and interrupt generation is controlled by the ADCGIRQEN1
           and ADCGIRQEN2 registers
       0 = Early interrupt generation is not overridden and interrupt generation is controlled by the ADCEIEN1
           and ADCEIEN2 registers

bit 11  ECRRI: External Conversion Request Interface Enable bit
       1 = Enables ADC conversion start from external module (such as PTG)
       0 = External modules cannot start ADC conversion

bit 10-8  ADCEIS<2:0>: Shared ADC Early Interrupt Select bits
       These bits select the number of clocks (TAD) prior to the arrival of valid data that the associated interrupt
       is generated.
       111 = The data ready interrupt is generated 8 ADC clocks prior to end of conversion
       110 = The data ready interrupt is generated 7 ADC clocks prior to end of conversion
       101 = The data ready interrupt is generated 6 ADC module clocks prior to end of conversion
       100 = The data ready interrupt is generated 5 ADC module clocks prior to end of conversion
       011 = The data ready interrupt is generated 2 ADC module clocks prior to end of conversion
       010 = The data ready interrupt is generated 1 ADC module clock prior to end of conversion
       001 = The data ready interrupt is generated 2 ADC module clocks prior to end of conversion
       000 = The data ready interrupt is generated 1 ADC module clock prior to end of conversion

Note: All options are available when the selected resolution, set by the SELRES<1:0> bits
       (ADCCON1<22:21>), is 12-bit or 10-bit. For a selected resolution of 8-bit, options from '000' to
       '101' are valid. For a selected resolution of 6-bit, options from '000' to '011' are valid.

bit 7  Unimplemented: Read as '0'

bit 6-0  ADCDIV<6:0>: Shared ADC Clock Divider bits
       1111111 = 254 * TQ = TAD
       1111110 = 252 * TQ = TAD
       1111101 = 250 * TQ = TAD
       1111100 = 248 * TQ = TAD
       1111011 = 246 * TQ = TAD
       1111010 = 244 * TQ = TAD
       1111001 = 242 * TQ = TAD
       1111000 = 240 * TQ = TAD
       1110111 = 238 * TQ = TAD
       1110110 = 236 * TQ = TAD
       1110101 = 234 * TQ = TAD
       1110100 = 232 * TQ = TAD
       1110011 = 230 * TQ = TAD
       1110010 = 228 * TQ = TAD
       1110001 = 226 * TQ = TAD
       1110000 = 224 * TQ = TAD

The ADCDIV<6:0> bits divide the ADC control clock (TQ) to generate the clock for the Shared ADC (TAD).
## Register 22-3: ADCCON3: ADC Control Register 3

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Bit 31/23/15/7</th>
<th>Bit 30/22/14/6</th>
<th>Bit 29/21/13/5</th>
<th>Bit 28/20/12/4</th>
<th>Bit 27/19/11/3</th>
<th>Bit 26/18/10/2</th>
<th>Bit 25/17/9/1</th>
<th>Bit 24/16/8/0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>31:24</td>
<td>ADCSEL&lt;1:0&gt;</td>
<td>CONCLKDIV&lt;5:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23:16</td>
<td>DIGEN7(5)</td>
<td>DIGEN6(5)</td>
<td>DIGEN5(5)</td>
<td>DIGEN4(5)</td>
<td>DIGEN3(5)</td>
<td>DIGEN2(5)</td>
<td>DIGEN1(5)</td>
<td>DIGEN0(5)</td>
</tr>
<tr>
<td>15:8</td>
<td>VREFSEL&lt;2:0&gt;</td>
<td>TRGSUSP</td>
<td>UPDIEN</td>
<td>UPDRDY</td>
<td>SAMP(1,2,3,4)</td>
<td>RQCNVRT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td>GLSWTRG</td>
<td>GSWTRG</td>
<td>ADINSEL&lt;5:0&gt;(5)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Legend:**
- **HC** = Hardware Set
- **HS** = Hardware Cleared
- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as '0'
- **-n** = Value at POR
  - '1' = Bit is set
  - '0' = Bit is cleared
  - **x** = Bit is unknown

### Bit 31-30: ADCSEL<1:0>
- Analog-to-Digital Clock Source (TCLK) bits
- Refer to the “12-bit High-Speed Successive Approximation Register (SAR)” chapter in the specific device data sheet for the ADC Clock source selections.

### Bit 29-24: CONCLKDIV<5:0>
- Analog-to-Digital Control Clock (TQ) Divider bits

- \[111111 = 126 \times \text{TCLK} = \text{TQ}\]
- \[000011 = 6 \times \text{TCLK} = \text{TQ}\]
- \[000010 = 4 \times \text{TCLK} = \text{TQ}\]
- \[000001 = 2 \times \text{TCLK} = \text{TQ}\]
- \[000000 = \text{TCLK} = \text{TQ}\]

### Bit 23: DIGEN7: ADC7 Digital Enable bit(5)
- 1 = ADC7 is digital enabled
- 0 = ADC7 is digital disabled

### Bit 22: DIGEN6: ADC6 Digital Enable bit(5)
- 1 = ADC6 is digital enabled
- 0 = ADC6 is digital disabled

### Bit 21: DIGEN5: ADC5 Digital Enable bit(5)
- 1 = ADC5 is digital enabled
- 0 = ADC5 is digital disabled

### Bit 20: DIGEN4: ADC4 Digital Enable bit(5)
- 1 = ADC4 is digital enabled
- 0 = ADC4 is digital disabled

**Note 1:**
- The SAMP bit has the highest priority and setting this bit will keep the S&H circuit in Sample mode until the bit is cleared. Also, usage of the SAMP bit will cause settings of the SAMC<9:0> bits (ADCCON2<25:16>) to be ignored.
- The SAMP bit only connects Class 2 and Class 3 analog inputs to the shared ADC. All Class 1 analog inputs are not affected by the SAMP bit.
- The SAMP bit is not a self-clearing bit and it is the responsibility of application software to first clear this bit and only after setting the RQCNVRT bit to start the analog-to-digital conversion.
- Normally, when the SAMP and RQCNVRT bits are used by software routines, all TRGSRC<4:0> bits and STRGSRC<4:0> bits should be set to ‘000000’ to disable all external hardware triggers and prevent them from interfering with the software-controlled sampling command signal SAMP and with the software-controlled trigger RQCNVRT.
- Depending on the device, the function will vary. Refer to the “ADC” chapter in the specific device data sheet to determine the function that is available for your device.
Register 22-3: ADCCON3: ADC Control Register 3 (Continued)

bit 19  DIGEN3: ADC3 Digital Enable bit
        1 = ADC3 is digital enabled
        0 = ADC3 is digital disabled

bit 18  DIGEN2: ADC2 Digital Enable bit
        1 = ADC2 is digital enabled
        0 = ADC2 is digital disabled

bit 17  DIGEN1: ADC1 Digital Enable bit
        1 = ADC1 is digital enabled
        0 = ADC1 is digital disabled

bit 16  DIGEN0: ADC0 Digital Enable bit
        1 = ADC0 is digital enabled
        0 = ADC0 is digital disabled

bit 15-13  VREFSEL<2:0>: Voltage Reference (VREF) Input Selection bits

<table>
<thead>
<tr>
<th>VREFSEL&lt;2:0&gt;</th>
<th>ADREF+</th>
<th>ADREF-</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>AVDD</td>
<td>Internal VREFL</td>
</tr>
<tr>
<td>110</td>
<td>Internal VREFH</td>
<td>AVSS</td>
</tr>
<tr>
<td>101</td>
<td>Internal VREFH</td>
<td>External VREFL</td>
</tr>
<tr>
<td>100</td>
<td>Internal VREFH</td>
<td>Internal VREFL</td>
</tr>
<tr>
<td>011</td>
<td>External VREFH</td>
<td>External VREFL</td>
</tr>
<tr>
<td>010</td>
<td>AVDD</td>
<td>External VREFL</td>
</tr>
<tr>
<td>001</td>
<td>External VREFH</td>
<td>AVSS</td>
</tr>
<tr>
<td>000</td>
<td>AVDD</td>
<td>AVSS</td>
</tr>
</tbody>
</table>

bit 12  TRGSUSP: Trigger Suspend bit
        1 = Triggers are blocked from starting a new analog-to-digital conversion, but the ADC module is not disabled
        0 = Triggers are not blocked

bit 11  UPDIEN: Update Ready Interrupt Enable bit
        1 = Interrupt will be generated when the UPDRDY bit is set by hardware
        0 = No interrupt is generated

bit 10  UPDRDY: ADC Update Ready Status bit
        1 = ADC SFRs can be updated
        0 = ADC SFRs cannot be updated

Note: This bit is only active while the TRGSUSP bit is set and there are no more running conversions of
any ADC modules.

bit 9  SAMP: Class 2 and Class 3 Analog Input Sampling Enable bit
        1 = The ADC S&H amplifier is sampling
        0 = The ADC S&H amplifier is holding

Note 1: The SAMP bit has the highest priority and setting this bit will keep the S&H circuit in Sample mode until the
bit is cleared. Also, usage of the SAMP bit will cause settings of the SAMC<9:0> bits (ADCCON2<25:16>)
to be ignored.

2: The SAMP bit only connects Class 2 and Class 3 analog inputs to the shared ADC. All Class 1 analog
inputs are not affected by the SAMP bit.

3: The SAMP bit is not a self-clearing bit and it is the responsibility of application software to first clear this bit
and only after setting the RQCNVRT bit to start the analog-to-digital conversion.

4: Normally, when the SAMP and RQCNVRT bits are used by software routines, all TRGSRCx<4:0> bits and
STRGSR<4:0> bits should be set to '00000' to disable all external hardware triggers and prevent them
from interfering with the software-controlled sampling command signal SAMP and with the
software-controlled trigger RQCNVRT.

5: Depending on the device, the function will vary. Refer to the “ADC” chapter in the specific device data
data sheet to determine the function that is available for your device.
Register 22-3: ADCCON3: ADC Control Register 3 (Continued)

bit 8  RQCNVRT: Individual ADC Input Conversion Request bit

This bit and its associated ADINSEL<5:0> bits enable the user to individually request an analog-to-digital conversion of an analog input through software.

1 = Trigger the conversion of the selected ADC input as specified by the ADINSEL<5:0> bits
0 = Do not trigger the conversion

Note: This bit is automatically cleared in the next ADC clock cycle.

bit 7  GLSWTRG: Global Level Software Trigger bit

1 = Trigger conversion for ADC inputs that have selected the GLSWTRG bit as the trigger signal, either through the associated TRGSR<4:0> bits in the ADCTRGx registers or through the STRGSR<4:0> bits in the ADCCON1 register
0 = Do not trigger an analog-to-digital conversion

bit 6  GSWTRG: Global Software Trigger bit

1 = Trigger conversion for ADC inputs that have selected the GSWTRG bit as the trigger signal, either through the associated TRGSR<4:0> bits in the ADCTRGx registers or through the STRGSR<4:0> bits in the ADCCON1 register
0 = Do not trigger an analog-to-digital conversion

Note: This bit is automatically cleared in the next ADC clock cycle.

bit 5-0  ADINSEL<5:0>: Analog Input Select bits

These bits select the analog input to be converted when the RQCNVRT bit is set, where MAX_AN_INPUT is the maximum analog inputs available on the device.

\[
\begin{align*}
\text{MAX_AN_INPUT} + 4 & = \text{Device dependent (see Note 5)} \\
\text{MAX_AN_INPUT} + 3 & = \text{Device dependent (see Note 5)} \\
\text{MAX_AN_INPUT} + 2 & = \text{Device dependent (see Note 5)} \\
\text{MAX_AN_INPUT} + 1 & = \text{Device dependent (see Note 5)} \\
\text{MAX_AN_INPUT} & = \text{AN[MAX_AN_INPUT]} \\
& \vdots \\
00001 & = \text{AN1} \\
00000 & = \text{AN0}
\end{align*}
\]

Note 1: The SAMP bit has the highest priority and setting this bit will keep the S&H circuit in Sample mode until the bit is cleared. Also, usage of the SAMP bit will cause settings of the SAMC<9:0> bits (ADCCON2<25:16>) to be ignored.

2: The SAMP bit only connects Class 2 and Class 3 analog inputs to the shared ADC. All Class 1 analog inputs are not affected by the SAMP bit.

3: The SAMP bit is not a self-clearing bit and it is the responsibility of application software to first clear this bit and only after setting the RQCNVRT bit to start the analog-to-digital conversion.

4: Normally, when the SAMP and RQCNVRT bits are used by software routines, all TRGSR<4:0> bits and STRGSR<4:0> bits should be set to '0000000000' to disable all external hardware triggers and prevent them from interfering with the software-controlled sampling command signal SAMP and with the software-controlled trigger RQCNVRT.

5: Depending on the device, the function will vary. Refer to the “ADC” chapter in the specific device data sheet to determine the function that is available for your device.
### Section 22. 12-bit High-Speed SAR ADC

**Register 22-4: ADCTRGMODE: ADC Triggering Mode for Dedicated ADC Register**

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Bit 31/23/15/7</th>
<th>Bit 29/21/13/5</th>
<th>Bit 27/19/11/3</th>
<th>Bit 25/17/9/1</th>
<th>Bit 24/16/8/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>U-0</td>
<td>U-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>23:16</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>15:8</td>
<td>SH6ALT&lt;1:0&gt;</td>
<td>SH5ALT&lt;1:0&gt;</td>
<td>SH4ALT&lt;1:0&gt;</td>
<td>SH3ALT&lt;1:0&gt;</td>
<td>SH2ALT&lt;1:0&gt;</td>
</tr>
<tr>
<td>7:0</td>
<td>U-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
</tbody>
</table>

**Legend:**

- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as ‘0’
- -n = Value at POR
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- x = Bit is unknown

Bit 31-30 Unimplemented: Read as ‘0’

Bit 29-28 SH6ALT<1:0>: ADC6 Analog Input Select bit

- 11 = 01 = Refer to the “ADC” chapter in the specific device data sheet for the available selections
- 00 = AN6

Bit 27-26 SH5ALT<1:0>: ADC5 Analog Input Select bit

- 11 = 01 = Refer to the “ADC” chapter in the specific device data sheet for the available selections
- 00 = AN5

Bit 25-24 SH4ALT<1:0>: ADC4 Analog Input Select bit

- 11 = 01 = Refer to the “ADC” chapter in the specific device data sheet for the available selections
- 00 = AN4

Bit 23-22 SH3ALT<1:0>: ADC3 Analog Input Select bit

- 11 = 01 = Refer to the “ADC” chapter in the specific device data sheet for the available selections
- 00 = AN3

Bit 21-20 SH2ALT<1:0>: ADC2 Analog Input Select bit

- 11 = 01 = Refer to the “ADC” chapter in the specific device data sheet for the available selections
- 00 = AN2

Bit 19-18 SH1ALT<1:0>: ADC1 Analog Input Select bit

- 11 = 01 = Refer to the “ADC” chapter in the specific device data sheet for the available selections
- 00 = AN1

Bit 17-16 SH0ALT<1:0>: ADC0 Analog Input Select bit

- 11 = 01 = Refer to the “ADC” chapter in the specific device data sheet for the available selections
- 00 = AN0

Bit 15 Unimplemented: Read as ‘0’

Bit 14 STRGEN6: ADC6 Presynchronized Triggers bit

- 1 = ADC6 uses presynchronized triggers
- 0 = ADC6 does not use presynchronized triggers

Bit 13 STRGEN5: ADC5 Presynchronized Triggers bit

- 1 = ADC5 uses presynchronized triggers
- 0 = ADC5 does not use presynchronized triggers

Bit 12 STRGEN4: ADC4 Presynchronized Triggers bit

- 1 = ADC4 uses presynchronized triggers
- 0 = ADC4 does not use presynchronized triggers

Bit 11 STRGEN3: ADC3 Presynchronized Triggers bit

- 1 = ADC3 uses presynchronized triggers
- 0 = ADC3 does not use presynchronized triggers
Register 22-4: ADCTRGMODE: ADC Triggering Mode for Dedicated ADC Register (Continued)

bit 10  **STRGEN2**: ADC2 Presynchronized Triggers bit
       1 = ADC2 uses presynchronized triggers
       0 = ADC2 does not use presynchronized triggers

bit 9  **STRGEN1**: ADC1 Presynchronized Triggers bit
       1 = ADC1 uses presynchronized triggers
       0 = ADC1 does not use presynchronized triggers

bit 8  **STRGEN0**: ADC0 Presynchronized Triggers bit
       1 = ADC0 uses presynchronized triggers
       0 = ADC0 does not use presynchronized triggers

bit 7  **Unimplemented**: Read as ‘ ‘

bit 6  **SSAMPEN6**: ADC6 Synchronous Sampling bit
       1 = ADC6 uses synchronous sampling for the first sample after being idle or disabled
       0 = ADC6 does not use synchronous sampling

bit 5  **SSAMPEN5**: ADC5 Synchronous Sampling bit
       1 = ADC5 uses synchronous sampling for the first sample after being idle or disabled
       0 = ADC5 does not use synchronous sampling

bit 4  **SSAMPEN4**: ADC4 Synchronous Sampling bit
       1 = ADC4 uses synchronous sampling for the first sample after being idle or disabled
       0 = ADC4 does not use synchronous sampling

bit 3  **SSAMPEN3**: ADC3 Synchronous Sampling bit
       1 = ADC3 uses synchronous sampling for the first sample after being idle or disabled
       0 = ADC3 does not use synchronous sampling

bit 2  **SSAMPEN2**: ADC2 Synchronous Sampling bit
       1 = ADC2 uses synchronous sampling for the first sample after being idle or disabled
       0 = ADC2 does not use synchronous sampling

bit 1  **SSAMPEN1**: ADC1 Synchronous Sampling bit
       1 = ADC1 uses synchronous sampling for the first sample after being idle or disabled
       0 = ADC1 does not use synchronous sampling

bit 0  **SSAMPEN0**: ADC0 Synchronous Sampling bit
       1 = ADC0 uses synchronous sampling for the first sample after being idle or disabled
       0 = ADC0 does not use synchronous sampling
### Register 22-5: ADCIMCON1: ADC Input Mode Control Register 1

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Bit 31/23/15/7</th>
<th>Bit 30/22/14/6</th>
<th>Bit 29/21/13/5</th>
<th>Bit 28/20/12/4</th>
<th>Bit 27/19/11/3</th>
<th>Bit 26/18/10/2</th>
<th>Bit 25/17/9/1</th>
<th>Bit 24/16/8/0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>31:24</td>
<td>DIFF15</td>
<td>SIGN15</td>
<td>DIFF14</td>
<td>SIGN14</td>
<td>DIFF13</td>
<td>SIGN13</td>
<td>DIFF12</td>
<td>SIGN12</td>
</tr>
<tr>
<td>23:16</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td></td>
<td>DIFF11</td>
<td>SIGN11</td>
<td>DIFF10</td>
<td>SIGN10</td>
<td>DIFF9</td>
<td>SIGN9</td>
<td>DIFF8</td>
<td>SIGN8</td>
</tr>
<tr>
<td>15:8</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td></td>
<td>DIFF7</td>
<td>SIGN7</td>
<td>DIFF6</td>
<td>SIGN6</td>
<td>DIFF5</td>
<td>SIGN5</td>
<td>DIFF4</td>
<td>SIGN4</td>
</tr>
<tr>
<td>7:0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td></td>
<td>DIFF3</td>
<td>SIGN3</td>
<td>DIFF2</td>
<td>SIGN2</td>
<td>DIFF1</td>
<td>SIGN1</td>
<td>DIFF0</td>
<td>SIGN0</td>
</tr>
</tbody>
</table>

#### Legend:
- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as ‘0’
- **-n** = Value at POR
  - ‘1’ = Bit is set
  - ‘0’ = Bit is cleared
  - **x** = Bit is unknown

#### Bit Descriptions:
- **bit 31**
  - **DIFF15**: AN15 Mode bit
    - 1 = AN15 is using Differential mode
    - 0 = AN15 is using Single-ended mode

- **bit 30**
  - **SIGN15**: AN15 Signed Data Mode bit
    - 1 = AN15 is using Signed Data mode
    - 0 = AN15 is using Unsigned Data mode

- **bit 29**
  - **DIFF14**: AN14 Mode bit
    - 1 = AN14 is using Differential mode
    - 0 = AN14 is using Single-ended mode

- **bit 28**
  - **SIGN14**: AN14 Signed Data Mode bit
    - 1 = AN14 is using Signed Data mode
    - 0 = AN14 is using Unsigned Data mode

- **bit 27**
  - **DIFF13**: AN13 Mode bit
    - 1 = AN13 is using Differential mode
    - 0 = AN13 is using Single-ended mode

- **bit 26**
  - **SIGN13**: AN13 Signed Data Mode bit
    - 1 = AN13 is using Signed Data mode
    - 0 = AN13 is using Unsigned Data mode

- **bit 25**
  - **DIFF12**: AN12 Mode bit
    - 1 = AN12 is using Differential mode
    - 0 = AN12 is using Single-ended mode

- **bit 24**
  - **SIGN12**: AN12 Signed Data Mode bit
    - 1 = AN12 is using Signed Data mode
    - 0 = AN12 is using Unsigned Data mode

- **bit 23**
  - **DIFF11**: AN11 Mode bit
    - 1 = AN11 is using Differential mode
    - 0 = AN11 is using Single-ended mode

- **bit 22**
  - **SIGN11**: AN11 Signed Data Mode bit
    - 1 = AN11 is using Signed Data mode
    - 0 = AN11 is using Unsigned Data mode

- **bit 21**
  - **DIFF10**: AN10 Mode bit
    - 1 = AN10 is using Differential mode
    - 0 = AN10 is using Single-ended mode
Register 22-5: ADCIMCON1: ADC Input Mode Control Register 1 (Continued)

bit 20  SIGN10: AN10 Signed Data Mode bit
1 = AN10 is using Signed Data mode
0 = AN10 is using Unsigned Data mode

bit 19  DIFF9: AN9 Mode bit
1 = AN9 is using Differential mode
0 = AN9 is using Single-ended mode

bit 18  SIGN9: AN9 Signed Data Mode bit
1 = AN9 is using Signed Data mode
0 = AN9 is using Unsigned Data mode

bit 17  DIFF8: AN 8 Mode bit
1 = AN8 is using Differential mode
0 = AN8 is using Single-ended mode

bit 16  SIGN8: AN8 Signed Data Mode bit
1 = AN8 is using Signed Data mode
0 = AN8 is using Unsigned Data mode

bit 15  DIFF7: AN7 Mode bit
1 = AN7 is using Differential mode
0 = AN7 is using Single-ended mode

bit 14  SIGN7: AN7 Signed Data Mode bit
1 = AN7 is using Signed Data mode
0 = AN7 is using Unsigned Data mode

bit 13  DIFF6: AN6 Mode bit
1 = AN6 is using Differential mode
0 = AN6 is using Single-ended mode

bit 12  SIGN6: AN6 Signed Data Mode bit
1 = AN6 is using Signed Data mode
0 = AN6 is using Unsigned Data mode

bit 11  DIFF5: AN5 Mode bit
1 = AN5 is using Differential mode
0 = AN5 is using Single-ended mode

bit 10  SIGN5: AN5 Signed Data Mode bit
1 = AN5 is using Signed Data mode
0 = AN5 is using Unsigned Data mode

bit 9   DIFF4: AN4 Mode bit
1 = AN4 is using Differential mode
0 = AN4 is using Single-ended mode

bit 8   SIGN4: AN4 Signed Data Mode bit
1 = AN4 is using Signed Data mode
0 = AN4 is using Unsigned Data mode

bit 7   DIFF3: AN3 Mode bit
1 = AN3 is using Differential mode
0 = AN3 is using Single-ended mode

bit 6   SIGN3: AN3 Signed Data Mode bit
1 = AN3 is using Signed Data mode
0 = AN3 is using Unsigned Data mode

bit 5   DIFF2: AN2 Mode bit
1 = AN2 is using Differential mode
0 = AN2 is using Single-ended mode
## Register 22-5: ADCIMCON1: ADC Input Mode Control Register 1 (Continued)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value 1</th>
<th>Value 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>SIGN2: AN2 Signed Data Mode bit</td>
<td>AN2 is using Signed Data mode</td>
<td>AN2 is using Unsigned Data mode</td>
</tr>
<tr>
<td>3</td>
<td>DIFF1: AN1 Mode bit</td>
<td>AN1 is using Differential mode</td>
<td>AN1 is using Single-ended mode</td>
</tr>
<tr>
<td>2</td>
<td>SIGN1: AN1 Signed Data Mode bit</td>
<td>AN1 is using Signed Data mode</td>
<td>AN1 is using Unsigned Data mode</td>
</tr>
<tr>
<td>1</td>
<td>DIFF0: AN0 Mode bit</td>
<td>AN0 is using Differential mode</td>
<td>AN0 is using Single-ended mode</td>
</tr>
<tr>
<td>0</td>
<td>SIGN0: AN0 Signed Data Mode bit</td>
<td>AN0 is using Signed Data mode</td>
<td>AN0 is using Unsigned Data mode</td>
</tr>
</tbody>
</table>
Register 22-6: ADCIMCON2: ADC Input Mode Control Register 2

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Bit 31/23/15/7</th>
<th>Bit 30/22/14/6</th>
<th>Bit 29/21/13/5</th>
<th>Bit 28/20/12/4</th>
<th>Bit 27/19/11/3</th>
<th>Bit 26/18/10/2</th>
<th>Bit 25/17/9/1</th>
<th>Bit 24/16/8/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td></td>
<td>DIFF31</td>
<td>SIGN31</td>
<td>DIFF30</td>
<td>SIGN30</td>
<td>DIFF29</td>
<td>SIGN29</td>
<td>DIFF28</td>
<td>SIGN28</td>
</tr>
<tr>
<td>23:16</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td></td>
<td>DIFF27</td>
<td>SIGN27</td>
<td>DIFF26</td>
<td>SIGN26</td>
<td>DIFF25</td>
<td>SIGN25</td>
<td>DIFF24</td>
<td>SIGN24</td>
</tr>
<tr>
<td>15:8</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td></td>
<td>DIFF23</td>
<td>SIGN23</td>
<td>DIFF22</td>
<td>SIGN22</td>
<td>DIFF21</td>
<td>SIGN21</td>
<td>DIFF20</td>
<td>SIGN20</td>
</tr>
<tr>
<td>7:0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td></td>
<td>DIFF19</td>
<td>SIGN19</td>
<td>DIFF18</td>
<td>SIGN18</td>
<td>DIFF17</td>
<td>SIGN17</td>
<td>DIFF16</td>
<td>SIGN16</td>
</tr>
</tbody>
</table>

Legend:
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as '0'
- -n = Value at POR
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- x = Bit is unknown

bit 31  DIFF31: AN31 Mode bit
        1 = AN31 is using Differential mode
        0 = AN31 is using Single-ended mode

bit 30  SIGN31: AN31 Signed Data Mode bit
        1 = AN31 is using Signed Data mode
        0 = AN31 is using Unsigned Data mode

bit 29  DIFF30: AN30 Mode bit
        1 = AN30 is using Differential mode
        0 = AN30 is using Single-ended mode

bit 28  SIGN30: AN30 Signed Data Mode bit
        1 = AN30 is using Signed Data mode
        0 = AN30 is using Unsigned Data mode

bit 27  DIFF29: AN29 Mode bit
        1 = AN29 is using Differential mode
        0 = AN29 is using Single-ended mode

bit 26  SIGN29: AN29 Signed Data Mode bit
        1 = AN29 is using Signed Data mode
        0 = AN29 is using Unsigned Data mode

bit 25  DIFF28: AN28 Mode bit
        1 = AN28 is using Differential mode
        0 = AN28 is using Single-ended mode

bit 24  SIGN28: AN28 Signed Data Mode bit
        1 = AN28 is using Signed Data mode
        0 = AN28 is using Unsigned Data mode

bit 23  DIFF27: AN27 Mode bit
        1 = AN27 is using Differential mode
        0 = AN27 is using Single-ended mode

bit 22  SIGN27: AN27 Signed Data Mode bit
        1 = AN27 is using Signed Data mode
        0 = AN27 is using Unsigned Data mode

bit 21  DIFF26: AN26 Mode bit
        1 = AN26 is using Differential mode
        0 = AN26 is using Single-ended mode
Section 22. 12-bit High-Speed SAR ADC

Register 22-6: ADCIMCON2: ADC Input Mode Control Register 2 (Continued)

bit 20  **SIGN26**: AN26 Signed Data Mode bit
  1 = AN26 is using Signed Data mode
  0 = AN26 is using Unsigned Data mode

bit 19  **DIFF25**: AN25 Mode bit
  1 = AN25 is using Differential mode
  0 = AN25 is using Single-ended mode

bit 18  **SIGN25**: AN25 Signed Data Mode bit
  1 = AN25 is using Signed Data mode
  0 = AN25 is using Unsigned Data mode

bit 17  **DIFF24**: AN24 Mode bit
  1 = AN24 is using Differential mode
  0 = AN24 is using Single-ended mode

bit 16  **SIGN24**: AN24 Signed Data Mode bit
  1 = AN24 is using Signed Data mode
  0 = AN24 is using Unsigned Data mode

bit 15  **DIFF23**: AN23 Mode bit
  1 = AN23 is using Differential mode
  0 = AN23 is using Single-ended mode

bit 14  **SIGN23**: AN23 Signed Data Mode bit
  1 = AN23 is using Signed Data mode
  0 = AN23 is using Unsigned Data mode

bit 13  **DIFF22**: AN22 Mode bit
  1 = AN22 is using Differential mode
  0 = AN22 is using Single-ended mode

bit 12  **SIGN22**: AN22 Signed Data Mode bit
  1 = AN22 is using Signed Data mode
  0 = AN22 is using Unsigned Data mode

bit 11  **DIFF21**: AN21 Mode bit
  1 = AN21 is using Differential mode
  0 = AN21 is using Single-ended mode

bit 10  **SIGN21**: AN21 Signed Data Mode bit
  1 = AN21 is using Signed Data mode
  0 = AN21 is using Unsigned Data mode

bit 9   **DIFF20**: AN20 Mode bit
  1 = AN20 is using Differential mode
  0 = AN20 is using Single-ended mode

bit 8   **SIGN20**: AN20 Signed Data Mode bit
  1 = AN20 is using Signed Data mode
  0 = AN20 is using Unsigned Data mode

bit 7   **DIFF19**: AN19 Mode bit
  1 = AN19 is using Differential mode
  0 = AN19 is using Single-ended mode

bit 6   **SIGN19**: AN19 Signed Data Mode bit
  1 = AN19 is using Signed Data mode
  0 = AN19 is using Unsigned Data mode

bit 5   **DIFF18**: AN18 Mode bit
  1 = AN18 is using Differential mode
  0 = AN18 is using Single-ended mode
<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>SIGN18: AN18 Signed Data Mode bit</td>
<td>1</td>
<td>AN18 is using Signed Data mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>AN18 is using Unsigned Data mode</td>
</tr>
<tr>
<td>6</td>
<td>DIFF17: AN17 Mode bit</td>
<td>1</td>
<td>AN17 is using Differential mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>AN17 is using Single-ended mode</td>
</tr>
<tr>
<td>5</td>
<td>SIGN17: AN17 Signed Data Mode bit</td>
<td>1</td>
<td>AN17 is using Signed Data mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>AN17 is using Unsigned Data mode</td>
</tr>
<tr>
<td>4</td>
<td>DIFF16: AN16 Mode bit</td>
<td>1</td>
<td>AN16 is using Differential mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>AN16 is using Single-ended mode</td>
</tr>
<tr>
<td>3</td>
<td>SIGN16: AN16 Signed Data Mode bit</td>
<td>1</td>
<td>AN16 is using Signed Data mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>AN16 is using Unsigned Data mode</td>
</tr>
</tbody>
</table>
### Section 22. 12-bit High-Speed SAR ADC

**Register 22-7: ADCIMCON3: ADC Input Mode Control Register 3**

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Bit</th>
<th>Bit</th>
<th>Bit</th>
<th>Bit</th>
<th>Bit</th>
<th>Bit</th>
<th>Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td></td>
<td>DIFF47</td>
<td>SGN47</td>
<td>DIFF46</td>
<td>SGN46</td>
<td>DIFF45</td>
<td>SGN45</td>
<td>DIFF44</td>
</tr>
<tr>
<td>23:16</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td></td>
<td>DIFF43</td>
<td>SGN43</td>
<td>DIFF42</td>
<td>SGN42</td>
<td>DIFF41</td>
<td>SGN41</td>
<td>DIFF40</td>
</tr>
<tr>
<td>15:8</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td></td>
<td>DIFF39</td>
<td>SGN39</td>
<td>DIFF38</td>
<td>SGN38</td>
<td>DIFF37</td>
<td>SGN37</td>
<td>DIFF36</td>
</tr>
<tr>
<td>7:0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td></td>
<td>DIFF35</td>
<td>SGN35</td>
<td>DIFF34</td>
<td>SGN34</td>
<td>DIFF33</td>
<td>SGN33</td>
<td>DIFF32</td>
</tr>
</tbody>
</table>

**Legend:**
- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as ‘0’
- **-n** = Value at POR
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- **x** = Bit is unknown

**Legend:**
- **DIFF47**: AN47 Mode bit
  - 1 = AN47 is using Differential mode
  - 0 = AN47 is using Single-ended mode

**Legend:**
- **SIGN47**: AN47 Signed Data Mode bit
  - 1 = AN47 is using Signed Data mode
  - 0 = AN47 is using Unsigned Data mode

**Legend:**
- **DIFF46**: AN46 Mode bit
  - 1 = AN46 is using Differential mode
  - 0 = AN46 is using Single-ended mode

**Legend:**
- **SIGN46**: AN46 Signed Data Mode bit
  - 1 = AN46 is using Signed Data mode
  - 0 = AN46 is using Unsigned Data mode

**Legend:**
- **DIFF45**: AN45 Mode bit
  - 1 = AN45 is using Differential mode
  - 0 = AN45 is using Single-ended mode

**Legend:**
- **SIGN45**: AN45 Signed Data Mode bit
  - 1 = AN45 is using Signed Data mode
  - 0 = AN45 is using Unsigned Data mode

**Legend:**
- **DIFF44**: AN44 Mode bit
  - 1 = AN44 is using Differential mode
  - 0 = AN44 is using Single-ended mode

**Legend:**
- **SIGN44**: AN44 Signed Data Mode bit
  - 1 = AN44 is using Signed Data mode
  - 0 = AN44 is using Unsigned Data mode

**Legend:**
- **DIFF43**: AN43 Mode bit
  - 1 = AN43 is using Differential mode
  - 0 = AN43 is using Single-ended mode

**Legend:**
- **SIGN43**: AN43 Signed Data Mode bit
  - 1 = AN43 is using Signed Data mode
  - 0 = AN43 is using Unsigned Data mode

**Legend:**
- **DIFF42**: AN42 Mode bit
  - 1 = AN42 is using Differential mode
  - 0 = AN42 is using Single-ended mode
Register 22-7: ADCIMCON3: ADC Input Mode Control Register 3 (Continued)

bit 20  SIGN42: AN42 Signed Data Mode bit
1 = AN42 is using Signed Data mode
0 = AN42 is using Unsigned Data mode

bit 19  DIFF41: AN41 Mode bit
1 = AN41 is using Differential mode
0 = AN41 is using Single-ended mode

bit 18  SIGN41: AN41 Signed Data Mode bit
1 = AN41 is using Signed Data mode
0 = AN41 is using Unsigned Data mode

bit 17  DIFF40: AN40 Mode bit
1 = AN40 is using Differential mode
0 = AN40 is using Single-ended mode

bit 16  SIGN40: AN40 Signed Data Mode bit
1 = AN40 is using Signed Data mode
0 = AN40 is using Unsigned Data mode

bit 15  DIFF39: AN39 Mode bit
1 = AN39 is using Differential mode
0 = AN39 is using Single-ended mode

bit 14  SIGN39: AN39 Signed Data Mode bit
1 = AN39 is using Signed Data mode
0 = AN39 is using Unsigned Data mode

bit 13  DIFF38: AN38 Mode bit
1 = AN38 is using Differential mode
0 = AN38 is using Single-ended mode

bit 12  SIGN38: AN38 Signed Data Mode bit
1 = AN38 is using Signed Data mode
0 = AN38 is using Unsigned Data mode

bit 11  DIFF37: AN37 Mode bit
1 = AN37 is using Differential mode
0 = AN37 is using Single-ended mode

bit 10  SIGN37: AN37 Signed Data Mode bit
1 = AN37 is using Signed Data mode
0 = AN37 is using Unsigned Data mode

bit 9  DIFF36: AN36 Mode bit
1 = AN36 is using Differential mode
0 = AN36 is using Single-ended mode

bit 8  SIGN36: AN36 Signed Data Mode bit
1 = AN36 is using Signed Data mode
0 = AN36 is using Unsigned Data mode

bit 7  DIFF35: AN35 Mode bit
1 = AN35 is using Differential mode
0 = AN35 is using Single-ended mode

bit 6  SIGN35: AN35 Signed Data Mode bit
1 = AN35 is using Signed Data mode
0 = AN35 is using Unsigned Data mode

bit 5  DIFF34: AN34 Mode bit
1 = AN34 is using Differential mode
0 = AN34 is using Single-ended mode
Register 22-7: ADCIMCON3: ADC Input Mode Control Register 3 (Continued)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>SIGN34: AN34 Signed Data Mode bit</td>
<td>1</td>
<td>AN34 is using Signed Data mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>AN34 is using Unsigned Data mode</td>
</tr>
<tr>
<td>3</td>
<td>DIFF33: AN33 Mode bit</td>
<td>1</td>
<td>AN33 is using Differential mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>AN33 is using Single-ended mode</td>
</tr>
<tr>
<td>2</td>
<td>SIGN33: AN33 Signed Data Mode bit</td>
<td>1</td>
<td>AN33 is using Signed Data mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>AN33 is using Unsigned Data mode</td>
</tr>
<tr>
<td>1</td>
<td>DIFF32: AN32 Mode bit</td>
<td>1</td>
<td>AN32 is using Differential mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>AN32 is using Single-ended mode</td>
</tr>
<tr>
<td>0</td>
<td>SIGN32: AN32 Signed Data Mode bit</td>
<td>1</td>
<td>AN32 is using Signed Data mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>AN32 is using Unsigned Data mode</td>
</tr>
</tbody>
</table>
## PIC32 Family Reference Manual

### Register 22-8: ADCIMCON4: ADC Input Mode Control Register 4

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Bit 31:24</th>
<th>Bit 23:16</th>
<th>Bit 15:8</th>
<th>Bit 7:0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td></td>
<td>DIFF63</td>
<td>SIGN63</td>
<td>DIFF62</td>
<td>SIGN62</td>
</tr>
<tr>
<td></td>
<td>31/24</td>
<td>30/22/14/6</td>
<td>29/21/13/5</td>
<td>28/20/12/4</td>
</tr>
<tr>
<td></td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td></td>
<td>DIFF59</td>
<td>SIGN59</td>
<td>DIFF58</td>
<td>SIGN58</td>
</tr>
<tr>
<td></td>
<td>23:16</td>
<td>22/14/6</td>
<td>21/13/5</td>
<td>20/12/4</td>
</tr>
<tr>
<td></td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td></td>
<td>DIFF55</td>
<td>SIGN55</td>
<td>DIFF54</td>
<td>SIGN54</td>
</tr>
<tr>
<td></td>
<td>15:8</td>
<td>14/6</td>
<td>13/5</td>
<td>12/4</td>
</tr>
<tr>
<td></td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td></td>
<td>DIFF51</td>
<td>SIGN51</td>
<td>DIFF50</td>
<td>SIGN50</td>
</tr>
<tr>
<td></td>
<td>7:0</td>
<td>6/0</td>
<td>5/0</td>
<td>4/0</td>
</tr>
<tr>
<td></td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
</tbody>
</table>

**Legend:**

- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as ‘0’
- -n = Value at POR
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- x = Bit is unknown

**bit 31**

**DIFF63**: AN63 Mode bit
- 1 = AN63 is using Differential mode
- 0 = AN63 is using Single-ended mode

**bit 30**

**SIGN63**: AN63 Signed Data Mode bit
- 1 = AN63 is using Signed Data mode
- 0 = AN63 is using Unsigned Data mode

**bit 29**

**DIFF62**: AN62 Mode bit
- 1 = AN62 is using Differential mode
- 0 = AN62 is using Single-ended mode

**bit 28**

**SIGN62**: AN62 Signed Data Mode bit
- 1 = AN62 is using Signed Data mode
- 0 = AN62 is using Unsigned Data mode

**bit 27**

**DIFF61**: AN61 Mode bit
- 1 = AN61 is using Differential mode
- 0 = AN61 is using Single-ended mode

**bit 26**

**SIGN61**: AN61 Signed Data Mode bit
- 1 = AN61 is using Signed Data mode
- 0 = AN61 is using Unsigned Data mode

**bit 25**

**DIFF60**: AN60 Mode bit
- 1 = AN60 is using Differential mode
- 0 = AN60 is using Single-ended mode

**bit 24**

**SIGN60**: AN60 Signed Data Mode bit
- 1 = AN60 is using Signed Data mode
- 0 = AN60 is using Unsigned Data mode

**bit 23**

**DIFF59**: AN59 Mode bit
- 1 = AN59 is using Differential mode
- 0 = AN59 is using Single-ended mode

**bit 22**

**SIGN59**: AN59 Signed Data Mode bit
- 1 = AN59 is using Signed Data mode
- 0 = AN59 is using Unsigned Data mode

**bit 21**

**DIFF58**: AN58 Mode bit
- 1 = AN58 is using Differential mode
- 0 = AN58 is using Single-ended mode
### Register 22-8: ADCIMCON4: ADC Input Mode Control Register 4 (Continued)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
<th>Mode</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>SIGN58: AN58 Signed Data Mode bit</td>
<td></td>
<td>1 = AN58 is using Signed Data mode</td>
<td>0 = AN58 is using Unsigned Data mode</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>DIFF57: AN57 Mode bit</td>
<td></td>
<td>1 = AN57 is using Differential mode</td>
<td>0 = AN57 is using Single-ended mode</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>SIGN57: AN57 Signed Data Mode bit</td>
<td></td>
<td>1 = AN57 is using Signed Data mode</td>
<td>0 = AN57 is using Unsigned Data mode</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>DIFF56: AN56 Mode bit</td>
<td></td>
<td>1 = AN56 is using Differential mode</td>
<td>0 = AN56 is using Single-ended mode</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>SIGN56: AN56 Signed Data Mode bit</td>
<td></td>
<td>1 = AN56 is using Signed Data mode</td>
<td>0 = AN56 is using Unsigned Data mode</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>DIFF55: AN55 Mode bit</td>
<td></td>
<td>1 = AN55 is using Differential mode</td>
<td>0 = AN55 is using Single-ended mode</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>SIGN55: AN55 Signed Data Mode bit</td>
<td></td>
<td>1 = AN55 is using Signed Data mode</td>
<td>0 = AN55 is using Unsigned Data mode</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>DIFF54: AN54 Mode bit</td>
<td></td>
<td>1 = AN54 is using Differential mode</td>
<td>0 = AN54 is using Single-ended mode</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>SIGN54: AN54 Signed Data Mode bit</td>
<td></td>
<td>1 = AN54 is using Signed Data mode</td>
<td>0 = AN54 is using Unsigned Data mode</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>DIFF53: AN53 Mode bit</td>
<td></td>
<td>1 = AN53 is using Differential mode</td>
<td>0 = AN53 is using Single-ended mode</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>SIGN53: AN53 Signed Data Mode bit</td>
<td></td>
<td>1 = AN53 is using Signed Data mode</td>
<td>0 = AN53 is using Unsigned Data mode</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>DIFF52: AN52 Mode bit</td>
<td></td>
<td>1 = AN52 is using Differential mode</td>
<td>0 = AN52 is using Single-ended mode</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>SIGN52: AN52 Signed Data Mode bit</td>
<td></td>
<td>1 = AN52 is using Signed Data mode</td>
<td>0 = AN52 is using Unsigned Data mode</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>DIFF51: AN51 Mode bit</td>
<td></td>
<td>1 = AN51 is using Differential mode</td>
<td>0 = AN51 is using Single-ended mode</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>SIGN51: AN51 Signed Data Mode bit</td>
<td></td>
<td>1 = AN51 is using Signed Data mode</td>
<td>0 = AN51 is using Unsigned Data mode</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>DIFF50: AN50 Mode bit</td>
<td></td>
<td>1 = AN50 is using Differential mode</td>
<td>0 = AN50 is using Single-ended mode</td>
<td></td>
</tr>
</tbody>
</table>
Register 22-8: ADCIMCON4: ADC Input Mode Control Register 4 (Continued)

bit 4 \textbf{SIGN50}: AN50 Signed Data Mode bit
- 1 = AN50 is using Signed Data mode
- 0 = AN50 is using Unsigned Data mode

bit 3 \textbf{DIFF49}: AN49 Mode bit
- 1 = AN49 is using Differential mode
- 0 = AN49 is using Single-ended mode

bit 2 \textbf{SIGN49}: AN49 Signed Data Mode bit
- 1 = AN49 is using Signed Data mode
- 0 = AN49 is using Unsigned Data mode

bit 1 \textbf{DIFF48}: AN48 Mode bit
- 1 = AN48 is using Differential mode
- 0 = AN48 is using Single-ended mode

bit 0 \textbf{SIGN48}: AN48 Signed Data Mode bit
- 1 = AN48 is using Signed Data mode
- 0 = AN48 is using Unsigned Data mode
### Section 22. 12-bit High-Speed SAR ADC

#### Register 22-9: ADCGIRQEN1: ADC Global Interrupt Enable Register 1

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Bit</th>
<th>Bit</th>
<th>Bit</th>
<th>Bit</th>
<th>Bit</th>
<th>Bit</th>
<th>Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>AGIEN31</td>
<td>AGIEN30</td>
<td>AGIEN29</td>
<td>AGIEN28</td>
<td>AGIEN27</td>
<td>AGIEN26</td>
<td>AGIEN25</td>
<td>AGIEN24</td>
</tr>
<tr>
<td>23:16</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>AGIEN23</td>
<td>AGIEN22</td>
<td>AGIEN21</td>
<td>AGIEN20</td>
<td>AGIEN19</td>
<td>AGIEN18</td>
<td>AGIEN17</td>
<td>AGIEN16</td>
</tr>
<tr>
<td>15:8</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>AGIEN15</td>
<td>AGIEN14</td>
<td>AGIEN13</td>
<td>AGIEN12</td>
<td>AGIEN11</td>
<td>AGIEN10</td>
<td>AGIEN9</td>
<td>AGIEN8</td>
</tr>
<tr>
<td>7:0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>AGIEN7</td>
<td>AGIEN6</td>
<td>AGIEN5</td>
<td>AGIEN4</td>
<td>AGIEN3</td>
<td>AGIEN2</td>
<td>AGIEN1</td>
<td>AGIEN0</td>
</tr>
</tbody>
</table>

**Legend:**
- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as ‘0’
- **-n** = Value at POR
- **‘1’** = Bit is set
- **‘0’** = Bit is cleared
- **x** = Bit is unknown

**bit 31-0 AGIEN31:AGIEN0: ADC Interrupt Enable bits**
- 1 = Interrupts are enabled for the selected analog input. The interrupt is generated after the converted data is ready (indicated by the ARDYx bit (‘x’ = 31-0) of the ADCDSTAT1 register)
- 0 = Interrupts are disabled

#### Register 22-10: ADCGIRQEN2: ADC Global Interrupt Enable Register 2

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Bit</th>
<th>Bit</th>
<th>Bit</th>
<th>Bit</th>
<th>Bit</th>
<th>Bit</th>
<th>Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>AGIEN63</td>
<td>AGIEN62</td>
<td>AGIEN61</td>
<td>AGIEN60</td>
<td>AGIEN59</td>
<td>AGIEN58</td>
<td>AGIEN57</td>
<td>AGIEN56</td>
</tr>
<tr>
<td>23:16</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>AGIEN55</td>
<td>AGIEN54</td>
<td>AGIEN53</td>
<td>AGIEN52</td>
<td>AGIEN51</td>
<td>AGIEN50</td>
<td>AGIEN49</td>
<td>AGIEN48</td>
</tr>
<tr>
<td>15:8</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>AGIEN47</td>
<td>AGIEN46</td>
<td>AGIEN45</td>
<td>AGIEN44</td>
<td>AGIEN43</td>
<td>AGIEN42</td>
<td>AGIEN41</td>
<td>AGIEN40</td>
</tr>
<tr>
<td>7:0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>AGIEN39</td>
<td>AGIEN38</td>
<td>AGIEN37</td>
<td>AGIEN36</td>
<td>AGIEN35</td>
<td>AGIEN34</td>
<td>AGIEN33</td>
<td>AGIEN32</td>
</tr>
</tbody>
</table>

**Legend:**
- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as ‘0’
- **-n** = Value at POR
- **‘1’** = Bit is set
- **‘0’** = Bit is cleared
- **x** = Bit is unknown

**bit 31-0 AGIEN63:AGIEN32 ADC Interrupt Enable bits**
- 1 = Interrupts are enabled for the selected analog input. The interrupt is generated after the converted data is ready (indicated by the ARDYx bit (‘x’ = 63-32) of the ADCDSTAT2 register)
- 0 = Interrupts are disabled
### Register 22-11: ADCCSS1: ADC Common Scan Select Register 1

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Bit 31/23/15/7</th>
<th>Bit 29/21/13/5</th>
<th>Bit 27/19/11/3</th>
<th>Bit 25/17/9/1</th>
<th>Bit 24/16/8/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td></td>
<td>CSS31</td>
<td>CSS30</td>
<td>CSS29</td>
<td>CSS28</td>
<td>CSS27</td>
</tr>
<tr>
<td>23:16</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td></td>
<td>CSS23</td>
<td>CSS22</td>
<td>CSS21</td>
<td>CSS20</td>
<td>CSS19</td>
</tr>
<tr>
<td>15:8</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td></td>
<td>CSS15</td>
<td>CSS14</td>
<td>CSS13</td>
<td>CSS12</td>
<td>CSS11</td>
</tr>
<tr>
<td>7:0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td></td>
<td>CSS7</td>
<td>CSS6</td>
<td>CSS5</td>
<td>CSS4</td>
<td>CSS3</td>
</tr>
</tbody>
</table>

**Legend:**

- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as ‘0’
- **-n** = Value at POR
- **‘1’** = Bit is set
- **‘0’** = Bit is cleared
- **x** = Bit is unknown

**bit 31-0**

- **CSS31:** Analog Common Scan Select bits
  - 1 = Select ANx for input scan
  - 0 = Skip ANx for input scan

**Note 1:**
In addition to setting the appropriate bits in this register, Class 1 and Class 2 analog inputs must select the STRIG input as the trigger source if they are to be scanned through the CSSx bits. Refer to the bit descriptions in the ADCTRGx register (Register 22-18) for selecting the STRIG option.

**Note 2:**
If a Class 1 or Class 2 input is included in the scan by setting the CSSx bit to ‘1’ and by setting the TRGSRCx<4:0> bits to STRIG mode (‘0b11), the user application must ensure that no other triggers are generated for that input using the RQCNVRT bit in the ADCCON3 register or the hardware input or any digital filter. Otherwise, the scan behavior is unpredictable.

### Register 22-12: ADCCSS2: ADC Common Scan Select Register 2

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Bit 31/23/15/7</th>
<th>Bit 29/21/13/5</th>
<th>Bit 27/19/11/3</th>
<th>Bit 25/17/9/1</th>
<th>Bit 24/16/8/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td></td>
<td>CSS63</td>
<td>CSS62</td>
<td>CSS61</td>
<td>CSS60</td>
<td>CSS59</td>
</tr>
<tr>
<td>23:16</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td></td>
<td>CSS55</td>
<td>CSS54</td>
<td>CSS53</td>
<td>CSS52</td>
<td>CSS51</td>
</tr>
<tr>
<td>15:8</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td></td>
<td>CSS47</td>
<td>CSS46</td>
<td>CSS45</td>
<td>CSS44</td>
<td>CSS43</td>
</tr>
<tr>
<td>7:0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td></td>
<td>CSS39</td>
<td>CSS38</td>
<td>CSS37</td>
<td>CSS36</td>
<td>CSS35</td>
</tr>
</tbody>
</table>

**Legend:**

- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as ‘0’
- **-n** = Value at POR
- **‘1’** = Bit is set
- **‘0’** = Bit is cleared
- **x** = Bit is unknown

**bit 31-0**

- **CSS63:** Analog Common Scan Select bits
  - 1 = Select ANx for input scan
  - 0 = Skip ANx for input scan

**Note:**
Analog inputs 63 to 32 are always Class 3, as there are only 32 triggers available.
### Section 22. 12-bit High-Speed SAR ADC

#### Register 22-13: ADCDSTAT1: ADC Data Ready Status Register 1

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Bit 31/23/15/7</th>
<th>Bit 30/22/14/6</th>
<th>Bit 29/21/13/5</th>
<th>Bit 28/20/12/4</th>
<th>Bit 27/19/11/3</th>
<th>Bit 26/18/10/2</th>
<th>Bit 25/17/9/1</th>
<th>Bit 24/16/8/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
</tr>
<tr>
<td></td>
<td>ARDY31</td>
<td>ARDY30</td>
<td>ARDY29</td>
<td>ARDY28</td>
<td>ARDY27</td>
<td>ARDY26</td>
<td>ARDY25</td>
<td>ARDY24</td>
</tr>
<tr>
<td>23:16</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
</tr>
<tr>
<td></td>
<td>ARDY23</td>
<td>ARDY22</td>
<td>ARDY21</td>
<td>ARDY20</td>
<td>ARDY19</td>
<td>ARDY18</td>
<td>ARDY17</td>
<td>ARDY16</td>
</tr>
<tr>
<td>15:8</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
</tr>
<tr>
<td></td>
<td>ARDY15</td>
<td>ARDY14</td>
<td>ARDY13</td>
<td>ARDY12</td>
<td>ARDY11</td>
<td>ARDY10</td>
<td>ARDY9</td>
<td>ARDY8</td>
</tr>
<tr>
<td>7:0</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
</tr>
<tr>
<td></td>
<td>ARDY7</td>
<td>ARDY6</td>
<td>ARDY5</td>
<td>ARDY4</td>
<td>ARDY3</td>
<td>ARDY2</td>
<td>ARDY1</td>
<td>ARDY0</td>
</tr>
</tbody>
</table>

**Legend:**

- **HS** = Hardware Set
- **HC** = Hardware Cleared
- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as ‘0’
- **-n** = Value at POR
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- **x** = Bit is unknown

**bit 31-0 ARDY31:ARDY0:** Conversion Data Ready for Corresponding Analog Input Ready bits

- 1 = This bit is set when converted data is ready in the data register
- 0 = This bit is cleared when the associated data register is read

#### Register 22-14: ADCDSTAT2: ADC Data Ready Status Register 2

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Bit 31/23/15/7</th>
<th>Bit 30/22/14/6</th>
<th>Bit 29/21/13/5</th>
<th>Bit 28/20/12/4</th>
<th>Bit 27/19/11/3</th>
<th>Bit 26/18/10/2</th>
<th>Bit 25/17/9/1</th>
<th>Bit 24/16/8/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
</tr>
<tr>
<td></td>
<td>ARDY63</td>
<td>ARDY62</td>
<td>ARDY61</td>
<td>ARDY60</td>
<td>ARDY59</td>
<td>ARDY58</td>
<td>ARDY57</td>
<td>ARDY56</td>
</tr>
<tr>
<td>23:16</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
</tr>
<tr>
<td></td>
<td>ARDY55</td>
<td>ARDY54</td>
<td>ARDY53</td>
<td>ARDY52</td>
<td>ARDY51</td>
<td>ARDY50</td>
<td>ARDY49</td>
<td>ARDY48</td>
</tr>
<tr>
<td>15:8</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
</tr>
<tr>
<td></td>
<td>ARDY47</td>
<td>ARDY46</td>
<td>ARDY45</td>
<td>ARDY44</td>
<td>ARDY43</td>
<td>ARDY42</td>
<td>ARDY41</td>
<td>ARDY40</td>
</tr>
<tr>
<td>7:0</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
</tr>
<tr>
<td></td>
<td>ARDY39</td>
<td>ARDY38</td>
<td>ARDY37</td>
<td>ARDY36</td>
<td>ARDY35</td>
<td>ARDY34</td>
<td>ARDY33</td>
<td>ARDY32</td>
</tr>
</tbody>
</table>

**Legend:**

- **HS** = Hardware Set
- **HC** = Hardware Cleared
- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as ‘0’
- **-n** = Value at POR
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- **x** = Bit is unknown

**bit 31-0 ARDY63:ARDY32:** Conversion Data Ready for Corresponding Analog Input Ready bits

- 1 = This bit is set when converted data is ready in the data register
- 0 = This bit is cleared when the associated data register is read
Register 22-15: ADCMPENx: ADC Digital Comparator ‘x’ Enable Register (‘x’ = 1 through 6)

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Bit 31/23/15/7</th>
<th>Bit 30/22/14/6</th>
<th>Bit 29/21/13/5</th>
<th>Bit 28/20/12/4</th>
<th>Bit 27/19/11/3</th>
<th>Bit 26/18/10/2</th>
<th>Bit 25/17/9/1</th>
<th>Bit 24/16/8/0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>31:24</td>
<td>CMPE31</td>
<td>CMPE30</td>
<td>CMPE29</td>
<td>CMPE28</td>
<td>CMPE27</td>
<td>CMPE26</td>
<td>CMPE25</td>
<td>CMPE24</td>
</tr>
<tr>
<td>23:16</td>
<td>CMPE23</td>
<td>CMPE22</td>
<td>CMPE21</td>
<td>CMPE20</td>
<td>CMPE19</td>
<td>CMPE18</td>
<td>CMPE17</td>
<td>CMPE16</td>
</tr>
<tr>
<td>15:8</td>
<td>CMPE15</td>
<td>CMPE14</td>
<td>CMPE13</td>
<td>CMPE12</td>
<td>CMPE11</td>
<td>CMPE10</td>
<td>CMPE9</td>
<td>CMPE8</td>
</tr>
<tr>
<td>7:0</td>
<td>CMPE7</td>
<td>CMPE6</td>
<td>CMPE5</td>
<td>CMPE4</td>
<td>CMPE3</td>
<td>CMPE2</td>
<td>CMPE1</td>
<td>CMPE0</td>
</tr>
</tbody>
</table>

Legend:

R = Readable bit  W = Writable bit  U = Unimplemented bit, read as ‘0’
-n = Value at POR  ‘1’ = Bit is set  ‘0’ = Bit is cleared  x = Bit is unknown

Bit 31-0  
CMPE31:CMPE0: ADC Digital Comparator ‘x’ Enable bits

These bits enable conversion results corresponding to the Analog Input to be processed by the Digital Comparator. CMPE0 enables AN0, CMPE1 enables AN1, and so on.

**Note 1:** CMPEx = ANx, where ‘x’ = 0-31 (Digital Comparator inputs are limited to AN0 through AN31).

**Note 2:** Changing the bits in this register while the Digital Comparator is enabled (ENDCMP = 1) can result in unpredictable behavior.

Register 22-16: ADCMPx: ADC Digital Comparator ‘x’ Limit Value Register (‘x’ = 1 through 6)

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Bit 31/23/15/7</th>
<th>Bit 30/22/14/6</th>
<th>Bit 29/21/13/5</th>
<th>Bit 28/20/12/4</th>
<th>Bit 27/19/11/3</th>
<th>Bit 26/18/10/2</th>
<th>Bit 25/17/9/1</th>
<th>Bit 24/16/8/0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>31:24</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td></td>
<td>DCMPHI&lt;15:8&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23:16</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td></td>
<td>DCMPHI&lt;7:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:8</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td></td>
<td>DCMPLO&lt;15:8&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td></td>
<td>DCMPLO&lt;7:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend:

R = Readable bit  W = Writable bit  U = Unimplemented bit, read as ‘0’
-n = Value at POR  ‘1’ = Bit is set  ‘0’ = Bit is cleared  x = Bit is unknown

Bit 31-16  
DCMPHI<15:0>: Digital Comparator ‘x’ High Limit Value bits<sup>(1,2,3)</sup>

These bits store the high limit value, which is used by digital comparator for comparisons with ADC converted data.

Bit 15-0  
DCMPLO<15:0>: Digital Comparator ‘x’ Low Limit Value bits<sup>(1,2,3)</sup>

These bits store the low limit value, which is used by digital comparator for comparisons with ADC converted data.

**Note 1:** Changing these bits while the Digital Comparator is enabled (ENDCMP = 1) can result in unpredictable behavior.

**Note 2:** The format of the limit values should match the format of the ADC converted value in terms of sign and fractional settings.

**Note 3:** For Digital Comparator 1 used in CVD mode, the DCMPHI<15:0> and DCMPLO<15:0> bits must always be specified in signed format, as the CVD output data is differential and is always signed.
Section 22. 12-bit High-Speed SAR ADC

Register 22-17: ADCFLTRx: ADC Digital Filter ‘x’ Register (‘x’ = 1 through 6)

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Bit 31/23/15/7</th>
<th>Bit 30/22/14/6</th>
<th>Bit 29/21/13/5</th>
<th>Bit 28/20/12/4</th>
<th>Bit 27/19/11/3</th>
<th>Bit 26/18/10/2</th>
<th>Bit 25/17/9/1</th>
<th>Bit 24/16/8/0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>31:24</td>
<td>AFEN</td>
<td>DATA16EN</td>
<td>DFMODE</td>
<td>OVRSAM&lt;2:0&gt;</td>
<td>AFGIEN</td>
<td>AFRDY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23:16</td>
<td>U-0</td>
<td>U-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:8</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
</tr>
<tr>
<td></td>
<td>FLTRDATA&lt;15:8&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
</tr>
<tr>
<td></td>
<td>FLTRDATA&lt;7:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend:
- HS = Hardware Set  
- HC = Hardware Cleared  
- R = Readable bit  
- W = Writable bit  
- U = Unimplemented bit, read as ‘0’  
- -n = Value at POR  
- ‘1’ = Bit is set  
- ‘0’ = Bit is cleared  
- x = Bit is unknown

bit 31  **AFEN**: Digital Filter ‘x’ Enable bit  
1 = Digital filter is enabled  
0 = Digital filter is disabled and the AFRDY status bit is cleared

bit 30  **DATA16EN**: Filter Significant Data Length bit  
1 = All 16 bits of the filter output data are significant  
0 = Only the first 12 bits are significant, followed by four zeros

Note: This bit is significant only if DFMODE = 1 (Averaging Mode) and FRACT (ADCCON1<23>) = 1 (Fractional Output Mode).

bit 29  **DFMODE**: ADC Filter Mode bit  
1 = Filter ‘x’ works in Averaging mode  
0 = Filter ‘x’ works in Oversampling Filter mode (default)

When the ADC filter is enabled and DFMODE = 0:

Once an ADC edge conversion trigger event occurs, it is held active until the filter OVRSAM sample count has expired.

The Minimum Trigger Period = (OVRSAM<2:0> bits in ADCFLTRx) (SAMC<7:0> bits in ADCxTIME TAD + ((SELRES<1:0> bits in ADCxTIME + 1) TAD).

Example:
- OVRSAM<2:0> bits in ADCFLTRx = 8x Samples  
- SAMC<7:0> bits in ADCxTIME = 3 TAD  
- SELRES<1:0> bits in ADCxTIME = 12 bits

User Min Trig period ≥ (8 * (3 TAD + 13 TAD)) = 128 TAD

When the ADC filter is enabled and DFMODE = 1:

All ADC conversions are initiated solely by trigger events. After OVRSAM normal edge trigger events and subsequent conversions, the ADC filter result will contain the average value of OVRSAM number of conversions.

Refer to Figure 22-18: “ADC Filter Comparisons Example” for more information.
PIC32 Family Reference Manual

Register 22-17: ADCFLTRx: ADC Digital Filter ‘x’ Register (‘x’ = 1 through 6) (Continued)

bit 28-26 OVRSAM<2:0>: Oversampling Filter Ratio bits
If DFMODE is ‘0’:
   111 = 128 samples (shift sum 3 bits to right, output data is in 15.1 format)
   110 = 32 samples (shift sum 2 bits to right, output data is in 14.1 format)
   101 = 8 samples (shift sum 1 bit to right, output data is in 13.1 format)
   100 = 2 samples (shift sum 0 bits to right, output data is in 12.1 format)
   011 = 256 samples (shift sum 4 bits to right, output data is 16 bits)
   010 = 64 samples (shift sum 3 bits to right, output data is 15 bits)
   001 = 16 samples (shift sum 2 bits to right, output data is 14 bits)
   000 = 4 samples (shift sum 1 bit to right, output data is 13 bits)

If DFMODE is ‘1’:
   111 = 256 samples (256 samples to be averaged)
   110 = 128 samples (128 samples to be averaged)
   101 = 64 samples (64 samples to be averaged)
   100 = 32 samples (32 samples to be averaged)
   011 = 16 samples (16 samples to be averaged)
   010 = 8 samples (8 samples to be averaged)
   001 = 4 samples (4 samples to be averaged)
   000 = 2 samples (2 samples to be averaged)

bit 25 AFGIEN: Digital Filter ‘x’ Interrupt Enable bit
1 = Digital filter interrupt is enabled and is generated by the AFRDY status bit
0 = Digital filter is disabled

bit 24 AFRDY: Digital Filter ‘x’ Data Ready Status bit
1 = Data is ready in the FLTRDATA<15:0> bits
0 = Data is not ready

Note: This bit is cleared by reading the FLTRDATA<15:0> bits or by disabling the Digital Filter module (by setting AFEN to ‘0’).

bit 23-21 Unimplemented: Read as ‘0’

bit 20-16 CHNLID<4:0>: Digital Filter Analog Input Selection bits
These bits specify the analog input to be used as the oversampling filter data source.
   11111 = AN31
   ...
   00010 = AN2
   00001 = AN1
   00000 = AN0

Note: Only the first 32 analog inputs (Class 1 and Class 2) can use a digital filter.

bit 15-0 FLTRDATA<15:0>: Digital Filter ‘x’ Data Output Value bits
The filter output data is as per the fractional format set in the FRACT bit (ADCCON1<23>). The FRACT bit should not be changed while the filter is enabled. Changing the state of the FRACT bit after the operation of the filter has ended will not update the value of the FLTRDATA<15:0> bits to reflect the new format.
## Section 22. 12-bit High-Speed SAR ADC

### Register 22-18: ADCTRGL: ADC Trigger Source 1 Register

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Bit 31/23/15/7</th>
<th>Bit 30/22/14/6</th>
<th>Bit 29/21/13/5</th>
<th>Bit 28/20/12/4</th>
<th>Bit 27/19/11/3</th>
<th>Bit 26/18/10/2</th>
<th>Bit 25/17/9/1</th>
<th>Bit 24/16/8/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>TRGSRC3&lt;4:0&gt;</td>
</tr>
<tr>
<td>23:16</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>TRGSRC2&lt;4:0&gt;</td>
</tr>
<tr>
<td>15:8</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>TRGSRC1&lt;4:0&gt;</td>
</tr>
<tr>
<td>7:0</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>TRGSRC0&lt;4:0&gt;</td>
</tr>
</tbody>
</table>

**Legend:**
- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as '0'
- **-n** = Value at POR
- **1** = Bit is set
- **0** = Bit is cleared
- **x** = Bit is unknown

**bit 31-29** Unimplemented: Read as '0'

**bit 28-24** TRGSRC3<4:0>: Trigger Source for Conversion of Analog Input AN3 Select bits
- 11111 - 00100 = Refer to the “ADC” chapter in the specific device data sheet for trigger source selections
- 00011 = Scan Trigger (STRIG)
- 00010 = Global level software trigger (GLSWTRG)
- 00001 = Global software edge Trigger (GSWTRG)
- 00000 = No Trigger

For STRIG, in addition to setting the trigger, it also requires programming of the STRGSRC<4:0> bits (ADCCON1<20:16>) to select the trigger source, and requires the appropriate CSSx bits to be set in the ADCCSSx registers.

**bit 23-21** Unimplemented: Read as '0'

**bit 20-16** TRGSRC2<4:0>: Trigger Source for Conversion of Analog Input AN2 Select bits
See bits 28-24 for bit value definitions.

**bit 15-13** Unimplemented: Read as '0'

**bit 12-8** TRGSRC1<4:0>: Trigger Source for Conversion of Analog Input AN1 Select bits
See bits 28-24 for bit value definitions.

**bit 7-5** Unimplemented: Read as '0'

**bit 4-0** TRGSRC0<4:0>: Trigger Source for Conversion of Analog Input AN0 Select bits
See bits 28-24 for bit value definitions.
### Register 22-19: ADCTRG2: ADC Trigger Source 2 Register

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Bit 31/23/15/7</th>
<th>Bit 30/22/14/6</th>
<th>Bit 29/21/13/5</th>
<th>Bit 28/20/12/4</th>
<th>Bit 27/19/11/3</th>
<th>Bit 26/18/10/2</th>
<th>Bit 25/17/9/1</th>
<th>Bit 24/16/8/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>U-0 U-0 U-0</td>
<td>R/W-0 R/W-0 R/W-0</td>
<td>TRGSRC7&lt;4:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23:16</td>
<td>U-0 U-0 U-0</td>
<td>R/W-0 R/W-0 R/W-0</td>
<td>TRGSRC6&lt;4:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:8</td>
<td>U-0 U-0 U-0</td>
<td>R/W-0 R/W-0 R/W-0</td>
<td>TRGSRC5&lt;4:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td>U-0 U-0 U-0</td>
<td>R/W-0 R/W-0 R/W-0</td>
<td>TRGSRC4&lt;4:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Legend:**
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as ‘0’
- -n = Value at POR
-  ‘1’ = Bit is set
-  ‘0’ = Bit is cleared
-  x = Bit is unknown

#### bit 31-29 Unimplemented: Read as ‘0’

#### bit 28-24 TRGSRC7<4:0>: Trigger Source for Conversion of Analog Input AN7 Select bits
- 11111 00100 = Refer to the “ADC” chapter in the specific device data sheet for trigger source selections
- 00011 = Scan Trigger (STRIG)
- 00010 = Global level software trigger (GLSWTRG)
- 00001 = Global software edge Trigger (GSWTRG)
- 00000 = No Trigger

For STRIG, in addition to setting the trigger, it also requires programming of the STRGSRC<4:0> bits (ADCCON1<20:16>) to select the trigger source, and requires the appropriate CSSx bits to be set in the ADCCSSx registers.

#### bit 23-21 Unimplemented: Read as ‘0’

#### bit 20-16 TRGSRC6<4:0>: Trigger Source for Conversion of Analog Input AN6 Select bits
See bits 28-24 for bit value definitions.

#### bit 15-13 Unimplemented: Read as ‘0’

#### bit 12-8 TRGSRC5<4:0>: Trigger Source for Conversion of Analog Input AN5 Select bits
See bits 28-24 for bit value definitions.

#### bit 7-5 Unimplemented: Read as ‘0’

#### bit 4-0 TRGSRC4<4:0>: Trigger Source for Conversion of Analog Input AN4 Select bits
See bits 28-24 for bit value definitions.
## Section 22. 12-bit High-Speed SAR ADC

### Register 22-20: ADCTRG3: ADC Trigger Source 3 Register

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Bit 31/23/15/7</th>
<th>Bit 30/22/14/6</th>
<th>Bit 29/21/13/5</th>
<th>Bit 28/20/12/4</th>
<th>Bit 27/19/11/3</th>
<th>Bit 26/18/10/2</th>
<th>Bit 25/17/9/1</th>
<th>Bit 24/16/8/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>23:16</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>15:8</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>7:0</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
</tbody>
</table>

Legend:
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as '0'
- -n = Value at POR
- '1' = Bit is set
- '0' = Bit is cleared
- x = Bit is unknown

bit 31-29 **Unimplemented**: Read as '0'

bit 28-24 **TRGSRC11<4:0>**: Trigger Source for Conversion of Analog Input AN11 Select bits

01111 = Refer to the "ADC" chapter in the specific device data sheet for trigger source selections
00110 = Scan Trigger (STRIG)
00101 = Global level software trigger (GLSWTRG)
00011 = Global software edge Trigger (GSWTRG)
00000 = No Trigger

For STRIG, in addition to setting the trigger, it also requires programming of the STRGSRC<4:0> bits (ADCCON1<20:16>) to select the trigger source, and requires the appropriate CSSx bits to be set in the ADCCSSx registers.

bit 23-21 **Unimplemented**: Read as '0'

bit 20-16 **TRGSRC10<4:0>**: Trigger Source for Conversion of Analog Input AN10 Select bits

See bits 28-24 for bit value definitions.

bit 15-13 **Unimplemented**: Read as '0'

bit 12-8 **TRGSRC9<4:0>**: Trigger Source for Conversion of Analog Input AN9 Select bits

See bits 28-24 for bit value definitions.

bit 7-5 **Unimplemented**: Read as '0'

bit 4-0 **TRGSRC8<4:0>**: Trigger Source for Conversion of Analog Input AN8 Select bits

See bits 28-24 for bit value definitions.
## Register 22-21: ADCTRG4: ADC Trigger Source 4 Register

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Bit 31/23/15/7</th>
<th>Bit 28/20/12/4</th>
<th>Bit 27/19/11/3</th>
<th>Bit 26/18/10/2</th>
<th>Bit 25/17/9/1</th>
<th>Bit 24/16/8/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>23:16</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>15:8</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>7:0</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
</tbody>
</table>

### Legend:
- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as '0'
- **-n** = Value at POR
  - ‘1’ = Bit is set
  - ‘0’ = Bit is cleared
  - x = Bit is unknown

#### bit 31-29
Unimplemented: Read as ‘0’

#### bit 28-24
**TRGSRC15<4:0>:** Trigger Source for Conversion of Analog Input AN15 Select bits
- 11111 = Scan Trigger (STRIG)
- 00101 = Global level software trigger (GLSWTRG)
- 00011 = Global software edge Trigger (GSWTRG)
- 00000 = No Trigger

For STRIG, in addition to setting the trigger, it also requires programming of the STRGSRC<4:0> bits (ADCCON1<20:16>) to select the trigger source, and requires the appropriate CSSx bits to be set in the ADCCSSx registers.

#### bit 23-21
Unimplemented: Read as ‘0’

#### bit 20-16
**TRGSRC14<4:0>:** Trigger Source for Conversion of Analog Input AN14 Select bits
See bits 28-24 for bit value definitions.

#### bit 15-13
Unimplemented: Read as ‘0’

#### bit 12-8
**TRGSRC13<4:0>:** Trigger Source for Conversion of Analog Input AN13 Select bits
See bits 28-24 for bit value definitions.

#### bit 7-5
Unimplemented: Read as ‘0’

#### bit 4-0
**TRGSRC12<4:0>:** Trigger Source for Conversion of Analog Input AN12 Select bits
See bits 28-24 for bit value definitions.
Register 22-22: ADCTRG5: ADC Trigger Source 5 Register

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Bit 31/23/15/7</th>
<th>Bit 30/22/14/6</th>
<th>Bit 29/21/13/5</th>
<th>Bit 28/20/12/4</th>
<th>Bit 27/19/11/3</th>
<th>Bit 26/18/10/2</th>
<th>Bit 25/17/9/1</th>
<th>Bit 24/16/8/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>23:16</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>15:8</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>7:0</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
</tbody>
</table>

Legend:
R = Readable bit  W = Writable bit  U = Unimplemented bit, read as '0'
-n = Value at POR  '1' = Bit is set  '0' = Bit is cleared  x = Bit is unknown

bit 31-29 Unimplemented: Read as '0'
bit 28-24 TRGSRC19<4:0>: Trigger Source for Conversion of Analog Input AN19 Select bits
11111 - 00100 = Refer to the “ADC” chapter in the specific device data sheet for trigger source selections
00011 = Scan Trigger (STRIG)
00010 = Global level software trigger (GLSWTRG)
00001 = Global software edge Trigger (GSWTRG)
00000 = No Trigger
For STRIG, in addition to setting the trigger, it also requires programming of the STRGSRC<4:0> bits (ADCCON1<20:16>) to select the trigger source, and requires the appropriate CSSx bits to be set in the ADCCSSx registers.

bit 23-21 Unimplemented: Read as '0'
bit 20-16 TRGSRC18<4:0>: Trigger Source for Conversion of Analog Input AN18 Select bits
See bits 28-24 for bit value definitions.

bit 15-13 Unimplemented: Read as '0'
bit 12-8 TRGSRC17<4:0>: Trigger Source for Conversion of Analog Input AN17 Select bits
See bits 28-24 for bit value definitions.

bit 7-5 Unimplemented: Read as '0'
bit 4-0 TRGSRC16<4:0>: Trigger Source for Conversion of Analog Input AN16 Select bits
See bits 28-24 for bit value definitions.
Register 22-23: ADCTRG6: ADC Trigger Source 6 Register

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Bit 31/23/15/7</th>
<th>Bit 30/22/14/6</th>
<th>Bit 29/21/13/5</th>
<th>Bit 28/20/12/4</th>
<th>Bit 27/19/11/3</th>
<th>Bit 26/18/10/2</th>
<th>Bit 25/17/9/1</th>
<th>Bit 24/16/8/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td></td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>TRGSRC23&lt;4:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23:16</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td></td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>TRGSRC22&lt;4:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:8</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td></td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>TRGSRC21&lt;4:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td></td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>TRGSRC20&lt;4:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend:
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as ‘0’
- -n = Value at POR
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- x = Bit is unknown

bit 31-29 Unimplemented: Read as ‘0’

bit 28-24 TRGSRC23<4:0>: Trigger Source for Conversion of Analog Input AN23 Select bits

11111-00100 = Refer to the “ADC” chapter in the specific device data sheet for information

00011 = Scan Trigger (STRIG)
00010 = Global level software trigger (GLSWTRG)
00001 = Global software edge Trigger (GSWTRG)
00000 = No Trigger

For STRIG, in addition to setting the trigger, it also requires programming of the STRGSRC<4:0> bits (ADCCON1<20:16>) to select the trigger source, and requires the appropriate CSSx bits to be set in the ADCCSx registers.

bit 23-21 Unimplemented: Read as ‘0’

bit 20-16 TRGSRC22<4:0>: Trigger Source for Conversion of Analog Input AN22 Select bits

See bits 28-24 for bit value definitions.

bit 15-13 Unimplemented: Read as ‘0’

bit 12-8 TRGSRC21<4:0>: Trigger Source for Conversion of Analog Input AN21 Select bits

See bits 28-24 for bit value definitions.

bit 7-5 Unimplemented: Read as ‘0’

bit 4-0 TRGSRC20<4:0>: Trigger Source for Conversion of Analog Input AN20 Select bits

See bits 28-24 for bit value definitions.
### Section 22. 12-bit High-Speed SAR ADC

#### Register 22-24: ADCTRG7: ADC Trigger Source 7 Register

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Bit 31/23/15/7</th>
<th>Bit 30/22/14/6</th>
<th>Bit 29/21/13/5</th>
<th>Bit 28/20/12/4</th>
<th>Bit 27/19/11/3</th>
<th>Bit 26/18/10/2</th>
<th>Bit 25/17/9/1</th>
<th>Bit 24/16/8/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>23:16</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>15:8</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>7:0</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
</tbody>
</table>

**Legend:**
- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as '0'
- **-n** = Value at POR
- '1' = Bit is set
- '0' = Bit is cleared
- **x** = Bit is unknown

#### bit 31-29 Unimplemented: Read as '0'

#### bit 28-24 TRGSRC27<4:0>: Trigger Source for Conversion of Analog Input AN27 Select bits

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>11111-00100</td>
<td>Refer to the “ADC” chapter in the specific device data sheet for information</td>
</tr>
<tr>
<td>00011</td>
<td>Scan Trigger (STRIG)</td>
</tr>
<tr>
<td>00010</td>
<td>Global level software trigger (GLSWTRG)</td>
</tr>
<tr>
<td>00001</td>
<td>Global software edge Trigger (GSWTRG)</td>
</tr>
<tr>
<td>00000</td>
<td>No Trigger</td>
</tr>
</tbody>
</table>

For STRIG, in addition to setting the trigger, it also requires programming of the STRGSRC<4:0> bits (ADCCON1<20:16>) to select the trigger source, and requires the appropriate CSSx bits to be set in the ADCCSSx registers.

#### bit 23-21 Unimplemented: Read as '0'

#### bit 20-16 TRGSRC26<4:0>: Trigger Source for Conversion of Analog Input AN26 Select bits

See bits 28-24 for bit value definitions.

#### bit 15-13 Unimplemented: Read as '0'

#### bit 12-8 TRGSRC25<4:0>: Trigger Source for Conversion of Analog Input AN25 Select bits

See bits 28-24 for bit value definitions.

#### bit 7-5 Unimplemented: Read as '0'

#### bit 4-0 TRGSRC24<4:0>: Trigger Source for Conversion of Analog Input AN24 Select bits

See bits 28-24 for bit value definitions.
**Register 22-25: ADCTRG8: ADC Trigger Source 8 Register**

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Bit 31/23/15/7</th>
<th>Bit 29/21/13/5</th>
<th>Bit 28/20/12/4</th>
<th>Bit 27/19/11/3</th>
<th>Bit 26/18/10/2</th>
<th>Bit 25/17/9/1</th>
<th>Bit 24/16/8/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23:16</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:8</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend:
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as ‘0’
- -n = Value at POR
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- x = Bit is unknown

bit 31-29 **Unimplemented:** Read as ‘0’

bit 28-24 **TRGSRC31<4:0>:** Trigger Source for Conversion of Analog Input AN31 Select bits

11111 = Refer to the “ADC” chapter in the specific device data sheet for information
00011 = Scan Trigger (STRIG)
00010 = Global level software trigger (GLSWTRG)
00001 = Global software edge Trigger (GSWTRG)
00000 = No Trigger

For STRIG, in addition to setting the trigger, it also requires programming of the TRGSRCC<4:0> bits (ADCCON1<20:16>) to select the trigger source, and requires the appropriate CSSx bits to be set in the ADCCSSx registers.

bit 23-21 **Unimplemented:** Read as ‘0’

bit 20-16 **TRGSRC30<4:0>:** Trigger Source for Conversion of Analog Input AN30 Select bits
See bits 28-24 for bit value definitions.

bit 15-13 **Unimplemented:** Read as ‘0’

bit 12-8 **TRGSRC29<4:0>:** Trigger Source for Conversion of Analog Input AN29 Select bits
See bits 28-24 for bit value definitions.

bit 7-5 **Unimplemented:** Read as ‘0’

bit 4-0 **TRGSRC28<4:0>:** Trigger Source for Conversion of Analog Input AN28 Select bits
See bits 28-24 for bit value definitions.
Section 22. 12-bit High-Speed SAR ADC

Register 22-26: ADCCMPCON1: ADC Digital Comparator 1 Control Register

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Bit 31:24</th>
<th>Bit 23:16</th>
<th>Bit 15:8</th>
<th>Bit 7:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
</tr>
<tr>
<td>23:16</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
</tr>
<tr>
<td>15:8</td>
<td>U-0, U-0</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>END CMP</td>
</tr>
<tr>
<td>7:0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>DCMPGIEN</td>
</tr>
</tbody>
</table>

**Legend:**
- HS = Hardware Set
- HC = Hardware Cleared
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as ‘0’
- -n = Value at POR
  - ‘1’ = Bit is set
  - ‘0’ = Bit is cleared
  - x = Bit is unknown

Bit 31-16: **CVDDATA<15:0>: CVD Data Status bits**

In CVD mode, these bits obtain the CVD differential output data (subtraction of CVD positive and negative measurement), whenever a Digital Comparator event is generated. The value in these bits is compliant with the FRACT bit (ADCCON1<23>) and is always signed.

Bit 15-14: **Unimplemented:** Read as ‘0’

Bit 13-8: **AINID<5:0>: Digital Comparator 1 Analog Input Identification (ID) bits**

When a digital comparator event occurs (DCMPED = 1), these bits identify the analog input being monitored by Digital Comparator 1.

**Note:** In normal ADC mode, only analog inputs <31:0> can be processed by the Digital Comparator 1. The Digital Comparator 1 also supports the CVD mode, in which Class 2 and Class 3 analog inputs may be stored in the AINID<5:0> bits.

- 111111 = AN63 is being monitored
- 111110 = AN62 is being monitored
- ...
- 000001 = AN1 is being monitored
- 000000 = AN0 is being monitored

Bit 7: **ENDCMP:** Digital Comparator 1 Enable bit

- 1 = Digital Comparator 1 is enabled
- 0 = Digital Comparator 1 is not enabled, and the DCMPED status bit (ADCCMPCON1<5>) is cleared

Bit 6: **DCMPG IEN:** Digital Comparator 1 Interrupt Enable bit

- 1 = A Digital Comparator 1 interrupt is generated when the DCMPED status bit (ADCCMPCON1<5>) is set
- 0 = A Digital Comparator 1 interrupt is disabled

Bit 5: **DCMPED:** Digital Comparator 1 "Output True" Event Status bit

The logical conditions under which the digital comparator gets "True" are defined by the IEBTWN, IEHIHI, IEHILO, IELOHI, and IELOLO bits.

**Note:** This bit is cleared by reading the AINID<5:0> bits or by disabling the Digital Comparator module (by setting ENDCMP to ‘0’).

- 1 = Digital Comparator 1 output true event has occurred (output of Comparator is ‘1’)
- 0 = Digital Comparator 1 output is false (output of comparator is ‘0’)

Bit 4: **IEBTWN:** Between Low/High Digital Comparator 1 Event bit

- 1 = Generate a digital comparator event when DCMPLO<15:0> ≤ DATA<31:0> < DCMPHI<15:0>
- 0 = Do not generate a digital comparator event

Bit 3: **IEHIHI:** High/High Digital Comparator 1 Event bit

- 1 = Generate a Digital Comparator 1 Event when DCMPHI<15:0> ≤ DATA<31:0>
- 0 = Do not generate an event

Bit 2: **IEHILO:** High/Low Digital Comparator 1 Event bit

- 1 = Generate a Digital Comparator 1 Event when DATA<31:0> < DCMPHI<15:0>
- 0 = Do not generate an event
Register 22-26: ADCCMPCON1: ADC Digital Comparator 1 Control Register

- **IELOHI**: Low/High Digital Comparator 1 Event bit
  - 1 = Generate a Digital Comparator 1 Event when DCMPLO<15:0> ≤ DATA<31:0>
  - 0 = Do not generate an event

- **IELOLO**: Low/Low Digital Comparator 1 Event bit
  - 1 = Generate a Digital Comparator 1 Event when DATA<31:0> < DCMPLO<15:0>
  - 0 = Do not generate an event
### Section 22. 12-bit High-Speed SAR ADC

#### Register 22-27: ADCCMPCONx: ADC Digital Comparator 'x' Control Register ('x' = 2 through 6)

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Bit 31/23/15/7</th>
<th>Bit 29/21/13/5</th>
<th>Bit 27/19/11/3</th>
<th>Bit 25/17/9/1</th>
<th>Bit 24/16/8/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>U-0 U-0 U-0 U-0</td>
<td>U-0 U-0 U-0 U-0</td>
<td>U-0 U-0 U-0 U-0</td>
<td>U-0 U-0 U-0 U-0</td>
<td>U-0 U-0 U-0 U-0</td>
</tr>
<tr>
<td>23:16</td>
<td>U-0 U-0 U-0 U-0</td>
<td>U-0 U-0 U-0 U-0</td>
<td>U-0 U-0 U-0 U-0</td>
<td>U-0 U-0 U-0 U-0</td>
<td>U-0 U-0 U-0 U-0</td>
</tr>
<tr>
<td>15:8</td>
<td>U-0 U-0 U-0 U-0</td>
<td>R-0, HS, HC R-0, HS, HC R-0, HS, HC R-0, HS, HC</td>
<td>AINID&lt;4:0&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td>R/W-0 R/W-0 R-0, HS, HC R/W-0 R/W-0 R/W-0 R/W-0 R/W-0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Legend:**
- HS = Hardware Set
- HC = Hardware Cleared
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as '0'
- -n = Value at POR
- '1' = Bit is set
- '0' = Bit is cleared
- x = Bit is unknown

### bit 31-13: Unimplemented
Read as '0'

### bit 12-8: AINID<4:0>: Digital Comparator 'x' Analog Input Identification (ID) bits
When a digital comparator event occurs (DCMPED = 1), these bits identify the analog input being monitored by the Digital Comparator.

**Note:** Only analog inputs <31:0> can be processed by the Digital Comparator module 'x' (‘x’ = 2-6).

- 11111 = AN31 is being monitored
- 11110 = AN30 is being monitored
- ...
- 00001 = AN1 is being monitored
- 00000 = AN0 is being monitored

### bit 7: ENDCMP: Digital Comparator 'x' Enable bit

- 1 = Digital Comparator 'x' is enabled
- 0 = Digital Comparator 'x' is not enabled, and the DCMPED status bit (ADCCMPCONx<5>) is cleared

### bit 6: DCMPGIEN: Digital Comparator 'x' Interrupt Enable bit

- 1 = A Digital Comparator 'x' interrupt is generated when the DCMPED status bit (ADCCMPCONx<5>) is set
- 0 = A Digital Comparator 'x' interrupt is disabled

### bit 5: DCMPED: Digital Comparator 'x' “Output True” Event Status bit
The logical conditions under which the digital comparator gets “True” are defined by the IEBTWN, IEHIHI, IEHILO, IELOHI and IELOLO bits.

**Note:** This bit is cleared by reading the AINID<5:0> bits (ADCCMPCON1<13:8>) or by disabling the Digital Comparator module (by setting ENDCMP to ‘0’).

- 1 = Digital Comparator 'x' output true event has occurred (output of Comparator is '1')
- 0 = Digital Comparator 'x' output is false (output of Comparator is '0')

### bit 4: IEBTWN: Between Low/High Digital Comparator 'x' Event bit

- 1 = Generate a digital comparator event when the DCMPLO<15:0> bits < DATA<31:0> bits
- 0 = Do not generate a digital comparator event

### bit 3: IEHIHI: High/High Digital Comparator 'x' Event bit

- 1 = Generate a Digital Comparator 'x' Event when the DCMPHI<15:0> bits < DATA<31:0> bits
- 0 = Do not generate an event

### bit 2: IEHILO: High/Low Digital Comparator 'x' Event bit

- 1 = Generate a Digital Comparator 'x' Event when the DATA<31:0> bits < DCMPHI<15:0> bits
- 0 = Do not generate an event
Register 22-27: ADCCMPCONx: ADC Digital Comparator ‘x’ Control Register (‘x’ = 2 through 6) (Continued)

bit 1  IELOHI: Low/High Digital Comparator ‘x’ Event bit
       1 = Generate a Digital Comparator ‘x’ Event when the DCMPLO<15:0> bits \( \leq \) DATA<31:0> bits
       0 = Do not generate an event

bit 0  IELOLO: Low/Low Digital Comparator ‘x’ Event bit
       1 = Generate a Digital Comparator ‘x’ Event when the DATA<31:0> bits < DCMPLO<15:0> bits
       0 = Do not generate an event
### Section 22. 12-bit High-Speed SAR ADC

#### Register 22-28: ADCFSTAT: ADC FIFO Status Register

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Bit 31/23</th>
<th>Bit 28/20</th>
<th>Bit 27/19</th>
<th>Bit 26/18</th>
<th>Bit 25/17</th>
<th>Bit 24/16</th>
<th>Bit 23/16</th>
<th>Bit 22/14</th>
<th>Bit 21/13</th>
<th>Bit 20/12</th>
<th>Bit 19/11</th>
<th>Bit 18/10</th>
<th>Bit 17/9</th>
<th>Bit 16/8</th>
<th>Bit 15/8</th>
<th>Bit 14/7</th>
<th>Bit 13/6</th>
<th>Bit 12/5</th>
<th>Bit 11/4</th>
<th>Bit 10/3</th>
<th>Bit 9/2</th>
<th>Bit 8/1</th>
<th>Bit 7/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>FEN</td>
<td>ADC6EN</td>
<td>ADC5EN</td>
<td>ADC4EN</td>
<td>ADC3EN</td>
<td>ADC2EN</td>
<td>ADC1EN</td>
<td>ADC0EN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23:16</td>
<td>R-0, HS, HC</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>FIEN</td>
<td>FRDY</td>
<td>FWROVERR</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>15:8</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>FCNT&lt;7:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td>R-0</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>FSIGN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Legend:
- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as ‘0’
- **-n** = Value at POR
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- **x** = Bit is unknown

**bit 31**
- **FEN**: FIFO Enable bit
  - 1 = FIFO is enabled
  - 0 = FIFO is disabled; no data is being saved into the FIFO

**bit 30-24**
- **ADC6EN:ADC0EN**: ADC6-ADC0 Enable bits
  - 1 = Converted output data of ADC6-ADC0 is stored in the FIFO
  - 0 = Converted output data of ADC6-ADC0 is not stored in the FIFO

  **Note:** While using FIFO, the output data is additionally stored in the respective output data register (ADCDATAx).

**bit 23**
- **FIEN**: FIFO Interrupt Enable bit
  - 1 = FIFO interrupts are enabled; an interrupt is generated once the FRDY bit is set
  - 0 = FIFO interrupts are disabled

**bit 22**
- **FRDY**: FIFO Data Ready Interrupt Status bit
  - 1 = FIFO has data to be read
  - 0 = No data is available in the FIFO

  **Note:** This bit is cleared when the FIFO output data in ADCFIFO has been read and there is no additional data ready in the FIFO (that is, the FIFO is empty).

**bit 21**
- **FWROVERR**: FIFO Write Overflow Error Status bit
  - 1 = A write overflow error in the FIFO has occurred (circular FIFO)
  - 0 = A write overflow error in the FIFO has not occurred

  **Note:** This bit is cleared after ADCFSTAT<23:16> are read by software.

**bit 15-8**
- **FCNT<7:0>**: FIFO Data Entry Count Status bit
  - The value in these bits indicates the number of data entries in the FIFO.

**bit 7**
- **FSIGN**: FIFO Sign Setting bit
  - This bit reflects the sign of data stored in the ADCFIFO register.

**bit 6-3**
- **Unimplemented**: Read as ‘0’

**bit 2-0**
- **ADCID<2:0>**: ADC6-ADC0 Identifier bits
  - These bits specify the ADC module whose data is stored in the FIFO.
  - 111 = Reserved
  - 110 = Converted data of ADC6 is stored in FIFO
  - ...  
  - 000 = Converted data of ADC0 is stored in FIFO
Register 22-29: ADCFIFO: ADC FIFO Data Register

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Bit 31/23/15/7</th>
<th>Bit 30/22/14/6</th>
<th>Bit 29/21/13/5</th>
<th>Bit 28/20/12/4</th>
<th>Bit 27/19/11/3</th>
<th>Bit 26/18/10/2</th>
<th>Bit 25/17/9/1</th>
<th>Bit 24/16/8/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
</tr>
<tr>
<td>23:16</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
</tr>
<tr>
<td>15:8</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
</tr>
<tr>
<td>7:0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
</tr>
</tbody>
</table>

Legend:
R = Readable bit  W = Writable bit  U = Unimplemented bit, read as ‘0’
-n = Value at POR  ‘1’ = Bit is set  ‘0’ = Bit is cleared  x = Bit is unknown

bit 31-0  DATA<31:0>: FIFO Data Output Value bits

Note: When an alternate input is used as the input source for a dedicated ADC module, the data output is still read from the Primary input Data Output Register.

Register 22-30: ADCBASE: ADC Base Register

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Bit 31/23/15/7</th>
<th>Bit 30/22/14/6</th>
<th>Bit 29/21/13/5</th>
<th>Bit 28/20/12/4</th>
<th>Bit 27/19/11/3</th>
<th>Bit 26/18/10/2</th>
<th>Bit 25/17/9/1</th>
<th>Bit 24/16/8/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
</tr>
<tr>
<td>23:16</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
</tr>
<tr>
<td>15:8</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>7:0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
</tbody>
</table>

Legend:
R = Readable bit  W = Writable bit  U = Unimplemented bit, read as ‘0’
-n = Value at POR  ‘1’ = Bit is set  ‘0’ = Bit is cleared  x = Bit is unknown

bit 31-0  Unimplemented: Read as ‘0’

bit 15-0  ADCBASE<15:0>: ADC ISR Base Address bits

This register, when read, contains the base address of the user's ADC ISR jump table. The interrupt vector address is determined by the IRQVS<2:0> bits of the ADCCON1 register specifying the amount of left shift done to the ARDYx status bits in the ADCDSTAT1 and ADCDSTAT2 registers, prior to adding with ADCBASE register.

Interrupt Vector Address = Read Value of ADCBASE = Value written to ADCBASE + x << IRQVS<2:0>, where ‘x’ is the smallest active analog input ID from the ADCDSTAT1 or ADCDSTAT2 registers (which has highest priority).
Section 22. 12-bit High-Speed SAR ADC

Register 22-31: ADCDATAx: ADC Output Data Register ('x' = 0 through 63)

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Bit 31/23/15/7</th>
<th>Bit 30/22/14/6</th>
<th>Bit 29/21/13/5</th>
<th>Bit 28/20/12/4</th>
<th>Bit 27/19/11/3</th>
<th>Bit 26/18/10/2</th>
<th>Bit 25/17/9/1</th>
<th>Bit 24/16/8/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
</tr>
<tr>
<td>23:16</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
</tr>
<tr>
<td>15:8</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
</tr>
<tr>
<td>7:0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
</tr>
</tbody>
</table>

Legend:
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as ‘0’
- -n = Value at POR
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- x = Bit is unknown

bit 31-0 DATA<31:0>: ADC Converted Data Output bits.

**Note 1:** When an alternate input is used as the input source for a dedicated ADC module, the data output is still read from the Primary input Data Output Register.

**Note 2:** Reading the ADCDATAx register value after changing the FRACT bit converts the data into the format specified by FRACT bit.
## Register 22-32: ADCDMASTAT: ADC DMA Status Register

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Bit 31/23/15/7</th>
<th>Bit 28/20/12/4</th>
<th>Bit 27/19/11/3</th>
<th>Bit 26/18/10/2</th>
<th>Bit 25/17/9/1</th>
<th>Bit 24/16/8/0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DMAGEN</td>
<td>DMACNTEN</td>
<td>DMAWROVERR</td>
<td>RBF6:RBF0</td>
<td>RAFIEN6:RAFIEN0</td>
<td>RAF6:RAF0</td>
</tr>
<tr>
<td>31:24</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>23:16</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
</tr>
<tr>
<td>15:8</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>7:0</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
</tr>
</tbody>
</table>

### Legend:
- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as '0'
- **-n** = Value at POR
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- **x** = Bit is unknown

### Bit 31: DMAGEN
- **DMA Global Enable bit**
  - 1 = DMA engine is enabled globally for all ADC modules. To use DMA, individual dedicated ADC modules should enable their DMAEN bits (ADCxTIME<23>).
  - 0 = DMA engine is globally disabled for all ADC modules.

### Bit 20-24: RBFIEN6:RBFIEN0
- **RAM Buffer B Full Interrupt Enable bit**
  - 1 = Interrupts are enabled and generated when the RBFx Status bit is set
  - 0 = Interrupts are disabled

### Bit 15: DMAWROVERR
- **DMA Write Overflow Error bit**
  - 1 = DMA write overflow error has occurred (circular buffer)
  - 0 = DMA write overflow error has not occurred

### Note:
- This bit is cleared by hardware after a software read of the ADCDMASTAT register.

### Bit 14-8: RAFIEN6:RAFIEN0
- **RAM Buffer A Full Interrupt Enable bit**
  - 1 = Interrupts are enabled and generated when the RAFx Status bit is set
  - 0 = Interrupts are disabled

### Bit 7: Unimplemented
- Read as ‘0’

### Bit 6-0: RAF6:RAF0
- **RAM Buffer A Full Status bit**
  - 1 = RAM Buffer A is full
  - 0 = RAM Buffer A is not full

### Note:
- These bits are self-clearing upon being a read by software.
Section 22. 12-bit High-Speed SAR ADC

Register 22-33: ADCCNTB: ADC Sample Count Base Address Register

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Bit 31/23/15/7</th>
<th>Bit 30/22/14/6</th>
<th>Bit 29/21/13/5</th>
<th>Bit 28/20/12/4</th>
<th>Bit 27/19/11/3</th>
<th>Bit 26/18/10/2</th>
<th>Bit 25/17/9/1</th>
<th>Bit 24/16/8/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>CNTBADDR&lt;31:24&gt;</td>
</tr>
<tr>
<td>23:16</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>CNTBADDR&lt;23:16&gt;</td>
</tr>
<tr>
<td>15:8</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>CNTBADDR&lt;15:8&gt;</td>
</tr>
<tr>
<td>7:0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>CNTBADDR&lt;7:0&gt;</td>
</tr>
</tbody>
</table>

Legend:
R = Readable bit  W = Writable bit  U = Unimplemented bit, read as ‘0’
-n = Value at POR  ‘1’ = Bit is set  ‘0’ = Bit is cleared  x = Bit is unknown

bit 31  CNTBADDR<31:0>: Analog-to-Digital Count Base Address bits

The ADCCNTB register contains the user-defined RAM address at which the DMA engine will start saving the current count of output samples (if the DMACNTEN bit (ADCDMASTAT<15>) is set), which is already written to each of the buffers in the System RAM for each ADC module. The ADCx module will have its Buffer A current sample count saved at the address ((ADCCNTB) + 2 * x) and its Buffer B current sample count saved at the address ((ADCCNTB) + (2 * x + 1)). Where ‘x’ is the dedicated ADC module ID.

Register 22-34: ADCDMAB: ADC DMA Base Address Register

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Bit 31/23/15/7</th>
<th>Bit 30/22/14/6</th>
<th>Bit 29/21/13/5</th>
<th>Bit 28/20/12/4</th>
<th>Bit 27/19/11/3</th>
<th>Bit 26/18/10/2</th>
<th>Bit 25/17/9/1</th>
<th>Bit 24/16/8/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>DMABADDR&lt;31:24&gt;</td>
</tr>
<tr>
<td>23:16</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>DMABADDR&lt;23:16&gt;</td>
</tr>
<tr>
<td>15:8</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>DMABADDR&lt;15:8&gt;</td>
</tr>
<tr>
<td>7:0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>DMABADDR&lt;7:0&gt;</td>
</tr>
</tbody>
</table>

Legend:
R = Readable bit  W = Writable bit  U = Unimplemented bit, read as ‘0’
-n = Value at POR  ‘1’ = Bit is set  ‘0’ = Bit is cleared  x = Bit is unknown

bit 31  DMABADDR<31:0>: DMA Base Address bits

The ADCDMAB register contains the user-specified RAM address at which DMA engine will start saving the converted data. The address of saving each data is specified by the following relations:
- Buffer A starting address at: ADCDMAB + (2 * x) * 2^ADCON1bits.DMABL + 1
- Buffer B starting at: ADCDMAB + (2 * (x + 1)) * 2^ADCON1bits.DMABL + 1

Where, ‘x’ is the dedicated ADC module ID.
### Register 22-35: ADCTRGSNS: ADC Trigger Level/Edge Sensitivity Register

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Bit 31/23/15/7</th>
<th>Bit 30/22/14/6</th>
<th>Bit 29/21/13/5</th>
<th>Bit 28/20/12/4</th>
<th>Bit 27/19/11/3</th>
<th>Bit 26/18/10/2</th>
<th>Bit 25/17/9/1</th>
<th>Bit 24/16/8/0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>31:24</td>
<td>LVL31</td>
<td>LVL30</td>
<td>LVL29</td>
<td>LVL28</td>
<td>LVL27</td>
<td>LVL26</td>
<td>LVL25</td>
<td>LVL24</td>
</tr>
<tr>
<td>23:16</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td></td>
<td>LVL23</td>
<td>LVL22</td>
<td>LVL21</td>
<td>LVL20</td>
<td>LVL19</td>
<td>LVL18</td>
<td>LVL17</td>
<td>LVL16</td>
</tr>
<tr>
<td>15:8</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td></td>
<td>LVL15</td>
<td>LVL14</td>
<td>LVL13</td>
<td>LVL12</td>
<td>LVL11</td>
<td>LVL10</td>
<td>LVL9</td>
<td>LVL8</td>
</tr>
<tr>
<td>7:0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td></td>
<td>LVL7</td>
<td>LVL6</td>
<td>LVL5</td>
<td>LVL4</td>
<td>LVL3</td>
<td>LVL2</td>
<td>LVL1</td>
<td>LVL0</td>
</tr>
</tbody>
</table>

**Legend:**
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as '0'
- -n = Value at POR
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- x = Bit is unknown

**bit 31**

LVL31:LVL0: Trigger Level and Edge Sensitivity bits
1 = Analog input is sensitive to the high level of its trigger (level sensitivity implies retriggering as long as the trigger signal remains high)
0 = Analog input is sensitive to the positive edge of its trigger (this is the value after a reset)

**Note 1:** This register specifies the trigger level for analog inputs 0 to 31.
**Note 2:** The higher analog input ID belongs to Class 3, and therefore, is only scan triggered. All Class 3 analog inputs use the Scan Trigger, for which the level/edge is defined by the STRGLVL bit (ADCCON1<3>).
Section 22. 12-bit High-Speed SAR ADC

Register 22-36: ADCxTIME: Dedicated ADCx Timing Register ‘x’ (‘x’ = 0 through 6)

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Bit 31:23/15/7</th>
<th>Bit 29/11/13/5</th>
<th>Bit 28/20/12/4</th>
<th>Bit 27/19/11/3</th>
<th>Bit 26/18/10/2</th>
<th>Bit 25/17/9/1</th>
<th>Bit 24/16/8/0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>U-0</td>
<td>U-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-1</td>
<td>R/W-1</td>
</tr>
<tr>
<td>31:24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ADCEIS&lt;2:0&gt;</td>
<td>SELRES&lt;1:0&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23:16</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DMAEN</td>
<td>ADCDIV&lt;6:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:8</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>U-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SAMC&lt;9:8&gt;</td>
</tr>
<tr>
<td>7:0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
</tbody>
</table>

Legend:
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as ‘0’
- -n = Value at POR
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- x = Bit is unknown

bit 31-29  Unimplemented: Read as ‘0’

bit 28-26  ADCEIS<2:0>: ADCx Early Interrupt Select bits

111 = The data ready interrupt is generated 8 ADC clocks prior to the end of conversion
110 = The data ready interrupt is generated 7 ADC clocks prior to the end of conversion
... = The data ready interrupt is generated 4 ADC clocks prior to the end of conversion
010 = The data ready interrupt is generated 1 ADC clock prior to the end of conversion
000 = The data ready interrupt is generated 1 ADC clock prior to the end of conversion

Note: All options are available when the selected resolution, specified by the SELRES<1:0> bits (ADCxTIME<25:24>), is 12-bit or 10-bit. For a selected resolution of 8-bit, options from ‘000’ to ‘101’ are valid. For a selected resolution of 6-bit, options from ‘000’ to ‘011’ are valid.

bit 25-24  SELRES<1:0>: ADC Resolution Select bits

11 = 12 bits
10 = 10 bits
01 = 8 bits
00 = 6 bits

bit 23  DMAEN: DMA Enable bit

1 = DMA engine is enabled for the ADC module. The ADC can save data on system RAM.
0 = DMA engine is disabled for the ADC module. The converted data can only be retrieved from the respective data register.

bit 22-16  ADCDIV<6:0>: ADC Clock Divisor bits

These bits divide the ADC control clock with period Tq to generate the clock for ADC (TAD).

11111111 = 254 * Tq = TAD
... = 6 * Tq = TAD
... = 4 * Tq = TAD
... = 2 * Tq = TAD
00000000 = Reserved

bit 15-10  Unimplemented: Read as ‘0’

bit 9-0  SAMC<9:0>: ADC Sample Time bits

Where TAD = period of the ADC conversion clock for the dedicated ADC controlled by the ADCDIV<6:0> bits.

1111111111 = 1025 TAD
... = 3 TAD
... = 2 TAD
### Register 22-37: ADCEIEN1: ADC Early Interrupt Enable Register 1

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Bit 31/23/15/7</th>
<th>Bit 30/22/14/6</th>
<th>Bit 29/21/13/5</th>
<th>Bit 28/20/12/4</th>
<th>Bit 27/19/11/3</th>
<th>Bit 26/18/10/2</th>
<th>Bit 25/17/9/1</th>
<th>Bit 24/16/8/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
</tr>
<tr>
<td>EIEN31</td>
<td>EIEN30</td>
<td>EIEN29</td>
<td>EIEN28</td>
<td>EIEN27</td>
<td>EIEN26</td>
<td>EIEN25</td>
<td>EIEN24</td>
<td></td>
</tr>
<tr>
<td>23:16</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
</tr>
<tr>
<td>EIEN23</td>
<td>EIEN22</td>
<td>EIEN21</td>
<td>EIEN20</td>
<td>EIEN19</td>
<td>EIEN18</td>
<td>EIEN17</td>
<td>EIEN16</td>
<td></td>
</tr>
<tr>
<td>15:8</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
</tr>
<tr>
<td>EIEN15</td>
<td>EIEN14</td>
<td>EIEN13</td>
<td>EIEN12</td>
<td>EIEN11</td>
<td>EIEN10</td>
<td>EIEN9</td>
<td>EIEN8</td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
</tr>
<tr>
<td>EIEN7</td>
<td>EIEN6</td>
<td>EIEN5</td>
<td>EIEN4</td>
<td>EIEN3</td>
<td>EIEN2</td>
<td>EIEN1</td>
<td>EIEN0</td>
<td></td>
</tr>
</tbody>
</table>

**Legend:**

- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as '0'
- **-n** = Value at POR
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- **x** = Bit is unknown

**bit 31-0**

**EIEN31:EIEN0:** Early Interrupt Enable for Analog Input bits

1 = Early Interrupts are enabled for the selected analog input. The interrupt is generated after the early interrupt event occurs (indicated by the EIRDYx bit (*x* = 31-0) of the ADCEISTAT1 register)

0 = Interrupts are disabled

### Register 22-38: ADCEIEN2: ADC Early Interrupt Enable Register 2

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Bit 31/23/15/7</th>
<th>Bit 30/22/14/6</th>
<th>Bit 29/21/13/5</th>
<th>Bit 28/20/12/4</th>
<th>Bit 27/19/11/3</th>
<th>Bit 26/18/10/2</th>
<th>Bit 25/17/9/1</th>
<th>Bit 24/16/8/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
</tr>
<tr>
<td>EIEN63</td>
<td>EIEN62</td>
<td>EIEN61</td>
<td>EIEN60</td>
<td>EIEN59</td>
<td>EIEN58</td>
<td>EIEN57</td>
<td>EIEN56</td>
<td></td>
</tr>
<tr>
<td>23:16</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
</tr>
<tr>
<td>EIEN55</td>
<td>EIEN54</td>
<td>EIEN53</td>
<td>EIEN52</td>
<td>EIEN51</td>
<td>EIEN50</td>
<td>EIEN49</td>
<td>EIEN48</td>
<td></td>
</tr>
<tr>
<td>15:8</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
</tr>
<tr>
<td>EIEN47</td>
<td>EIEN46</td>
<td>EIEN45</td>
<td>EIEN44</td>
<td>EIEN43</td>
<td>EIEN42</td>
<td>EIEN41</td>
<td>EIEN40</td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
</tr>
<tr>
<td>EIEN39</td>
<td>EIEN38</td>
<td>EIEN37</td>
<td>EIEN36</td>
<td>EIEN35</td>
<td>EIEN34</td>
<td>EIEN33</td>
<td>EIEN32</td>
<td></td>
</tr>
</tbody>
</table>

**Legend:**

- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as '0'
- **-n** = Value at POR
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- **x** = Bit is unknown

**bit 31-0**

**EIEN63:EIEN32:** Early Interrupt Enable for Analog Input bits

1 = Early Interrupts are enabled for the selected analog input. The interrupt is generated after the early interrupt event occurs (indicated by the EIRDYx bit (*x* = 63-32) of the ADCEISTAT2 register)

0 = Interrupts are disabled
Section 22. 12-bit High-Speed SAR ADC

Register 22-39: ADCEISTAT1: ADC Early Interrupt Status Register 1

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Bit 31/23/15/7</th>
<th>Bit 30/22/14/6</th>
<th>Bit 29/21/13/5</th>
<th>Bit 28/20/12/4</th>
<th>Bit 27/19/11/3</th>
<th>Bit 26/18/10/2</th>
<th>Bit 25/17/9/1</th>
<th>Bit 24/16/8/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
</tr>
<tr>
<td></td>
<td>EIRDY31</td>
<td>EIRDY30</td>
<td>EIRDY29</td>
<td>EIRDY28</td>
<td>EIRDY27</td>
<td>EIRDY26</td>
<td>EIRDY25</td>
<td>EIRDY24</td>
</tr>
<tr>
<td>23:16</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
</tr>
<tr>
<td></td>
<td>EIRDY23</td>
<td>EIRDY22</td>
<td>EIRDY21</td>
<td>EIRDY20</td>
<td>EIRDY19</td>
<td>EIRDY18</td>
<td>EIRDY17</td>
<td>EIRDY16</td>
</tr>
<tr>
<td>15:8</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
</tr>
<tr>
<td></td>
<td>EIRDY15</td>
<td>EIRDY14</td>
<td>EIRDY13</td>
<td>EIRDY12</td>
<td>EIRDY11</td>
<td>EIRDY10</td>
<td>EIRDY9</td>
<td>EIRDY8</td>
</tr>
<tr>
<td>7:0</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
</tr>
<tr>
<td></td>
<td>EIRDY7</td>
<td>EIRDY6</td>
<td>EIRDY5</td>
<td>EIRDY4</td>
<td>EIRDY3</td>
<td>EIRDY2</td>
<td>EIRDY1</td>
<td>EIRDY0</td>
</tr>
</tbody>
</table>

Legend:  
HS = Hardware Set  
HC = Hardware Cleared  
R = Readable bit  
W = Writable bit  
U = Unimplemented bit, read as ‘0’  
-n = Value at POR  
‘1’ = Bit is set  
‘0’ = Bit is cleared  
x = Bit is unknown

bit 31-0  
**EIRDY31:EIRDY0**: Early Interrupt for Corresponding Analog Input Ready bits  
1 = This bit is set when the early interrupt event occurs for the specified analog input. An interrupt will be generated if early interrupts are enabled in the ADCEIEN1 register. For the Class 1 analog inputs, this bit will set as per the configuration of the ADCEIS<2:0> bits in the ADCxTIME register. For the shared ADC module, this bit will be set as per the configuration of the ADCEIS<2:0> bits in the ADCCON2 register.  
0 = Interrupts are disabled

Register 22-40: ADCEISTAT2: ADC Early Interrupt Status Register 2

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Bit 31/23/15/7</th>
<th>Bit 30/22/14/6</th>
<th>Bit 29/21/13/5</th>
<th>Bit 28/20/12/4</th>
<th>Bit 27/19/11/3</th>
<th>Bit 26/18/10/2</th>
<th>Bit 25/17/9/1</th>
<th>Bit 24/16/8/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
</tr>
<tr>
<td></td>
<td>EIRDY63</td>
<td>EIRDY62</td>
<td>EIRDY61</td>
<td>EIRDY60</td>
<td>EIRDY59</td>
<td>EIRDY58</td>
<td>EIRDY57</td>
<td>EIRDY56</td>
</tr>
<tr>
<td>23:16</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
</tr>
<tr>
<td></td>
<td>EIRDY55</td>
<td>EIRDY54</td>
<td>EIRDY53</td>
<td>EIRDY52</td>
<td>EIRDY51</td>
<td>EIRDY50</td>
<td>EIRDY49</td>
<td>EIRDY48</td>
</tr>
<tr>
<td>15:8</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
</tr>
<tr>
<td></td>
<td>EIRDY47</td>
<td>EIRDY46</td>
<td>EIRDY45</td>
<td>EIRDY44</td>
<td>EIRDY43</td>
<td>EIRDY42</td>
<td>EIRDY41</td>
<td>EIRDY40</td>
</tr>
<tr>
<td>7:0</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
</tr>
<tr>
<td></td>
<td>EIRDY39</td>
<td>EIRDY38</td>
<td>EIRDY37</td>
<td>EIRDY36</td>
<td>EIRDY35</td>
<td>EIRDY34</td>
<td>EIRDY33</td>
<td>EIRDY32</td>
</tr>
</tbody>
</table>

Legend:  
HS = Hardware Set  
HC = Hardware Cleared  
R = Readable bit  
W = Writable bit  
U = Unimplemented bit, read as ‘0’  
-n = Value at POR  
‘1’ = Bit is set  
‘0’ = Bit is cleared  
x = Bit is unknown

bit 31-0  
**EIRDY63:EIRDY32**: Early Interrupt for Corresponding Analog Input Ready bits  
1 = This bit is set when the early interrupt event occurs for the specified analog input. An interrupt will be generated if early interrupts are enabled in the ADCEIEN2 register. For the Class 1 analog inputs, this bit will set as per the configuration of the ADCEIS<2:0> bits in the ADCxTIME register. For the shared ADC module, this bit will be set as per the configuration of the ADCEIS<2:0> bits in the ADCCON2 register.  
0 = Interrupts are disabled
Register 22-41: ADCANCON: ADC Analog Warm-up Control Register

| Bit Range | Bit | Range | Bit | Range | Bit | Range | Bit | Range | Bit | Range | Bit | Range | Bit | Range | Bit | Range | Bit | Range |
|-----------|-----|-------|-----|-------|-----|-------|-----|-------|-----|-------|-----|-------|-----|-------|-----|-------|-----|-------|-----|-------|
| 31:24     | U-0 | U-0   | U-0 | U-0   | R/W-0 | R/W-0 | R/W-0 | R/W-0 |          |          |          |          |          |          |          |          |          |          |
| 23:16     | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |          |          |          |          |          |          |          |          |          |          |
| 15:8      | R-0, HS, HC | R-0, HS, HC | R-0, HS, HC | R-0, HS, HC | R-0, HS, HC | R-0, HS, HC | R-0, HS, HC | R-0, HS, HC |          |          |          |          |          |          |          |          |          |          |
| 7:0       | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |          |          |          |          |          |          |          |          |          |          |

Legend:
- HS = Hardware Set
- HC = Hardware Cleared
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as ‘0’
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- x = Bit is unknown

bit 31-28 Unimplemented: Read as ‘0’

bit 27-24 \textbf{WKUPCLKCNT<3:0>:} Wake-up Clock Count bits

These bits represent the number of ADC clocks required to warm-up the ADC module before it can perform conversion. Although the clocks are specific to each ADC, the WKUPCLKCNT bit is common to all ADC modules.

\[
\begin{align*}
1111 & = 2^{15} = 32,768 \\
1110 & = 2^5 = 16 \\
1010 & = 2^5 = 16 \\
0110 & = 2^4 = 16 \\
0101 & = 2^4 = 16 \\
0100 & = 2^4 = 16 \\
0011 & = 2^3 = 8 \\
0010 & = 2^3 = 8 \\
0010 & = 2^3 = 8 \\
0001 & = 2^2 = 4 \\
0000 & = 2^2 = 4
\end{align*}
\]

bit 23-16 \textbf{WKIEN7:WKIEN0:} ADC Wake-up Interrupt Enable bit\(^{(1)}\)

- 1 = Enable interrupt and generate interrupt when the WKRDYx status bit is set
- 0 = Disable interrupt

bit 15-8 \textbf{WKRDY7:WKRDY0:} ADC Wake-up Status bit\(^{(1)}\)

- 1 = ADC Analog and Bias circuitry ready after the wake-up count number \(2^{\text{WKUPEXP}}\) clocks after setting ANEN\(_x\) to ‘1’
- 0 = ADC Analog and Bias circuitry is not ready

\textbf{Note:} These bits are cleared by hardware when the ANEN\(_x\) bit is cleared

bit 7-0 \textbf{ANEN7:ANEN0:} ADC Analog and Bias Circuitry Enable bits\(^{(1)}\)

- 1 = Analog and bias circuitry enabled. Once the analog and bias circuit is enabled, the ADC module needs a warm-up time, as defined by the WKUPCLKCNT<3:0> bits.
- 0 = Analog and bias circuitry disabled

\textbf{Note 1:} Refer to the “ADC” chapter in the specific device data sheet to determine the available bits for your device.
### Register 22.42: ADCxCFG: ADCx Configuration Register \('x' (\('x' = 0 \text{ through 7})\)

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Bit 31/23/15/7</th>
<th>Bit 30/22/14/6</th>
<th>Bit 29/21/13/5</th>
<th>Bit 28/20/12/4</th>
<th>Bit 27/19/11/3</th>
<th>Bit 26/18/10/2</th>
<th>Bit 25/17/9/1</th>
<th>Bit 24/16/8/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>23:16</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>15:8</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>7:0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
</tbody>
</table>

Legend:
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as '0'
- \(-n\) = Value at POR
- '1' = Bit is set
- '0' = Bit is cleared
- \(x\) = Bit is unknown

**Note:** These bits can only change when the applicable ANENx bit in the ADCANCON register is cleared.
Register 22-43: ADCSYSCFG0: ADC System Configuration Register 0

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Bit 31/23/15/7</th>
<th>Bit 30/22/14/6</th>
<th>Bit 29/21/13/5</th>
<th>Bit 28/20/12/4</th>
<th>Bit 27/19/11/3</th>
<th>Bit 26/18/10/2</th>
<th>Bit 25/17/9/1</th>
<th>Bit 24/16/8/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23:16</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:8</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
</tr>
</tbody>
</table>

Legend:
- HS = Hardware Set
- HC = Hardware Cleared
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as '0'
- -n = Value at POR
- '1' = Bit is set
- '0' = Bit is cleared
- x = Bit is unknown

bit 31-0 AN<31:0>: ADC Analog Input bits
These bits reflect the system configuration and are updated during boot-up time. By reading these read-only bits, the user application can determine whether or not an analog input in the device is available. AN<31:0> reflects the presence or absence of the respective analog input (AN31-AN0).

Register 22-44: ADCSYSCFG1: ADC System Configuration Register 1

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Bit 31/23/15/7</th>
<th>Bit 30/22/14/6</th>
<th>Bit 29/21/13/5</th>
<th>Bit 28/20/12/4</th>
<th>Bit 27/19/11/3</th>
<th>Bit 26/18/10/2</th>
<th>Bit 25/17/9/1</th>
<th>Bit 24/16/8/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23:16</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:8</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
<td>R-0, HS, HC</td>
</tr>
</tbody>
</table>

Legend:
- HS = Hardware Set
- HC = Hardware Cleared
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as '0'
- -n = Value at POR
- '1' = Bit is set
- '0' = Bit is cleared
- x = Bit is unknown

bit 31-0 AN<63:32>: ADC Analog Input bits
These bits reflect the system configuration and are updated during boot-up time. By reading these read-only bits, the user application can determine whether or not an analog input in the device is available. AN<63:32> reflects the presence or absence of the respective analog input (AN63-AN32).
22.3 ADC OPERATION

The High-Speed Successive Approximation Register (SAR) ADC is designed to support power conversion and motor control applications and consists of up to eight individual ADC modules. The dedicated ADC modules have single analog inputs (after alternate selection) connected to their S&H circuit. Since these ADC modules sample a dedicated analog input, they are termed “dedicated” ADC modules. Dedicated ADC modules are used to measure/capture time sensitive or transitory analog signals. The shared ADC module has multiple analog input connected to its S&H circuit through a multiplexer. Since multiple analog input share this ADC, it is termed the “shared” ADC module. The shared ADC module is used to measure analog signals of lower frequencies and signals, which are static in nature (i.e., do not change significantly with time).

The analog inputs that are connected to the dedicated ADC modules are considered Class 1 inputs. The analog inputs connected to the shared ADC module are Class 2 and Class 3 inputs. The number of inputs designated for each “class” depends on the specific device. For example, a device with eight ADC modules and 54 analog inputs will have the following arrangement:

- Class 1 = AN0 to AN6
- Class 2 = AN7 to AN31
- Class 3 = AN32 to AN53

Note: Refer to the “ADC” chapter in the specific device data sheet for the actual number of analog inputs for each class.

The property of each class of analog input is described in Table 22-2:

<table>
<thead>
<tr>
<th>Table 22-2: Analog Input Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC Module</td>
</tr>
<tr>
<td>------------</td>
</tr>
<tr>
<td>Dedicated ADC modules</td>
</tr>
<tr>
<td>Shared ADC module</td>
</tr>
<tr>
<td>Shared ADC module with input scan</td>
</tr>
</tbody>
</table>

Class 1 analog input properties:

- Class 1 inputs are associated with a dedicated ADC module. Each dedicated ADC has a single Class 1 input associated with it at a given time. The (alternate) input selection is made through the SHxALT<1:0> bits in the ADCTRGMODE register. Regardless of the alternate input selection, the trigger source and the result register remains the same.
- Each Class 1 input has a unique trigger (selected by the ADCTRGrx register) and upon arrival of the trigger, ends sampling and starts conversion. Upon completion of conversion, the ADC module reverts back to sampling mode. When a Class 1 input is enabled and is not being converted, it is always sampled.
- All Class 1 inputs have same priority, work independently and it is possible to start conversion on all Class 1 inputs at the same time.
- Class 1 inputs can be part of a scan list, triggered by the common scan trigger source.
Class 2 and Class 3 analog input properties:

- Class 2 inputs are used on the shared ADC module, either individually triggered or as part of a scan list. When used individually they are triggered by their unique trigger selected by the ADCTRGx register.
- The analog inputs on the shared ADC have a natural order of priority (for example, AN7 has a higher priority than AN12)
- Class 3 inputs are used exclusively for scanning and share a common trigger source (scan trigger)
- Since Class 3 analog inputs share both the ADC module and the trigger source, the only method possible to convert them is to scan them sequentially for each incoming scan trigger event, where scanning occurs in the natural order of priority
- Unlike Class 1 analog inputs, the arrival of a trigger in the shared ADC module only starts the sampling. When the trigger arrives, the ADC module goes into sampling mode for the sampling time decided by the SAMC<9:0> bits (ADCCON2<25:16>). At the end of sampling, the ADC starts conversion. Upon completion of conversion, the ADC module is used to convert the next in line Class 2 or Class 3 inputs, according to the natural order of priority. When a shared analog input (Class 2 or Class 3) has completed all conversion and no trigger is pending, the ADC module is disconnected from all analog inputs.
22.3.1 Class 2 Triggering

When a single Class 2 input is triggered, it is sampled and converted by the shared S&H using the sequence illustrated in Figure 22-5. When multiple Class 2 inputs are triggered, it is important to understand the consequences of trigger timing. If a conversion is underway and another Class 2 trigger occurs then the sample-hold-conversion for the new trigger will be stalled until the in-process sample-hold cycle is complete, as shown in the Figure 22-6.

When multiple inputs to the shared S&H are triggered simultaneously, the processing order is determined by their natural priority (the lowest numbered input has the highest priority). As an example, if AN9, AN10, and AN11 are triggered simultaneously, AN9 will be sampled and converted first, followed by AN10 and finally, AN11. When using the independent Class 2 triggering on the shared S&H, the SAMC<9:0> bits (ADCCON2<25:16>) determine the sample time for all inputs while the appropriate TRGSRC<4:0> bits in the ADCTRGx Register (see Register 22-18) determine the trigger source for each input.
22.3.2 Input Scan

Input scanning is a feature that allows an automated scanning sequence of multiple Class 1, Class 2 or Class 3 inputs. All Class 2 and Class 3 inputs are scanned using the single shared S&H. Class 1 inputs are scanned using their dedicated S&H on the dedicated ADC module. The selection of analog inputs for scanning is done with the CSSx bits of the ADCCSS1 and ADCCSS2 registers. Class 1 and Class 2 inputs are triggered using STRIG selection in ADCTRgx register and Class 3 inputs are triggered using the STRGSRC<4:0> of ADCCON1<20:16> register. When a trigger occurs, all Class 1 inputs are captured simultaneously and conversions are started simultaneously. For Class 2 or Class 3 inputs, the sampling and conversion occur in the natural input order is used; lower number inputs are sampled before higher number inputs.

Figure 22-7: Input Scan Conversion Sequence for Three Class 2 Inputs

When using the shared analog inputs in scan mode, the SAMC<9:0> bits in the ADC Control Register 2 (ADCCON2<25:16>) determine the sample time for all inputs while the Scan Trigger Source Selection bits (STRGSRC<4:0>) in the ADC Control Register 1 (ADCCON1<20:16>), determine the trigger source.

To ensure predictable results, a scan should not be retriggered until sampling of all inputs has completed. Care should be taken in the system design to preclude retriggering a scan while a scan is in progress.

Individual Class 2 triggers that occur during a scan will pre-empt the scan sequence if they are a higher priority than the sample currently being processed. In Figure 22-8, a scan of AN11, AN12, and AN13 is underway when an independent trigger of Class 2 input AN8 takes place. The scan is interrupted for the sampling and conversion of AN8.

Figure 22-8: Scan Conversion Pre-empted by Class 2 Input Trigger
22.4 ADC MODULE CONFIGURATION

Operation of the ADC module is directed through bit settings in the specific registers. The following instructions summarize the actions and the settings. The options and details for each configuration step are provided in the subsequent sections.

To configure the ADC module, perform the following steps:

1. Configure the analog port pins, as described in 22.4.1 “Configuring the Analog Port Pins”.
2. Initialize the ADC calibration values by copying them from the factory-programmed DEVADCx Flash registers into the corresponding ADCxCFG registers.
3. Select the analog inputs to the ADC multiplexers, as described in 22.4.2 “Selecting the ADC Multiplexer Analog Inputs”.
4. Select the format of the ADC result, as described in 22.4.3 “Selecting the Format of the ADC Result”.
5. Select the conversion trigger source, as described in 22.4.4 “Selecting the Conversion Trigger Source”.
6. Select the voltage reference source, as described in 22.4.5 “Selecting the Voltage Reference Source”.
7. Select the scanned inputs, as described in 22.4.6 “Selecting the Scanned Inputs”.
8. Select the analog-to-digital conversion clock source and prescaler, as described in 22.4.7 “Selecting the Analog-to-Digital Conversion Clock Source and Prescaler”.
9. Specify any additional acquisition time, if required, as described in 22.10 “ADC Sampling Requirements”.
10. Turn on the ADC module, as described in Equation 22-2: “Sample Time for the Shared ADC Module”.
11. Poll (or wait for the interrupt) for the voltage reference to be ready, as described in 22.4.5 “Selecting the Voltage Reference Source”.
12. Enable the analog and bias circuit for required ADC modules and after the ADC module wakes-up, enable the digital circuit, as described in 22.7.3 “ADC Low-power Mode”.
13. Configure the ADC interrupts (if required), as described in 22.6 “Interrupts”.

22.4.1 Configuring the Analog Port Pins

The ANSElx registers for the I/O ports associated with the analog inputs are used to configure the corresponding pin as an analog or a digital pin. A pin is configured as analog input when the corresponding ANSElx bit = 1. When the ANSElx bit = 0, the pin is set to digital control. The ANSElx registers are set when the device comes out of Reset, causing the ADC input pins to be configured as analog inputs by default.

The TRISx registers control the digital function of the port pins. The port pins that are required as analog inputs must have their corresponding bit set in the specific TRISx register, configuring the pin as an input. If the I/O pin associated with an ADC input is configured as an output by clearing the TRISx bit, the port’s digital output level (VOH or VOL) will be converted. After a device Reset, all of the TRISx bits are set. For more information on port pin configuration, refer to the “I/O Ports” chapter of the specific device data sheet.

Note: When reading a PORT register that shares pins with the ADC, any pin configured as an analog input reads as a ‘0’ when the PORT latch is read. Analog levels on any pin that is defined as a digital input but not configured as an analog input, may cause the input buffer to consume current that exceeds the device specification.
Example 22-1: Initializing and Using ADC Class 1 Input

```c
int main(int argc, char** argv) {
    int result[3];

    /* initialize ADC calibration setting */
    ADC0CFG = DEVADC0;
    ADC1CFG = DEVADC1;
    ADC2CFG = DEVADC2;
    ADC3CFG = DEVADC3;
    ADC4CFG = DEVADC4;
    ADC5CFG = DEVADC5;
    ADC7CFG = DEVADC7;

    /* Configure ADCCON1 */
    ADCCON1 = 0;  // No ADCCON1 features are enabled including: Stop-in-Idle, turbo,
                    // CVD mode, Fractional mode and scan trigger source.

    /* Configure ADCCON2 */
    ADCCON2 = 0;  // Since, we are using only the Class 1 inputs, no setting is
                   // required for ADCDIV

    /* Initialize warm up time register */
    ADCANCON = 0;
    ADCANCONbits.WKUPCLKCNT = 5;  // Wakeup exponent = 32 * TADx

    /* Clock setting */
    ADCCON3 = 0;
    ADCCON3bits.ADCSEL = 0;  // Select input clock source
    ADCCON3bits.CONCLKDIV = 1;  // Control clock frequency is half of input clock
    ADCCON3bits.VREFSEL = 0;  // Select AVDD and AVSS as reference source

    /* Select ADC sample time and conversion clock */
    ADC0TIMEbits.ADCDIV = 1;  // ADC0 clock frequency is half of control clock = TAD0
    ADC0TIMEbits.SAMC = 5;  // ADC0 sampling time = 5 * TAD0
    ADC0TIMEbits.SELRES = 3;  // ADC0 resolution is 12 bits
    ADC1TIMEbits.ADCDIV = 1;  // ADC1 clock frequency is half of control clock = TAD1
    ADC1TIMEbits.SAMC = 5;  // ADC1 sampling time = 5 * TAD1
    ADC1TIMEbits.SELRES = 3;  // ADC1 resolution is 12 bits
    ADC2TIMEbits.ADCDIV = 1;  // ADC2 clock frequency is half of control clock = TAD2
    ADC2TIMEbits.SAMC = 5;  // ADC2 sampling time = 5 * TAD2
    ADC2TIMEbits.SELRES = 3;  // ADC2 resolution is 12 bits

    /* Select analog input for ADC modules, no presync trigger, not sync sampling */
    ADCTRGMODEbits.SH0ALT = 0;  // ADC0 = AN0
    ADCTRGMODEbits.SH1ALT = 0;  // ADC1 = AN1
    ADCTRGMODEbits.SH2ALT = 0;  // ADC2 = AN2

    /* Select ADC input mode */
    ADCTIMCON1bits.SIGN0 = 0;  // unsigned data format
    ADCTIMCON1bits.DIFF0 = 0;  // Single ended mode
    ADCTIMCON1bits.SIGN1 = 0;  // unsigned data format
    ADCTIMCON1bits.DIFF1 = 0;  // Single ended mode
    ADCTIMCON1bits.SIGN2 = 0;  // unsigned data format
    ADCTIMCON1bits.DIFF2 = 0;  // Single ended mode

    /* Configure ADCGIRQENx */
    ADCGIRQEN1 = 0;  // No interrupts are used
    ADCGIRQEN2 = 0;

    /* Configure ADCCSSx */
    ADCCSS1 = 0;  // No scanning is used
    ADCCSS2 = 0;

    /* Configure ADCCMPCONx */
    ADCCMPCON1 = 0;  // No digital comparators are used. Setting the ADCCMPCONx
    ADCCMPCON2 = 0;  // register to '0' ensures that the comparator is disabled.
    ADCCMPCON3 = 0;  // Other registers are “don't care”.
    ADCCMPCON4 = 0;
    ADCCMPCON5 = 0;
    ADCCMPCON6 = 0;
}
/* Configure ADCFLTRx */
ADCFLTR1 = 0; // No oversampling filters are used.
ADCFLTR2 = 0;
ADCFLTR3 = 0;
ADCFLTR4 = 0;
ADCFLTR5 = 0;
ADCFLTR6 = 0;

/* Set up the trigger sources */
ADCTRGSNSbits.LVL0 = 0; // Edge trigger
ADCTRGSNSbits.LVL1 = 0; // Edge trigger
ADCTRGSNSbits.LVL2 = 0; // Edge trigger
ADCTRGIbits.TRGSRC0 = 1; // Set AN0 to trigger from software.
ADCTRGIbits.TRGSRC1 = 1; // Set AN1 to trigger from software.
ADCTRGIbits.TRGSRC2 = 1; // Set AN2 to trigger from software.

/* Early interrupt */
ADCEIEN1 = 0; // No early interrupt
ADCEIEN2 = 0;

/* Turn the ADC on */
ADCCON1bits.ON = 1;

/* Wait for voltage reference to be stable */
while(!ADCCON2bits.BGVRRDY); // Wait until the reference voltage is ready
while(ADCCON2bits.REFFLT); // Wait if there is a fault with the reference voltage

/* Enable clock to analog circuit */
ADCCON1bits.ANEN0 = 1; // Enable the clock to analog bias
ADCCON1bits.ANEN1 = 1; // Enable the clock to analog bias
ADCCON1bits.ANEN2 = 1; // Enable the clock to analog bias

/* Wait for ADC to be ready */
while(!ADCCON1bits.WKRDY0); // Wait until ADC0 is ready
while(!ADCCON1bits.WKRDY1); // Wait until ADC1 is ready
while(!ADCCON1bits.WKRDY2); // Wait until ADC2 is ready

/* Enable the ADC module */
ADCCON3bits.DIGEN0 = 1; // Enable ADC0
ADCCON3bits.DIGEN1 = 1; // Enable ADC1
ADCCON3bits.DIGEN2 = 1; // Enable ADC2

while (1) {
    /* Trigger a conversion */
    ADCCON3bits.GSWTRG = 1;

    /* Wait the conversions to complete */
    while (ADCDSTAT1bits.ARDY0 == 0);
    /* fetch the result */
    result[0] = ADCDATA0;
    while (ADCDSTAT1bits.ARDY1 == 0);
    /* fetch the result */
    result[1] = ADCDATA1;
    while (ADCDSTAT1bits.ARDY2 == 0);
    /* fetch the result */
    result[2] = ADCDATA2;

    /* Process results here */
    * Note: Loop time determines the sampling time since all inputs are Class 1.
    * If the loop time is small and the next trigger happens before the completion
    * of set sample time, the conversion will happen only after the sample time
    * has elapsed.
    */

    return (1);
}
22.4.2 Selecting the ADC Multiplexer Analog Inputs

Each ADC module has two inputs referred to as the positive and negative inputs. Input selection options vary for the dedicated and shared ADC module are described in the following sections.

22.4.2.1 SELECTION OF POSITIVE INPUTS

For dedicated ADC modules, four alternate selection is provided for each positive input. This alternate input can be chosen using the SHxALT<1:0> bits in the ADC Triggering Mode Register (ADCTRGMODE) as follows:

- SH0ALT<1:0> (ADCTRGMODE<17:16>)
- SH1ALT<1:0> (ADCTRGMODE<19:18>)
- SH2ALT<1:0> (ADCTRGMODE<21:20>)
- SH3ALT<1:0> (ADCTRGMODE<23:22>)
- SH4ALT<1:0> (ADCTRGMODE<25:24>)
- SH5ALT<1:0> (ADCTRGMODE<27:26>)
- SH6ALT<1:0> (ADCTRGMODE<29:28>)

Refer to the device data sheet for possible alternate selection of ADC inputs.

For the shared ADC module, the positive input is shared among all Class 2 and Class 3 inputs. Input connection of the analog input ANx to the shared ADC is automatic for either Class 2 input trigger or during a scan of Class 2 and or Class 3 inputs. Selecting inputs for scanning is discussed separately in 22.4.6 “Selecting the Scanned Inputs”.

22.4.2.2 SELECTION OF NEGATIVE INPUTS

Negative input selection is determined by the setting of the DIFFx bit of the ADCIMCONx register. The DIFFx bit allows the inputs to be rail-to-rail, and either single-ended or differential. The SIGNx and DIFFx bits in the ADCIMCONx registers scale the internal ADC analog inputs and reference voltages and configure the digital result to align with the expected full-scale output range.

For the shared ADC module, the analog inputs have individual settings for the DIFFx bit. Therefore, the user has the ability to select certain inputs as single-ended and others as differential, while being connected to the same shared ADC module. While sampling, the signal will change on-the-fly as single-ended or differential, according to its corresponding DIFFx bit setting.

Table 22-3: Negative Input Selection

<table>
<thead>
<tr>
<th>ADCIMCON1</th>
<th>Input Configuration</th>
<th>Input Voltage</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIFFx</td>
<td>SIGNx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Differential 2’s complement</td>
<td>V\text{IN}P - V\text{IN}N = -\text{VREF}</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>Differential unipolar</td>
<td>V\text{IN}P = \text{VREF}</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Single-ended 2’s complement</td>
<td>V\text{IN}P = V\text{IN}N = \text{VREF}</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>Single-ended unipolar</td>
<td>V\text{IN}P = \text{VREF}</td>
</tr>
</tbody>
</table>

Legend: V\text{IN}P = Positive S&H input; V\text{IN}N = Negative S&H input; V\text{REF} = V\text{REFH} - V\text{REFL}

Note: For proper operation and to prevent device damage, input voltage levels should not exceed the limits listed in the “Electrical Characteristics” chapter of the specific device data sheet.
22.4.3 Selecting the Format of the ADC Result

The data in the ADC Result register can be read in any of the four supported data formats. The user can select from unsigned integer, signed integer, unsigned fractional, or signed fractional. Integer data is right-justified and fractional data is left-justified.

- The integer or fractional data format selection is specified globally for all analog inputs using the Fractional Data Output Format bit, FRACT (ADCCON1<23>)
- The signed or unsigned data format selection can be independently specified for each individual analog input using the SIGNx bits in the ADCIMCONx registers.

Table 22-4 provides how ADC result is formatted.

<table>
<thead>
<tr>
<th>FRACT</th>
<th>SIGNx</th>
<th>Description</th>
<th>32-bit Output Data Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Unsigned integer</td>
<td>0000 0000 0000 dddd</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Signed integer</td>
<td>ssss ssss ssss ssss</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Fractional</td>
<td>0000 0000 0000 dddd</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Signed fractional</td>
<td>sddd sddd sddd sddd</td>
</tr>
</tbody>
</table>

**Table 22-4: ADC Result Format Results**
```c
int main(int argc, char** argv) {
    int result[3];

    /* Configure ADCCON1 */
    ADCCON1bits.FRACT = 1; // use Fractional output format
    ADCCON1bits.SELRES = 3; // ADC7 resolution is 12 bits
    ADCCON1bits.STRGSRC = 0; // No scan trigger.

    /* Configure ADCCON2 */
    ADCCON2bits.SAMC = 5; // ADC7 sampling time = 5 * TAD7
    ADCCON2bits.ADCDIV = 1; // ADC7 clock freq is half of control clock = TAD7

    /* Initialize warm up time register */
    ADCANCON = 0;
    ADCANCONbits.WKUPCLKCNT = 5; // Wakeup exponent = 32 * TADx

    /* Clock setting */
    ADCCON3 = 0;
    ADCCON3bits.ADCSEL = 0; // Select input clock source
    ADCCON3bits.CONCLKDIV = 1; // Control clock frequency is half of input clock
    ADCCON3bits.VREFSEL = 0; // Select AVDD and AVSS as reference source

    /* No selection for dedicated ADC modules, no presync trigger, not sync sampling */
    ADCTRGMODEbits = 0;

    /* Select ADC input mode */
    ADCMICON1bits.SIGN7 = 0; // unsigned data format
    ADCMICON1bits.DIFF7 = 0; // Single ended mode
    ADCMICON1bits.SIGN8 = 0; // unsigned data format
    ADCMICON1bits.DIFF8 = 0; // Single ended mode
    ADCMICON1bits.SIGN9 = 0; // unsigned data format
    ADCMICON1bits.DIFF9 = 0; // Single ended mode

    /* Configure ADCGIRQENx */
    ADCGIRQEN1 = 0; // No interrupts are used
    ADCGIRQEN2 = 0;

    /* Configure ADCCSSx */
    ADCCSS1 = 0; // No scanning is used
    ADCCSS2 = 0;

    /* Configure ADCMPCONx */
    ADCMPCON1 = 0; // No digital comparators are used. Setting the ADCMPCONx
    ADCMPCON2 = 0; // register to '0' ensures that the comparator is disabled.
    ADCMPCON3 = 0; // Other registers are "don't care".
    ADCMPCON4 = 0;
    ADCMPCON5 = 0;
    ADCMPCON6 = 0;

    /* Configure ADCFLTRx */
    ADCFLTR1 = 0; // No oversampling filters are used.
    ADCFLTR2 = 0;
    ADCFLTR3 = 0;
    ADCFLTR4 = 0;
    ADCFLTR5 = 0;
    ADCFLTR6 = 0;

    /* Set up the trigger sources */
    ADCTRGSNSbits.LVL7 = 0; // Edge trigger
    ADCTRGSNSbits.LVL8 = 0; // Edge trigger
    ADCTRGSNSbits.LVL9 = 0; // Edge trigger
    ADC1TRG2bits.TRGSRC7 = 1; // Set AN7 to trigger from software
    ADC2TRG3bits.TRGSRC8 = 1; // Set AN8 to trigger from software
    ADC2TRG3bits.TRGSRC9 = 1; // Set AN9 to trigger from software
} ```
/* Early interrupt */
ADCEIEN1 = 0; // No early interrupt
ADCEIEN2 = 0;

/* Turn the ADC on */
ADCCON1bits.ON = 1;

/* Wait for voltage reference to be stable */
while(!ADCCON2bits.BGVRRDY); // Wait until the reference voltage is ready
while(ADCCON2bits.REFFLT); // Wait if there is a fault with the reference voltage

/* Enable clock to analog circuit */
ADCANCONbits.ANEN7 = 1; // Enable the clock to analog bias

/* Wait for ADC to be ready */
while(!ADCANCONbits.WKRDY7); // Wait until ADC7 is ready

/* Enable the ADC module */
ADCCON3bits.DIGEN7 = 1; // Enable ADC7

while (1) {
    /* Trigger a conversion */
    ADCCON3bits.GSWTRG = 1;

    /* Wait the conversions to complete */
    while (ADCSTAT1bits.ARDY7 == 0);
    /* fetch the result */
    result[0] = ADCDATA7;

    while (ADCSTAT1bits.ARDY8 == 0);
    /* fetch the result */
    result[1] = ADCDATA8;

    while (ADCSTAT1bits.ARDY9 == 0);
    /* fetch the result */
    result[2] = ADCDATA9;

    /* Process results here
     * 
     * Note 1: Loop time determines the sampling time since all inputs are Class 2.
     * If the loop time happens is small and the next trigger happens before the
     * completion of set sample time, the conversion will happen only after the
     * sample time has elapsed.
     * 
     * Note 2: Results are in fractional format
     */
    return (1);
}
22.4.4 Selecting the Conversion Trigger Source

Class 1 and Class 2 inputs to the ADC module can be triggered for conversion either individually, or as part of a scan sequence. Class 3 inputs can only be triggered as part of a scan sequence. Individual or scan triggers can originate from an on-board timer or output compare peripheral event, from external digital circuits connected to INT0, from external analog circuits connected to an analog comparator, or through software by setting a trigger bit in a SFR.

22.4.4.1 TRIGGER SELECTION FOR CLASS 1 AND CLASS 2 INPUTS

For each one of the Class 1 and Class 2 inputs, the user application can independently specify a conversion trigger source. The individual trigger source for an analog input ‘x’ is specified by the TRGSRC<4:0> bits located in registers ADCTRG1 through ADCTRG8. Refer to the “ADC” chapter of the specific device data sheet for more information on the available conversion trigger options. For example, these trigger sources may include:

- General Purpose (GP) Timers: When a period match occurs for the 32-bit timer, Timer3/2 or Timer5/4, or the 16-bit Timer1, Timer3 or Timer5, a special ADC trigger event signal is generated by the timer. This feature does not exist for other timers. For more information, refer to Section 14. “Timers” (DS60001105) and the “Timer” chapters in the specific device data sheet.

- Output Compare: The Output Compare peripherals, OC1, OC3, and OC5, can be used to generate an ADC trigger when the output transitions from a low to high state. For more information, refer to Section 16. “Output Compare” (DS60001111), and the “Output Compare” chapter in the specific device data sheet.

- Comparators: The analog Comparators can be used to generate an ADC trigger when the output transitions from a low state to a high state. For more information, refer to Section 19. “Comparator” (DS60001110), and the “Comparator” chapter in the specific device data sheet.

- External INT0 Pin Trigger: In this mode, the ADC module starts a conversion on an active transition on the INT0 pin. The INT0 pin may be programmed for either a rising edge input or a falling edge input to trigger the conversion process.

- Global Software Trigger: The ADC module can be configured for manually triggering a conversion for all inputs that have selected this trigger option. The user can manually trigger a conversion by setting the Global Software Trigger bit, GSWTRG (ADCCON3<6>).

22.4.4.2 PRESYNCHRONIZED TRIGGER AND ASYNCHRONOUS SAMPLING FOR CLASS 1 ANALOG INPUTS

The ADCTRGx register is used for ADC triggers.

- Converted data can be stored in dedicated output registers (ADCDATAx)
- Converted data can be stored in the FIFO
- Converted data can be stored in system RAM using the DMA engine

The ADCTRGMODE register contains the trigger mode for the ADC module determined by the STRGEN bit (presynchronized trigger enable) and the SSAMPEN bit (synchronous sampling also for the first sample after being idle or disabled).

Note 1: The trigger and synchronization works on “control clock (Tq)”. But the sampling time (TSAMCx) is based on TADx clock (which is derived from the Tq signal after passing through a divisor).

Note 2: When Synchronous sampling is used (SSAMPEN = 1), the presynchronized trigger value is ignored (STRGEN = “don’t care”).
22.4.4.2.1 Synchronous Sampling (SSAMPEN = 1) →

After a trigger event, sampling ends and conversion starts exactly \((2 \cdot T_Q + tsAMCX)\) after presynchronized trigger or \(tsAMC\) after the previous conversion, where \(tsAMCX\) is the sample time set by the SAMC<9:0> (ADCxTIME<9:0>) bits.

22.4.4.2.2 Asynchronous Sampling (SSAMPEN = 0) →

After a trigger event, sampling ends and conversion starts immediately (with or without presynchronized), provided that \((2 \cdot T_Q + tsAMCX)\) after pre-synchronized trigger or \(tsAMCX\) after previous conversion, is met and already completed (elapsed). If the time is not met, then the sampling will go on for \(tsAMCX\) time and only then conversion starts. Effectively, this becomes a synchronous sampling, but only when the \(tsAMCX\) requirement is not met before the last asynchronous trigger. If the \(tsAMCX\) requirement is met before the last asynchronous trigger, in fact the last trigger will issue an asynchronous end-of-sampling. It is only the start-of-conversion which is always synchronous. The variable delay between the asynchronous end-of-sampling and the next clock-positive edge-aligned start-of-conversion is bridged by the sampling capacitor which holds the analog voltage value unchanged until it can be converted by the synchronous ADC.

### Table 22-5: Class 1 ADC Triggering

<table>
<thead>
<tr>
<th>STRGEN</th>
<th>SSAMPEN</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>No presynchronized trigger and asynchronous sampling.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Presynchronized trigger and asynchronous sampling.</td>
</tr>
<tr>
<td>x</td>
<td>1</td>
<td>Synchronous sampling.</td>
</tr>
</tbody>
</table>

Figure 22-9 graphically explains the various cases for the STRGEN and SSAMPEN bits. Any trigger is asynchronous by nature and similarly, the trigger for ADC module being asynchronous can arrive any time, with reference to a single \(T_o\) clock. This is depicted by "(1 period jitter)". Once the trigger is received, the "presynchronized trigger" occurs 1 \(T_o\) clock after the actual trigger. Please note that the presynchronized trigger is an internal signal.

Considering the case when both STRGENx and SSAMPENx are ‘0’ (no presynchronized trigger and asynchronous sampling), the asynchronous trigger causes the ADC to switch from Sample mode to Conversion mode. In other words, in such a situation, the presynchronized trigger is ignored (i.e., not used). This is true only if the sample time \(tsAMCX\) is met.

Considering the next case, when STRGENx = 1 and SSAMPENx = 0 (presynchronized trigger and asynchronous sampling), the presynchronized trigger is the signal that causes the ADC to switch from Sample mode to Conversion mode. This condition is true only if the sample time \(tsAMCX\) is met.

For both conditions previously mentioned STRGENx, SSAMPENx = 00 and STRGENx, SSAMPENx = 10, the explained behaviors are true if sample time \(tsAMCX\) is met. In a case of a repeated trigger and when sample time \(tsAMCX\) is met, the explained behavior is graphically depicted in Figure 22-10. The figure shows that the second trigger occurs after the \(tsAMCX\) time is elapsed. Therefore, the second trigger causes the ADC to switch from Sample mode to Convert mode. Please note that, the trigger (first or second trigger) is an asynchronous trigger or a presynchronized trigger, based on the setting of STRGENx, SSAMPENx as ‘00’ or ‘10’.

In case where the \(tsAMCX\) time is not met, the trigger (whether asynchronous or presynchronized) would not cause the switch from sample to convert. Instead, the ADC will wait for \(tsAMCX\) time before switching from sample to convert mode. This is depicted in Figure 22-11. In the figure, the second trigger occurs well before the \(tsAMCX\) time is elapsed. Therefore, this trigger does not cause a start of conversion. Instead, the sample-to-conversion happens only after the \(tsAMCX\) time is complete. In other words, even while setting the ADC to Asynchronous Sampling mode (SSAMPEN = 0), the behavior of the ADC will be synchronous if the sample time \(tsAMCX\) is not met between each trigger.

Returning to Figure 22-9 and considering the case when STRGENx = x and SSAMPENx = 1 (synchronous sampling), the ADC switches from sample to conversion \((2 \cdot T_o + tsAMCX)\) after the presynchronized trigger.
Figure 22-9: Presynchronized Trigger (STRGEN bit) and Synchronized Sampling (SSAMPEN bit)

- **Control Clock (TQ)**
- **Trigger** (1 period jitter)
- **Presynchronized Trigger**
- **Sample ends**
- **Conversion starts**
- **Total Sample Time**
- **ADCxTIME<9:0>**
- **Sample ends, Conversion starts**
- **Sample ends, Conversion starts**
- **Sample ends, Conversion starts**

### '00': STRGENx = 0, SSAMPENx = 0
- Conversion starts after 2 * TQ

### '10': STRGENx = 1, SSAMPENx = 0
- Conversion starts after 2 * TQ

### '11': STRGENx = 0 or 1, SSAMPENx = 1
- Conversion starts immediately after the trigger

- ADCxTIME<9:0>
Figure 22-10: Asynchronous Sampling (SSAMPEN = 0) (When trigger is at rate such that t\text{SAMCx} is met, the ADC allows asynchronous trigger to start conversion)
Class 1 inputs are usually meant to convert analog signals that are fast and transitory in nature. In addition, multiple Class 1 inputs would be required to convert different analog signals that are in phase relation to each other. Examples of such a signal would be 3-phase current signals of a motor.

Considering a case when ADC0, ADC1, and ADC2 are used to sample 3-phase current and all of the ADC modules use STRGENx, SSAMPENx = 00 (no presynchronized trigger and asynchronous sampling), the individual trigger for ADC occurs at slightly different times (due to propagation delay of trigger signal). This is depicted in Figure 22-12. This difference in trigger causes a phase error in the sampled analog signal. To avoid such errors, the presynchronized trigger and asynchronous sampling should be used (i.e., STRGENx, SSAMPENx = 10). The usage of a presynchronized trigger will ensure that all of the ADC modules receive the trigger at exactly the same time.
Since the trigger and synchronization works with the control clock (To) and the ADC modules work on their own clock (Tox), proper synchronization should be maintained between the two clock domains. For proper synchronization, two bits are provided, FSSCLKEN (ADCCON1<10> and FSPBCLKEN (ADCCON1<9>). The usage of these bits are as follows:

- Set the FSSCLKEN bit if one of the following conditions is true:
  - ADCSEL<1:0> = (SYSCLK)
  - ADCSEL<1:0> = (REFCLK3) and the REFCLK3 is also a divided clock derived from the system clock (SYSCLK)
  - ADCSEL<1:0> = (FRC) and the system clock is also set to be driven by FRC
  - ADCSEL<1:0> = PBCLK (for PIC32MZ family)/SYSCLK (for PIC32MK family) and the PBCLK (for PIC32MZ family) clock is a divided clock derived from the system clock

- Set the FSPBCLKEN bit if one of the following conditions is true:
  - ADCSEL<1:0> = SYSCLK and the peripheral clock is a divided clock from the system clock and the ADC is not faster than the peripheral clock. This means that both the peripheral clock and the ADC clock are divided from the same system clock, but the division ratio of the ADC clock is higher than or equal to the division ratio of the peripheral clock, which makes the peripheral clock synchronous to, but not slower than the ADC clock.
  - ADCSEL<1:0> = REFCLOCK3 and REFCLOCK3 is synchronous to, and slower than the peripheral clock, which is true if the peripheral clock is also derived from the REFCLOCK3, but is not slower than the ADC clock
  - ADCSEL<1:0> = FRC and the peripheral clock is also divided from the FRC and is not slower than the ADC clock
  - ADCSEL<1:0> = PBCLK (for PIC32MZ family/SYSCLK (for PIC32MK family)
22.4.4.3 CONVERSION TRIGGER SOURCES AND CONTROL

The following are the possible sources for each trigger signal:

- **External trigger selection through the TRGSRCx<4:0> bits in the ADCTRGx registers.** This capability is supported only for class 1 and class 2 analog inputs. Typically, the user specifies a particular trigger source to initiate a conversion for specific input. All of the analog inputs may select the same trigger source if desired. In such an event, the result will resemble a “scanned conversion”, which will have its order of completion enforced by the priority of the inputs associated with the same trigger source. The first trigger selection is ‘00000’ (no trigger), which amounts to temporarily disabling that particular trigger and consequently, temporarily disabling that analog input from being converted. The next two selections for trigger source (GSWTRG and GLSWTRG) are software generated trigger sources. The second software generated trigger selection is the Global Software Trigger (GSWTRG). This trigger links to the GSWTRG bit in the ADCCON3 register, which may be used to enable the user application to initiate a single conversion. Since GSWTRG is a self-clearing bit, it clears itself on the next ADC clock cycle after being set by the user application. The third software generated trigger selection is the Global Level Software Trigger (GLSWTRG), which is linked to the GLSWTRG bit in the ADCCON3 register. This trigger may be used by the user application to initiate a burst of consecutive samples, as the GLSWTRG bit is not self-clearing. The fourth trigger selection is a special selection, the Scan Trigger selection, which allows the Class 1 and Class 2 analog inputs to be included as members of a global scan of all inputs. The remaining trigger selections 5 to 31 are device dependent. Refer to the specific device data sheet for more information.

- **Scanned trigger selection via the STRGSRC<4:0> bits in the ADCCON1 register and select bits in the ADCCSSx registers.** This mode is typically used to initiate the conversion of a group of analog inputs. This capability works for Class 1, 2, and 3 analog inputs, but is typically used for Class 3 inputs because they do not have individual associated TRGSRC bits. One of the trigger selections is the GSWTRG bit in the ADCCON3 register, which may be used to enable the user software to initiate a conversion.

- **User initiated trigger via the ADINSEL<5:0> bits and the RQCNVRT bit in the ADCCON3 register.** This mode enables the user application to create an individual conversion trigger request for a specified analog input. Using this mode enables the user application to trigger the conversion of an input without changing the trigger source configuration of the ADC. This is useful in handling error situations where another software module wants ADC information without disrupting the normal operation of the ADC. This is also the preferred method to generate the initial trigger to start an digital filter sequence.

- **User controlled sampling of Class 2 and Class 3 inputs through the ADINSEL<5:0> bits and the SAM bit in the ADCCON3 register.** Setting the SAM bit causes the Class 2 and Class 3 inputs to be in Sampling mode, while ignoring the selection of the SAMC<9:0> bits. This mode is also useful in software conversion of ADC with software selectable sample time.

- **External module (such as PTG) may specify an analog input for conversion through the setting of ECRRIEN bit in the ADCCON2 register.** This method operates independently of the normal TRGSRC and STRGSRC methods. External modules may still use individual trigger signals and initiate conversions through the normal TRGSRC and STRGSRC methods.
22.4.4.4 USER-REQUESTED INDIVIDUAL CONVERSION TRIGGER
(SOFTWARE ADC CONVERSION) (ONLY FOR CLASS 2 AND CLASS 3 INPUTS)

The user can explicitly request a single conversion (by software) of any selected analog input at any time during program execution, without changing the trigger source configuration of the ADC. The steps to be followed for conversion are as follows:

- The analog input ID to be converted is specified by the ADC Input Select bits, ADINSEL<5:0> (ADCCON3<5:0>).
- The sampling of analog input is started by setting the SAMP bit (ADCCON3<9>)
- After the required sampling time (time delay), the SAMP bit is cleared
- The conversion of sampled signal is started by setting the RQCNVRT bit (ADCCON3<8>)
- Once the conversion is complete, the ARDYx bit of the ADCSTATx register will be set. The data can be read from the ADCDATAx register.

Figure 22-13 illustrates the conversion process in graphical form.

![Individual Conversion Trigger Process Diagram]
22.4.5 Selecting the Voltage Reference Source

The user application can select the voltage reference for the ADC module, which can be internal or external. The Voltage Reference Input Selection bits, VREFSEL<2:0> (ADCCON3<15:13>), select the voltage reference for analog-to-digital conversions. The upper voltage reference (VREFH) and the lower voltage reference (VREFL) may be the internal AVDD and AVSS voltage rails or the band gap reference generator or the external VREF+ and VREF- input pins. When the voltage reference and band gap reference are ready, the BGVRRDY (ADCCON2<31>) bit is set. If a Fault occurs in the voltage reference (such as a brown-out), the REFFLT bit (ADCCON2<30>) is set. The BGVRRDY and REFFLT bits can also generate interrupts if the BGVRIEN bit (ADCCON2<15>) and REFFLTIEN bit (ADCCON2<14>) are set, respectively.

The voltages applied to the external reference pins must comply with certain specifications. Refer to the “Electrical Characteristics” chapter in the specific device data sheet for the electrical specifications.

The Analog Input Charge Pump Enable bit, AICPMPEN (ADCCON1<12>), should be set when the difference between the selected reference voltages (VREFH - VREFL) is less than 0.65 * (AVDD - AVSS). Setting this bit does not increase the magnitude of reference voltage; however, setting this bit reduces the series source resistance to the sampling capacitors. This maximizes the SNR for analog-to-digital conversions using small reference voltage rails.

22.4.6 Selecting the Scanned Inputs

All available analog inputs can be configured for scanning. Class 1 inputs are sampled using their dedicated ADC module. Class 2 and Class 3 inputs are sampled using the shared ADC module.

A single conversion trigger source is selected for all of the inputs selected for scanning using the STRGSRC<4:0> bits (ADCCON1<20:16>). On each conversion trigger, the ADC module starts converting in the natural priority all inputs specified in the user-specified scan list (ADCCSS1 or ADCCSS2). For Class 1 inputs, sampling ends at the time of the trigger and conversion for all of them begin in parallel. For Class 2 and Class 3 inputs, the trigger initiates a sequential sample/conversion process in the natural priority order.

An analog input belongs to the scan if it is:

• A Class 3 input. For Class 3 inputs, scan is the only mechanism for conversion.
• A Class 1 or a Class 2 input, which has the scan trigger selected as the trigger source, by selecting the STRIG option in the TRGSRCx<4:0> bits located in the ADCTRG1 through ADCTRG8 registers.

The trigger options available for scan are identical to those available for independent triggering of Class 1 and Class 2 inputs. Any Class 1 or Class 2 inputs that are part of the scan must have the STRIG option selected as their trigger source in the TRGSRCx<4:0> bits.

Note 1: Since the clock divisor, resolution and sampling time for each dedicated ADC module (SELRES, ADCDIV, and SAMC<9:0> bits in the ADCxTIME register), and the shared ADC module (SELRES in the ADCCON1 register, ADCDIV and SAMC<9:0> bits in the ADCCON2 register) are handled through separate registers, if a scan sequence is to be set involving both dedicated and shared inputs, each of these registers must be initialized individually, even in scan mode.

2: The end-of-scan (EOS) is generated only if all Class 1 inputs have completed the scan, and the last shared input conversion has completed. Until both of these conditions are met, the scan sequence is still in effect. Therefore, the EOS Interrupt can be used for any scan sequence, with any combination of input types.

Note 1: The external VREF+ and VREF- pins can be shared with other analog peripherals. Refer to the “Pin Tables” section of the specific device data sheet for more information. In addition, the ANSELx bits for the VREF+ and VREF- pins must be set to Analog mode.

2: The Band Gap reference source is not available on all devices. Please refer to the specific device data sheet to determine availability of this feature.
Example 22-3: ADC Scanning Multiple Inputs

```c
int main(int argc, char** argv) {
    int result[3];

    /* Configure ADCCON1 */
    ADCCON1 = 0; // No ADCCON1 features are enabled including: Stop-in-Idle, turbo,
                  // CVD mode, Fractional mode and scan trigger source.
    ADCCON1bits.SELRES = 3; // ADC7 resolution is 12 bits
    ADCCON1bits.STRGSRC = 1; // Select scan trigger.

    /* Configure ADCCON2 */
    ADCCON2bits.SAMC = 5; // ADC7 sampling time = 5 * TAD7
    ADCCON2bits.ADCDIV = 1; // ADC7 clock freq is half of control clock = TAD7

    /* Initialize warm up time register */
    ADCANCON = 0;
    ADCANCONbits.WKUPCLKCNT = 5; // Wakeup exponent = 32 * TADx

    /* Clock setting */
    ADCCON3bits.ADCSEL = 0; // Select input clock source
    ADCCON3bits.CONCLKDIV = 1; // Control clock frequency is half of input clock
    ADCCON3bits.VREFSEL = 0; // Select AVDD and AVSS as reference source
    ADCOTIMEbits.ADCDIV = 1; // ADC0 clock frequency is half of control clock = TAD0
    ADCOTIMEbits.SAMC = 5; // ADC0 sampling time = 5 * TAD0
    ADCOTIMEbits.SELRES = 3; // ADC0 resolution is 12 bits

    /* Select analog input for ADC modules, no presync trigger, not sync sampling */
    ADCTRGMODEbits.SH0ALT = 0; // ADC0 = AN0

    /* Select ADC input mode */
    ADCIMCON1bits.SIGN0 = 0; // unsigned data format
    ADCIMCON1bits.DIFF0 = 0; // Single ended mode
    ADCIMCON1bits.SIGN8 = 0; // unsigned data format
    ADCIMCON1bits.DIFF8 = 0; // Single ended mode
    ADCIMCON1bits.SIGN40 = 0; // unsigned data format
    ADCIMCON1bits.DIFF40 = 0; // Single ended mode

    /* Configure ADCGIRQENx */
    ADCGIRQEN1 = 0; // No interrupts are used.
    ADCGIRQEN2 = 0;

    /* Configure ADCCSSx */
    ADCCSS1 = 0; // Clear all bits
    ADCCSS2 = 0;
    ADCCSS1bits.CSS0 = 1; // AN0 (Class 1) set for scan
    ADCCSS1bits.CSS8 = 1; // AN8 (Class 2) set for scan
    ADCCSS2bits.CSS40 = 1; // AN40 (Class 3) set for scan

    /* Configure ADCCMPCONx */
    ADCMPCON1 = 0; // No digital comparators are used. Setting the ADCMPCONx
    ADCMPCON2 = 0; // register to '0' ensures that the comparator is disabled.
    ADCMPCON3 = 0; // Other registers are 'don't care'.
    ADCMPCON4 = 0;
    ADCMPCON5 = 0;
    ADCMPCON6 = 0;

    /* Configure ADCFLTRx */
    ADCFLTR1 = 0; // No oversampling filters are used.
    ADCFLTR2 = 0;
    ADCFLTR3 = 0;
    ADCFLTR4 = 0;
    ADCFLTR5 = 0;
    ADCFLTR6 = 0;
}
```
Example 22-3: ADC Scanning Multiple Inputs (Continued)

```c
/* Set up the trigger sources */
ADCTRG1bits.TRGSRC0 = 3; // Set AN0 (Class 1) to trigger from scan source
ADCTRG3bits.TRGSRC8 = 3; // Set AN8 (Class 2) to trigger from scan source

/* AN40 (Class 3) always uses scan trigger source */

/* Early interrupt */
ADCEIEN1 = 0; // No early interrupt
ADCEIEN2 = 0;

/* Turn the ADC on */
ADCCON1bits.ON = 1;

/* Wait for voltage reference to be stable */
while(!ADCCON2bits.BGVRRDY); // Wait until the reference voltage is ready
while(ADCCON2bits.REFFLT); // Wait if there is a fault with the reference voltage

/* Enable clock to analog circuit */
ADCANCONbits.ANEN0 = 1; // Enable the clock to analog bias ADC0
ADCANCONbits.ANEN7 = 1; // Enable, ADC7

/* Wait for ADC to be ready */
while(!ADCCON3bits.WKRDY0); // Wait until ADC0 is ready
while(!ADCCON3bits.WKRDY7); // Wait until ADC7 is ready

/* Enable the ADC module */
ADCCON3bits.DIGEN0 = 1; // Enable ADC0
ADCCON3bits.DIGEN7 = 1; // Enable ADC7

while (1) {
    /* Trigger a conversion */
    ADCCON3bits.GSWTRG = 1;

    /* Wait the conversions to complete */
    while (ADCDSTAT1bits.ARDY0 == 0);
    /* fetch the result */
    result[0] = ADCDATA0;

    while (ADCDSTAT1bits.ARDY8 == 0);
    /* fetch the result */
    result[1] = ADCDATA8;

    while (ADCDSTAT2bits.ARDY40 == 0);
    /* fetch the result */
    result[2] = ADCDATA40;

    /*
    * Process results here
    *
    */
}
return (1);
```

22.4.7 Selecting the Analog-to-Digital Conversion Clock Source and Prescaler

The ADC module can use the internal Fast RC (FRC) oscillator output, system clock (SYSCLK), reference clock (REFCLK3) or peripheral bus clock (PBCLK) as the conversion clock source (To). Refer to the “ADC” chapter in the specific device data sheet to determine which reference clock output source may be available.

When the ADCSEL<1:0> bits (ADCCON2<31:30>) = 01, the internal FRC oscillator is used as the ADC clock source. When using the internal FRC oscillator, the ADC module can continue to function in Sleep and Idle modes.

**Note:** It is recommended that applications that require precise timing of ADC acquisitions use SYSCLK as the clock source for the ADC.

For correct analog-to-digital conversions, the conversion clock limits must not be exceeded. Clock frequencies from 1 MHz to 28 MHz are supported by the ADC module.

The maximum rate at which analog-to-digital conversions may be completed by the ADC module (effective conversion throughput) is >3 Msps. However, the maximum rate at which a single input can be converted is dependent on the sampling time requirements. In addition, the sampling time depends on the output impedance of the analog signal source. More information on sampling time is provided in the section 22.10 “ADC Sampling Requirements”.

The ADC clock structure is designed to provide separate and independent clocks to dedicated ADC modules and to the shared ADC module. The input clock source for the ADC is selected using the ADCSEL<1:0> bits (ADCCON3<31:30>). The input clock is further divided by the control clock divider CONCLKDIV<5:0> bits (ADCCON3<29:24>). The output clock is called the “ADC control clock” with a time period of To.

The ADC control clock is divided by the ADCDIV<6:0> bits (ADCTIME<22:16>). This acts as the clock source for the respective dedicated ADC modules with a time period of TADx.

The ADC control clock is divided before it is used for the shared ADC by the ADCDIV<6:0> bits (ADCCON2<6:0>). The time period for this clock is denoted as TAD7.

**Figure 22-14: Clock Derivation for Dedicated (Class 1) and Shared ADC Modules**

Note: Refer to the “ADC” chapter in the specific device data sheet to determine the actual number of dedicated and shared ADC modules.
22.4.8 Sample and Conversion Time

Equation 22-1: Sample Time for Dedicated ADC Modules

\[ t_{SAMC} = ((ADCxTIME<9:0> + 2) \times T_{AD}) \]

\[ t_{conversion} = ((Number \ of \ Bits \ Resolution + 1) \times T_{AD}) \]

Equation 22-2: Sample Time for the Shared ADC Module

\[ t_{SAMC} = ((ADCCON2<25:16> + 2) \times T_{AD}) \]

\[ t_{conversion} = ((Number \ of \ Bits \ Resolution + 1) \times T_{AD}) \]

ADC Scan Throughput rate = \( \frac{1}{(t_{SAMC} + t_{conversion}) \times \#ADC \ ANx \ Scan \ inputs \ used)} \)

22.4.9 Turning ON the ADC

Turning ON the ADC module involves these steps:

When the ADC module enable bit, ON (ADCCON1<15>), is set to '1', the module is in Active mode and is fully powered and functional. When the ON bit is '0', the ADC module is disabled. Once disabled, the digital and analog portions of the ADC are turned off for maximum current savings. In addition to setting the ON bit, the analog and digital circuits of ADC should be turned ON. More details are provided in the Section 22.7.3 “ADC Low-power Mode”.

Note: Writing to the ADC control bits that control the ADC clock, input assignments, scanning, voltage reference selection, S&H circuit operating modes, and interrupt configuration is not recommended while the ADC module is enabled.

22.4.10 ADC Status Bits

The ADC module includes the WKRDYx/WKRDY7 status bit in the ADCANCON register, which indicates the current state of ADC Analog and bias circuit. The user application should not perform any ADC operations until this bit is set.
22.5 ADDITIONAL ADC FUNCTIONS

This section describes some additional features of the ADC module, which includes:

- Digital comparator
- Oversampling filter
- Turbo mode
- CVD mode

22.5.1 Digital Comparator

The ADC module features digital comparators that can be used to monitor selected analog input conversion results and generate interrupts when a conversion result is within the user-specified limits. Conversion triggers are still required to initiate conversions. The comparison occurs automatically once the conversion is complete. This feature is enabled by setting the Digital Comparator Module Enable bit, ENDCMP (ADCCMPCONx<7>).

The user application makes use of an interrupt that is generated when the analog-to-digital conversion result is higher or lower than the specified high and low limit values in the ADCCMPx register. The high and low limit values are specified in the DCMPHI<15:0> bits (ADCCMPx<31:16>) and the DCMPLO<15:0> bits (ADCCMPx<15:0>).

The CMPEx bits (’x’ = 0 through 31) in the ADCCMPENx registers are used to specify which analog inputs are monitored by the digital comparator (for the first 32 analog inputs, ANx, where ’x’ = 0 through 31). The ADCCMPCONx register specifies the comparison conditions that will generate an interrupt, as follows:

- When IEBTWN = 1, interrupt is generated when DCMPLO ≤ ADCDATA < DCMPHI
- When IEHIHI = 1, interrupt is generated when DCMPHI ≤ ADCDATA
- When IEHILO = 1, interrupt is generated when ADCDATA < DCMPHI
- When IEOLOH = 1, interrupt is generated when DCMPLO ≤ ADCDATA
- When IEOLOL = 1, interrupt is generated when ADCDATA < DCMPLO

The comparator event generation is illustrated in Figure 22-15. When the ADC module generates a conversion result, the conversion result is provided to the comparator. The comparator uses the DIFFx and SIGNx bits of the ADCIMCONx register (depending on the analog input used) to determine the data format used and appropriately select whether the comparison should be signed or unsigned. The global ADC setting, which is specified by the FRACT bit (ADCCON1<23>), is also used to set the fractional or integer format. The digital comparator compares the ADC result with the high and low limit values (depending on the selected comparison criteria) in the ADCCMPx register.

Depending on the comparator results, a digital comparator interrupt event may be generated. If a comparator event occurs, the Digital Comparator Interrupt Event Detected status bit, DCMPED (ADCCMPCONx<5>), is set, and the Analog Input Identification (ID) bits, AINID<4:0> (ADCCMPCONx<12:8>), are automatically updated so that the user application knows which analog input generated the interrupt event.

Note 1: The Digital Comparator module supports only the first 32 analog inputs (AN0 through AN31).

2: The user software must format the values contained in the ADCCMPx registers to match converted data format as either signed or unsigned, and fractional or integer.
Figure 22-15: Digital Comparator

Interrupt Generation Logic For Digital Comparator

ADCDATAx

ADCDATAx

ADCDATAx

IELOLO

DCMPLO

DCMPLO

DCMPLO

IELOHI

ADCDATAx

ADCDATAx

IEHILO

DCMPHI

DCMPHI

IEHIHI

DCMPHI

IEBTWN

ENDCMP
Example 22-4: ADC Digital Comparator

```c
int main(int argc, char** argv) {
    int result = 0, eventFlag = 0;

    /* Configure ADCCON1 */
    ADCCON1 = 0; // No ADCCON1 features are enabled including: Stop-in-Idle,
                  // turbo, CVD mode, Fractional mode and scan trigger source.
    ADCCON1bits.SELRES = 3; // ADC resolution is 12 bits
    ADCCON1bits.STRGSRC = 0; // No scan trigger.

    /* Configure ADCCON2 */
    ADCCON2bits.SAMC = 5; // ADC7 sampling time = 5 * TAD7
    ADCCON2bits.ACDCDIV = 1; // ADC7 clock freq = TAD7

    /* Initialize warm up time register */
    ADCANCON = 0;
    ADCANCONbits.WKUPCLKCNT = 5; // Wakeup exponent = 32 * TADx

    /* Clock setting */
    ADCCON3 = 0;
    ADCCON3bits.ADCSEL = 0; // Select input clock source
    ADCCON3bits.CONCLKDIV = 1; // Control clock frequency is half of input clock
    ADCCON3bits.VREFSEL = 0; // Select AVDD and AVSS as reference source

    /* No selection for dedicated ADC modules, no presync trigger, not sync sampling */
    ADCTRGMODEbits = 0;

    /* Select ADC input mode */
    ADCIMCON1bits.SIGN8 = 0; // unsigned data format
    ADCIMCON1bits.DIFF8 = 0; // Single ended mode

    /* Configure ADCGIRQENx */
    ADCGIRQEN1 = 0; // No interrupts are used
    ADCGIRQEN2 = 0;

    /* Configure ADCCSSx */
    ADCCSS1 = 0; // No scanning is used
    ADCCSS2 = 0;

    /* Configure ADCCMPCONx */
    ADCCMP1 = 0; // Clear the register
    ADCCMP1bits.DCMPHI = 0xC00; // High limit is a 3072 result.
    ADCCMP1bits.DCMFL0 = 0x500; // Low limit is a 1280 result.
    ADCCMP1bits.IEBTWN = 1; // Create an event when the measured result is
                            // >= low limits and < high limit.
    ADCCMPEN1 = 0; // Clear all enable bits
    ADCCMP1bits.CMPE8 = 1; // set the bit corresponding to AN8
    ADCCMP1bits.ENDCMP = 1; // enable comparator
    ADCCMPCON2 = 0;
    ADCCMPCON3 = 0;
    ADCCMPCON4 = 0;
    ADCCMPCON5 = 0;
    ADCCMPCON6 = 0;

    /* Configure ADCFLTRx */
    ADCFLTR1 = 0; // No oversampling filters are used.
    ADCFLTR2 = 0;
    ADCFLTR3 = 0;
    ADCFLTR4 = 0;
    ADCFLTR5 = 0;
    ADCFLTR6 = 0;
}```
Example 22-4: ADC Digital Comparator (Continued)

```c
/* Set up the trigger sources */
ADCTRGR3bits.TRGSRC8 = 3; // Set AN8 (Class 2) to trigger from scan source

/* Early interrupt */
ADCEIEN1 = 0; // No early interrupt
ADCEIEN2 = 0;

/* Turn the ADC on */
ADCCON1bits.ON = 1;

/* Wait for voltage reference to be stable */
while(!ADCCON2bits.BGVRRDY); // Wait until the reference voltage is ready
while(ADCCON2bits.REFFLT); // Wait if there is a fault with the reference voltage

/* Enable clock to analog circuit */
ADCCANCONbits.ANEN7 = 1; // Enable the clock to analog bias

/* Wait for ADC to be ready */
while(!ADCCANCONbits.WKRDY7); // Wait until ADC7 is ready

/* Enable the ADC module */
ADCCON3bits.DIGEN7 = 1; // Enable ADC7

while (1) {
    /* Trigger a conversion */
    ADCCON3bits.GSWTRG = 1;

    while (ADCDSTAT1bits.ARDY8 == 0);
    /* fetch the result */
    result = ADCDATA8;

    /* Note: It is not necessary to fetch the result for the digital 
    * comparator to work. In this example we are triggering from 
    * software so we are using the ARDY8 to gate our loop. Reading the 
    * data clears the ARDY bit. 
    */
    /* See if we have a comparator event*/
    if (ADCCMPCON1bits.DCMPED == 1) {
        eventFlag = 1;
        /* 
        * Process results here 
        */
        }
    return (1);
}
```
22.5.2 Oversampling Digital Filter

The ADC module supports up to six oversampling digital filters. The oversampling digital filter consists of an accumulator and a decimator (down-sampler), which function together as a low-pass filter. By sampling an analog input at a higher-than-required sample rate, and then processing the data through the oversampling digital filter, the effective resolution of the ADC module can be increased at the expense of decreased conversion throughput.

To obtain ‘x’ bits of extra resolution, number of samples required (over and above the Nyquist rate) = \((2^x)^2\):
- 4x oversampling yields one extra bit of resolution (total 13 bits resolution)
- 16x oversampling yields two extra bits of resolution (total 14 bits resolution)
- 64x oversampling provides three extra bits of resolution (total 15 bits resolution)
- 256x oversampling provides four extra bits of resolution (total 16 bits resolution)

The digital filter also has an averaging mode, where it accumulates the samples and divides it by the number of samples.

**Note 1:** Only Class 1 and Class 2 analog inputs can engage the digital filter. Therefore, the CHNLID<4:0> bits are 5 bits wide (0 to 31).

2: During the burst conversion process (repeated trigger until all required data for oversampling is obtained) in the case of filtering Class 2 input using the shared ADC module, higher priority ADC inputs may still process conversions; lower priority ADC conversion requests are held waiting until the filter burst sequence has been completed.

3: If higher priority requests occur during the digital filter sequence, they delay the completion of the filtering process. This delay may affect the accuracy of the result because the multiple samples will not be contiguous. The user should arrange the initiation trigger for the oversampling filters to occur while there are no expected interruptions from higher priority ADC conversion requests.

The user application should configure the following bits to perform an oversampling conversion:
- Select the amount of oversampling through the Oversampling Filter Oversampling Ratio (OVRSAM<2:0>) bits in the ADC Filter register (ADCFLTR<28:26>)
- Set the filter mode to either Oversampling mode or Averaging mode using the DFMODE bit (ADCFLTR<29>)
- If the filter is set to Averaging mode and the data format is set to fractional (FRACT bit), set or clear the DATA16EN bit (ADCFLTR<30>) to set the output resolution
- Set the sample time for subsequent samples:
  - If using Class 1 inputs, select the sample time of the recurring conversions using the SAMC<9:0> bits ADCxTIME<9:0>)
  - If using Class 2 inputs, select the sample time using the SAMC<9:0> bits (ADCCON2<25:16>)
- Select the specific analog input to be oversampled by configuring the Analog Input ID Selection bits, CHNLID<4:0> (ADCFLTR<20:16>)
- If needed, include the oversampling filter interrupt event in the global ADC interrupt, by setting the Accumulator Filter Global Interrupt Enable bit, AFGIEN (ADCFLTR<25>)
- Enable the oversampling filter by setting the Oversampling Filter Accumulator Enable bit, AFEN (ADCFLTR<31>)

Once the digital filter module is configured, the filter’s control logic waits for an external trigger to initiate the process. The trigger signal for the analog input to be oversampled causes the accumulator to be cleared and initiates the first conversion. The trigger also force the trigger sensitivity into level mode and forces the trigger itself to 1 as long as the filter needs to acquire the user specified number of samples via the OVRSAM<2:0> bits (ADCFLTR<28:26>). The time delay between each acquired sample is decided by the set sample time (SAMC<9:0> bits in the ADCxTIME register for Class 1 or the SAMC<9:0> bits in the ADCCON2 register for Class 2) and the time for conversion. When the required number set by OVRSAM<2:0> have been received and processed, the data stored in the FLTRDATA<15:0> bit (ADCFLTR<15:0>) and the AFRDY bit (ADCFLTR<24>) is set, and the interrupt is generated (if enabled).
Figure 22-16 illustrates 4x oversampling on a Class 1 input. Prior to the trigger, the Class 1 input is sampling the input signal. The trigger starts the first conversion. Once the conversion is complete, the ADC goes into sampling mode. When the sampling time expires, a new conversion sequence occurs. After each sample is converted it is added to the accumulator. The sequence repeats until the number of samples specified by the OVRSAM<2:0> bits have been accumulated. When the last sample has been converted, its value is added to the accumulator. The result is right shifted and then stored in the FLTRDATA<15:0> bits.

Figure 22-16: 4x Oversampling of a Class 1 Input

If multiple Class 1 inputs use filtering, they all operate in parallel and independent from each other.

Figure 22-17 illustrates 4x Oversampling using a Class 2 input. Triggering a Class 2 input initiates sampling for the length of time defined by the SAMC<9:0> bits. Retriggers generated by the oversampling logic use the SAMC<9:0> bits, to set the sample time.

Since Class 2 inputs use the shared S&H, oversampling blocks lower priority Class 2 and Class 3 triggers. Higher priority Class 2 triggers will completely disrupt the oversampling process, and therefore, should be avoided completely. The same priority rule applies to two Class 2 inputs that use two digital filters. In such a case, the higher priority input will also use the shared ADC module in Burst mode and will prevent the lower priority input to use the shared ADC. Only after all required samples are obtained by the higher priority input, the lower priority input can use the shared ADC to acquire samples for its own digital filtering.
Section 22. 12-bit High-Speed SAR ADC

Example 22-5: ADC Digital Oversampling Filter

```c
int main(int argc, char** argv) {
    int result;

    /* Configure ADCCON1 */
    ADCCON1 = 0; // No ADCCON1 features are enabled including: Stop-in-Idle, turbo, CVD mode, Fractional mode and scan trigger source.

    /* Configure ADCCON2 */
    ADCCON2 = 0; // Since, we are using only the Class 1 inputs, no setting is required for ADCDIV

    /* Initialize warm up time register */
    ADCANCON = 0;
    ADCANCONbits.WKUPCLKCNT = 5; // Wake-up exponent = 32 * TADx

    /* Clock setting */
    ADCCON3 = 0;
    ADCCON3bits.ADCSEL = 0; // Select input clock source
    ADCCON3bits.CONCLKDIV = 1; // Control clock frequency is half of input clock
    ADCCON3bits.VREFSEL = 0; // Select AVDD and AVSS as reference source
    ADC0TIMEbits.ADCDIV = 1; // ADC0 clock frequency is half of control clock = TAD0
    ADC0TIMEbits.SAMC = 5; // ADC0 sampling time = 5 * TAD0
    ADC0TIMEbits.SELRES = 3; // ADC0 resolution is 12 bits

    /* Select analog input for ADC modules, no presync trigger, not sync sampling */
    ADCTRGMODEbits.SH0ALT = 0; // ADC0 = AN0

    /* Select ADC input mode */
    ADCIMCON1bits.SIGN0 = 0; // unsigned data format
    ADCIMCON1bits.DIFF0 = 0; // Single ended mode

    /* Configure ADCGIRQENx */
    ADCGIRQEN1 = 0; // No interrupts are used
    ADCGIRQEN2 = 0;

    /* Configure ADCCSSx */
    ADCCSS1 = 0; // No scanning is used
    ADCCSS2 = 0;
}
```
Example 22-5: ADC Digital Oversampling Filter (Continued)

```c
/* Configure ADCCMPCONx */
ADCCMPCON1 = 0; // No digital comparators are used. Setting the ADCCMPCONx
ADCCMPCON2 = 0; // register to '0' ensures that the comparator is disabled.
ADCCMPCON3 = 0; // Other registers are 'don't care'.
ADCCMPCON4 = 0;
ADCCMPCON5 = 0;
ADCCMPCON6 = 0;

/* Configure ADCFLTRx */
ADCFLTR1 = 0; // Clear all bits
ADCFLTR1bits.CHNLID = 0; // Use AN0 as the source
ADCFLTR1bits.OVRSAM = 3; // 16x oversampling
ADCFLTR1bits.DFMODE = 0; // Oversampling mode
ADCFLTR1bits.AFEN = 1; // Enable filter 1
ADCFLTR2 = 0; // Clear all bits
ADCFLTR3 = 0;
ADCFLTR4 = 0;
ADCFLTR5 = 0;
ADCFLTR6 = 0;

/* Set up the trigger sources */
ADCTGSNSbits.LVL0 = 0; // Edge trigger
ADCTRG1bits.TRGSRC0 = 1; // Set AN0 to trigger from software.

/* Turn the ADC on */
ADCCON1bits.ON = 1;

/* Wait for voltage reference to be stable */
while(!ADCCON2bits.BGVRRDY); // Wait until the reference voltage is ready
while(ADCCON2bits.REFFLT); // Wait if there is a fault with the reference voltage

/* Enable clock to analog circuit */
ADCANCONbits.ANEN0 = 1; // Enable the clock to analog bias and digital control

/* Wait for ADC to be ready */
while(!ADCANCONbits.WKRDY0); // Wait until ADC0 is ready

/* Enable the ADC module */
ADCCON3bits.DIGEN0 = 1; // Enable ADC0

while (1) {
    /* Trigger a conversion */
    ADCCON3bits.GSWTRG = 1;

    /* Wait for the oversampling process to complete */
    while (ADCFLTR1bits.AFRDY == 0);
    /* fetch the result */
    result = ADCFLTR1bits.FLTRDATA;

    /* Process result Here */
    * Note 1: Loop time determines the sampling time for the first sample.
    * remaining samples sample time is determined by set sampling + conversion time.
    * Note 2: The first 5 samples may have reduced accuracy.

    */
}
return (1);
```
Section 22. 12-bit High-Speed SAR ADC

Figure 22-18: ADC Filter Comparisons Example

<table>
<thead>
<tr>
<th>DFMODE bit (ADCFLTRx&lt;29&gt;)</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAMC + Conv Period</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Minimum Trigger Period</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRGSRCx&lt;4:0&gt; bits (ADCTRGx&lt;31:0&gt;)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADC_Meas_1A+B+C+D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADC_Meas_2A+B+C+D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADC_Meas_3A+B+C+D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADC_Meas_4A+B+C+D</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

When the DFMODE bit (ADCFLTRx<29>) = 0

Minimum Trigger Period = \((OVRSAM<2:0>\text{ bits (ADCFLTRx<28:26>)) \times TAD + ((SELRES<1:0>\text{ bits (ADCxTIME<25:24>) + 1) \times TAD})\)

Example:

- \(OVRSAM<2:0>\text{ bits (ADCFLTRx<28:26>)} = 4\) Samples
- \(SAMC<7:0>\text{ bits (ADCxTIME<7:0>) = 3 TAD}\)
- \(SELRES<1:0>\text{ bits (ADCxTIME<25:24>) = 12 bits}\)

Minimum Trigger Period = \(64 \times TAD\)

AFRDY bit (ADCFLTRx<24>) = 1

13-bit ADC Result (Interrupt)

AFRDY bit (ADCFLTRx<24>) = 1

13-bit ADC Result (Interrupt)

AFRDY bit (ADCFLTRx<24>) = 1

13-bit ADC Result (Interrupt)

AFRDY bit (ADCFLTRx<24>) = 1

13-bit ADC Result (Interrupt)

AFRDY bit (ADCFLTRx<24>) = 1

12-bit ADC Result (Interrupt)

(AFADMeas_1+2+3+4) / 4
### 22.5.3 FIFO Data Output

The dedicated ADC module has a provision to store the output data into a FIFO. This could be useful while the ADC is used to convert signals at a very high throughput. The ADCFIFO register is the read port for the FIFO. The FIFO is 128 level deep and is circular in nature.

The ADC module that needs to use the FIFO is selected by the ADCxEN bits (ADCFSTAT<30:24>). If more than one ADC module needs to use FIFO, all those respective bits can be set among ADCxEN. While reading the data from FIFO, the ADCID<2:0> bits (ADCFSTAT<2:0>) should be read first and then the data from the ADCFIFO register. This way, the user application knows from which ADC module the data originated from the FIFO. The FEN bit (ADCFSTAT<31>) bit is set to enable the FIFO operation. Once data is available in FIFO, the FRDY bit (ADCFSTAT<22>) is set and remains set until the FIFO is entirely read and is devoid of any data. If required, the FIEN bit (ADCFSTAT<23>) can be set to generate interrupt.

**Example 22-6: ADC FIFO Usage for Class 1 Input**

```c
int main(int argc, char** argv) {
    int result[128], index = 0;

    /* Configure ADCCON1 */
    ADCCON1 = 0; // No ADCCON1 features are enabled including: Stop-in-Idle, turbo,
    // CVD mode, Fractional mode and scan trigger source.

    /* Configure ADCCON2 */
    ADCCON2 = 0; // Since, we are using only the Class 1 inputs, no setting is
    // required for ADCDIV

    /* Initialize warm up time register */
    ADCANCON = 0;
    ADCANCONbits.WKUPCLKCNT = 5; // Wakeup exponent = 32 * TAdx

    /* Clock setting */
    ADCCON3 = 0; // Select input clock source
    ADCCON3bits.ADCSEL = 0;
    ADCCON3bits.CONCLKDIV = 1; // Control clock frequency is half of input clock
    ADCCON3bits.VREFSEL = 0; // Select AVDD and AVSS as reference source
    ADC0TIMEbits.ADCDIV = 1; // ADC0 clock frequency is half of control clock = TAd0
    ADC0TIMEbits.SAMC = 5; // ADC0 sampling time = 5 * TAd0
    ADC0TIMEbits.SELRES = 3; // ADC0 resolution is 12 bits

    /* Select analog input for ADC modules, no presync trigger, not sync sampling */
    ADCTROMODEbits.SH0ALT = 0; // ADC0 = AN0
    ADCTROMODEbits.SH0ALT = 0; // ADC0 = AN0

    /* Select ADC input mode */
    ADCTRGMODEbits.SH0ALT = 0; // unsigned data format
    ADCTRGMODEbits.DIFF0 = 0; // Single ended mode

    /* Configure ADCGIRQENx */
    ADCGIRQEN1 = 0; // No interrupts are used
    ADCGIRQEN2 = 0;

    /* Configure ADCCSSx */
    ADCCSS1 = 0; // No scanning is used
    ADCCSS2 = 0;

    /* Configure ADCCMPCONx */
    ADCCMPCON1 = 0; // No digital comparators are used. Setting the ADCCMPCONx
    ADCCMPCON2 = 0; // register to '0' ensures that the comparator is disabled.
    ADCCMPCON3 = 0;
    ADCCMPCON4 = 0;
    ADCCMPCON5 = 0;
    ADCCMPCON6 = 0;
}
```

**Note 1:** FIFO depth depends on the device. Please refer to the specific device data sheet for details.

**Note 2:** While retrieving data from the FIFO, user code should read both the ADCID<2:0> bits and the ADCFIFO register during each successive reads.
Example 22-6: ADC FIFO Usage for Class 1 Input (Continued)

```c
/* Configure ADCFLTRx */
ADCFLTR1 = 0; // Clear all bits
ADCFLTR2 = 0;
ADCFLTR3 = 0;
ADCFLTR4 = 0;
ADCFLTR5 = 0;
ADCFLTR6 = 0;

/* Set up the trigger sources */
ADCTGNSbits.LVLO = 0; // Edge trigger
ADCTRG0bits.TRGSRC0 = 1; // Set AN0 to trigger from software.

/* Early interrupt */
ADCEIEN1 = 0; // No early interrupt
ADCEIEN2 = 0;

/* Set FIFO */
ADCFSTAT = 0; // Clear all bits
ADCFSTATbits.ADC0EN = 1; // Select ADC0
ADCFSTATbits.FEN = 1; // Enable FIFO

/* Turn the ADC on */
ADCON1bits.ON = 1;

/* Wait for voltage reference to be stable */
while(!ADCCON2bits.BGVRRDY); // Wait until the reference voltage is ready
while((ADCCON2bits.REFFLT)); // Wait if there is a fault with the reference voltage

/* Enable clock to analog circuit */
ADCCON0bits.ANEN0 = 1; // Enable the clock to analog bias and digital control

/* Wait for ADC to be ready */
while(!ADCCON0bits.WKRDY0); // Wait until ADC0 is ready

/* Enable the ADC module */
ADCCON3bits.DIGEN0 = 1; // Enable ADC0

while (1) {
    /* Trigger a conversion */
    ADCCON3bits.GSWTRG = 1;

    /* Wait the conversions to complete and data available in FIFO */
    while((ADCFSTATbits.FRDY == 0));

    /* Once FIFO bit is set, read all possible data, until bit is clear */
    while((ADCFSTATbits.FRDY))
    {
        /* If data overflow occurred in FIFO, break read process */
        if(ADCFSTATbits.FWROVERR)
        {
            break;
        }

        /* if ADC ID is really '0', read data. This is more appropriate when multiple ADC modules are using the FIFO, therefore, reading the ADC ID will identify the ADC and data relation */
        if(ADCFSTATbits.ADCID == 0)
        {
            /* Read data from FIFO */
            result[index] = ADCFIFO;

            /* For each FIFO read, ADCFSTATbits.FCNT will keep reducing */
            index++;

            if(index >= 128)
            {
                index = 0;
            }
        }
    }

    /* Process results here */
    *
    *}
    return (1);
```
22.5.4 Dedicated DMA-based Data Output

Depending on the device, each dedicated ADC module has a DMA engine that can be used to store converted data directly into system memory (RAM). Refer to the “ADC” chapter in the specific device data sheet to determine whether this feature is available for your device.

If the DMACNTEN bit in the ADCDMASTAT register is set, the ADC module current sample count is also written by the DMA engine into system RAM starting at the address specified in the ADCCNTB register.

The size of the buffer used for storage is specified by the DMABL<2:0> bits (ADCCON1<2:0>). Also, the RAM address where the output data should be stored is specified by the DMABADDR<31:0> bits in the ADCDMAB register. Each dedicated ADC module has two buffers: Buffer A and Buffer B.

Since DMABL<2:0> counts the number of data and each data is 16 bits, the actual size of the buffer (in bytes) for each ADC module is given by the formula:

\[ = 2^{(ADCCON1bits\cdot DMABL + 1)} \]

The channel storage buffers spaces are contiguous in the System RAM, starting at the base address (given by the ADCDMAB register) with the ADC0 Buffer A, followed by the ADC0 Buffer B, followed by the ADC1 Buffer A, followed by the ADC1 Buffer B, and so on in the natural order of the channel number assignments.

### Table 22-6: DMA Buffer Format When Used by Multiple ADC Modules

<table>
<thead>
<tr>
<th>ADC Module ID</th>
<th>Buffer</th>
<th>RAM Address Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A</td>
<td>(DMABADDR&lt;31:0&gt;) + 0 to (DMABADDR&lt;31:0&gt;) + 255</td>
</tr>
<tr>
<td>0</td>
<td>B</td>
<td>(DMABADDR&lt;31:0&gt;) + 256 to (DMABADDR&lt;31:0&gt;) + 511</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>(DMABADDR&lt;31:0&gt;) + 512 to (DMABADDR&lt;31:0&gt;) + 767</td>
</tr>
<tr>
<td>1</td>
<td>B</td>
<td>(DMABADDR&lt;31:0&gt;) + 768 to (DMABADDR&lt;31:0&gt;) + 1023</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>(DMABADDR&lt;31:0&gt;) + 1024 to (DMABADDR&lt;31:0&gt;) + 1279</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>(DMABADDR&lt;31:0&gt;) + 1280 to (DMABADDR&lt;31:0&gt;) + 1535</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>6</td>
<td>A</td>
<td>(DMABADDR&lt;31:0&gt;) + 3072 to (DMABADDR&lt;31:0&gt;) + 3327</td>
</tr>
<tr>
<td>6</td>
<td>B</td>
<td>(DMABADDR&lt;31:0&gt;) + 3328 to (DMABADDR&lt;31:0&gt;) + 3583</td>
</tr>
</tbody>
</table>

As shown in Table 22-6, the user can get up to 128 samples (two bytes per sample) for each ADC module until data loss occurs by new data overwriting old data. The DMA engine will wrap around the address for each channel inside its own allocated RAM space.

ADCx, ‘x’ = 0 through 6, will have Buffer A starting at:

\[ = ADCDMAB + (2 \cdot x) \cdot 2^{(ADCCON1bits\cdot DMABL + 1)} \]

ADCx, ‘x’ = 0 through 6, will have Buffer B starting at:

\[ = ADCDMAB + (2 \cdot (x + 1)) \cdot 2^{(ADCCON1bits\cdot DMABL + 1)} \]

The ADCCNTB register contains the user-given address at which (if the DMACNTEN bit (ADCDMASTAT<15>) is set) the DMA engine will start saving the current count of output samples already written to each of the buffers in the System RAM for each ADC module. The ADCx will have its Buffer A current sample count saved at the address ((ADCCNTB) + 2 \cdot x) and its Buffer B current sample count saved at the address ((ADCCNTB) + (2 \cdot x + 1)). Each ADC buffer count takes only 1 byte of RAM storage, because the maximum value is 128. At any time, by reading the memory location specified by ADCCNTB register, the user can know the number of available data for each ADC (0 through 6) and each buffer (A, B); even before the RAFx or RBFx bit is set.
Section 22. 12-bit High-Speed SAR ADC

Table 22-7: DMA Buffer Count Table

<table>
<thead>
<tr>
<th>ADC Module ID</th>
<th>Buffer</th>
<th>Sample Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A</td>
<td>Read memory (specified by ADCCNTB register) to determine the number of samples stored.</td>
</tr>
<tr>
<td>0</td>
<td>B</td>
<td>Read memory (specified by ADCCNTB register) + 1 to determine the number of samples stored.</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>Read memory (specified by ADCCNTB register) + 2 to determine the number of samples stored.</td>
</tr>
<tr>
<td>1</td>
<td>B</td>
<td>Read memory (specified by ADCCNTB register) + 3 to determine the number of samples stored.</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>Read memory (specified by ADCCNTB register) + 4 to determine the number of samples stored.</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>Read memory (specified by ADCCNTB register) + 5 to determine the number of samples stored.</td>
</tr>
<tr>
<td>6</td>
<td>A</td>
<td>Read memory (specified by ADCCNTB register) + 12 to determine the number of samples stored.</td>
</tr>
<tr>
<td>6</td>
<td>B</td>
<td>Read memory (specified by ADCCNTB register) + 13 to determine the number of samples stored.</td>
</tr>
</tbody>
</table>

Immediately after ADC module (0 through 6) has been enabled by the user by setting the DIGEnx bit (ADCCON3 register), the DMA engine will reset to zero in both the ADCx Buffer A sample count at address ((ADCCNTB) + 2 * x) and the Buffer B sample count at address ((ADCCNTB)+(2 * x + 1)).
22.5.4.1 DMA PING-PONG MODE

The DMA engine Ping-Pong mode of operation when used with a single ADC module, is described in the following steps:

1. DMA engine saves converted data into a buffer in system RAM for Buffer A.
2. Once Buffer A is full, the RAFx bit (ADCDMASTAT<6:0>) for the used ADC is set. Also, the DMA engine will start using and saving data into Buffer B.
3. An interrupt will be generated if the RAFIENx bit (ADCDMASTAT<14:8>) is set for the selected ADC.
4. Inside the ISR, the user application must know the buffer ID (either A or B) by reading the RAFx or RBFx bits.
5. Whichever bit is set, the user application must read the converted data from the buffer ID and perform suitable processing on the converted data.
6. While the user application is processing the converted data, the DMA engine is saving data into Buffer B.
7. Once the user application completes the processing of data from Buffer A, it waits for completion of Buffer B.

If an application needs to use the data being stored in a buffer before the buffer is full, the following steps are necessary:

1. To locate the data being saved in buffer, the user code should have enabled DMACNTEN bit (ADCDMASTAT<15> register).
2. The current sample count can be retrieved from (ADCCNTB + (2 * x + 1)) (considering Buffer B being used), and then read the suitable memory offset specified in the ADCDMAB register.
3. Once Buffer B is full, the RBFx bit for the used channel is set. Also, the DMA engine will start saving data into Buffer A.

Note 1: The initialization of the buffer sample count table to all zeros before enabling the DMA engine is left completely as a task for the software. It should be done in order to avoid garbage data in the buffer sample count table before the first sample is being saved in each of the buffers.

Note 2: A read of the ADCDMASTAT register will clear all status bits in this register. Because the dedicated ADC modules work completely independent one of another, more than one status bit can be set at the same time. The user must read the ADCDMASTAT register and save its contents into a separate variable for logic bit-level operations to identify which dedicated ADC module buffers are full at that time.

Figure 22-19 illustrates Ping-Pong mode with continuous operation.
Section 22. 12-bit High-Speed SAR ADC

Figure 22-19: Repetitive Data Transfer in Ping-Pong Mode

DMABADDR<31:0>

Transfer 1
Transfer 2
Transfer 3
... ...
Transfer n

COUNT = 0
Set RAFx (ADCDMASTAT).
Generate interrupt if RAFIENx bit is set.

COUNT++

Transfer 1
Transfer 2
Transfer 3
... ...
Transfer n

COUNT = 2DMABL<2:0>
Set RBFx (ADCDMASTAT).
Generate interrupt if RBFIENx bit is set.

COUNT++

Buffer A

COUNT = 2DMABL<2:0>

Buffer B
22.5.5 Capacitive Voltage Divider (CVD) mode

The ADC has CVD mode, which can be used to detect an event of touch in a touch sensor application. The principle of CVD measurement is based on charge balancing between two capacitors and then measuring the voltage.

In touch sensing applications, the presence of finger near a pad alters the capacitance of the pad. This variation in capacitance can be sensed by the CVD module to detect the presence (and subsequent absence) of a finger.

**Note:** Only shared analog inputs (Class 2 and Class 3) can be used for CVD measurement.

- When a finger is touching the sensor pad, external capacitance = \( C_{\text{EXT}} = C_{\text{PAD}} + C_{\text{FINGER}} \)
- External capacitance of pad (when finger is not touching) = \( C_{\text{EXT}} = C_{\text{PAD}} \)
- Internal capacitance = \( C_{\text{INT}} = C_{\text{PLINE}} + C_{\text{SAMP}} \)

**Figure 22-20: CVD Mode Diagram**
Section 22. 12-bit High-Speed SAR ADC

CVD measurement consists of two phases, Positive and Negative.

Positive Phase: The \( C_{EXT} \) is connected to \( V_{DD} \) and \( C_{INT} \) is connected to ground (GND) (as shown in Figure 22-21). Then, both capacitors are connected in parallel for half of sampling time, set by the SAMC<9:0> bits (ADCCON2<25:16>). After the half sampling time is complete, the voltage is converted by the ADC module and is named \( V_{INP} \).

Figure 22-21: CVD Mode Positive Phases Diagram

When finger is touching the pad:
\[ C_{EXT} = C_{PAD} + C_{FINGER} \]

When finger is removed:
\[ C_{EXT} = C_{PAD} \]
Negative Phase: The $C_{EXT}$ is connected to GND and $C_{INT}$ is connected to $V_{DD}$, as shown in Figure 22-22. Similar to the positive phase, the voltage is measured for the negative phase and is named as $V_{INN}$.

Figure 22-22: CVD Mode - Negative Phase Diagram

When finger is touching the pad:
$C_{EXT} = C_{PAD} + C_{FINGER}$

When finger is removed:
$C_{EXT} = C_{PAD}$
Section 22. 12-bit High-Speed SAR ADC

The CVD module internally calculates the difference between \( V_{\text{INP}} \) and \( V_{\text{INN}} \) and stores the data in the CVDDATA<15:0> bits (ADCCMPCON1<31:16>) in signed format. The plot in Figure 22-23 shows how during a touch event, the increase in external capacitance causes the \( (V_{\text{INP}} - V_{\text{INN}}) \) to be higher than the non-touch condition (decreased external capacitance).

**Figure 22-23: CVD Signal - Difference Between Pressed and Released Differential Values**

![Diagram showing the voltage levels and phases of a touch event](image)

**Equation 22-3** shows the relation \( (V_{\text{INP}} - V_{\text{INN}}) \).

**Equation 22-3: \( V_{\text{INP}} \) and \( V_{\text{INN}} \) Relation**

\[
(V_{\text{INP}} - V_{\text{INN}}) = V_{DD} \cdot \left( \frac{(C_{\text{EXT}} - C_{\text{INT}})}{(C_{\text{EXT}} + C_{\text{INT}})} \right)
\]

During a touch condition, \( C_{\text{EXT}} \) increases and \( (V_{\text{INP}} - V_{\text{INN}}) \) increases.

During no touch condition, \( C_{\text{EXT}} \) decreases and \( (V_{\text{INP}} - V_{\text{INN}}) \) reduces.

Digital Comparator 1 is linked to CVD and it can be used to generate a comparator event and interrupt when a touch is detected, by setting the IEHIHI bit (ADCCMPCON1<3>). The digital comparator can generate an event when measured \( (V_{\text{INP}} - V_{\text{INN}}) \) is above the value set in the DCMPHI<15:0> bit (ADCCMP1<31:16>). When the comparator event is detected by reading the DCMPED bit (ADCCMPCON1<5>), or inside the ISR, the analog input that caused the touch event can be read from the AINID<5:0> bits (ADCCMPCON1<13:8>).

To ensure maximum sensitivity, \( C_{\text{INT}} \) should have similar value as \( C_{\text{PAD}} \), which can be done by suitably selecting the value of \( C_{\text{PLINE}} \) capacitance through the CVDCPL<2:0> bits (ADCCON2<28:26>).

**Note:** Before CVD is enabled by setting the CVDEN bit, external triggers for all Class 2 and Class 3 inputs are disabled by clearing the TRGSRC<4:0> and STRGSRC<4:0> bits.
Example 22-7: ADC CVD Mode

```c
unsigned char touchDetected = 0;

void __ISR(_ADC_DC1_VECTOR, IPL3AUTO) __IntHandlerDrvAdc(void)
{
  IFS1bits.ADCDC1IF = 0;
  /* Check and clear event flag for comparator 1 */
  if(ADCCMPCON1bits.DCMPED == 1)
  {
    /* Verify if comparator event really due to AN21 */
    if(ADCCMPCON1bits.AINID == 21)
    {
      touchDetected = 1;
    }
  }
}

int main(void)
{
  /* Configure ADCCON1 */
  ADCCON1bits.SELRES = 3; // ADC resolution is 12 bits
  ADCCON1bits.STRGSR = 0; // No scan trigger.
  ADCCON1bits.AICMPFEN = 0;
  ADCCON1bits.FRACT = 0; // Integer format

  /* Configure ADCCON2 */
  ADCCON2bits.SAMC = 32; // ADC7 sampling time
  ADCCON2bits.ADCDIV = 4; // ADC7 clock freq is half of control clock = TAD7
  ADCCON2bits.CVDCPL = 2; // 5 pF is the CVD capacitor value selected

  /* Initialize warm up time register */
  ADCANCON = 0;
  ADCANCONbits.WKUPCLKCNT = 5; // Wakeup exponent = 32 * TADx

  /* Clock setting */
  ADCCON3 = 0;
  ADCCON3bits.ADCSEL = 0; // Select input clock source
  ADCCON3bits.CONCLKDIV = 1; // Control clock frequency is half of input clock
  ADCCON3bits.VREFSEL = 0; // Select AVDD and AVSS as reference source

  /* No selection for dedicated ADC modules, no presync trigger, not sync sampling */
  ADCTRGMODE = 0;
  /* Select ADC input mode */
  ADCIMCON2bits.SIGN21 = 1; // signed data format
  ADCIMCON2bits.DIFF21 = 0; // Single ended mode

  /* Configure ADCGIRQENx */
  ADCGIRQEN1 = 0; // No interrupts are used.
  ADCGIRQEN2 = 0;

  /* Configure ADCCSSx */
  ADCCSS1 = 0; // No scanning is used
  ADCCSS2 = 0;
  ADCCSS1bits.CSS21 = 1; // AN21 (Class 3) set for CVD

  /* Configure ADCCMPCONx */
  ADCCMP1 = 0; // Clear the register
  ADCCMP1bits.DCMPHI = 1600; // High limit, depends on touch pad layout and selected
  ADCCMP1bits.DCMPL0 = 0; // Low limit, not very important for sensing touch event
  ADCCMP1bits.IEHIHI = 1; // When touched, the CVDDATA > HI_LIMIT(DCMPHI)
```

---

PIC32 Family Reference Manual

Example 22-7: ADC CVD Mode

```c
unsigned char touchDetected = 0;

void __ISR(_ADC_DC1_VECTOR, IPL3AUTO) __IntHandlerDrvAdc(void)
{
  IFS1bits.ADCDC1IF = 0;
  /* Check and clear event flag for comparator 1 */
  if(ADCCMPCON1bits.DCMPED == 1)
  {
    /* Verify if comparator event really due to AN21 */
    if(ADCCMPCON1bits.AINID == 21)
    {
      touchDetected = 1;
    }
  }
}

int main(void)
{
  /* Configure ADCCON1 */
  ADCCON1bits.SELRES = 3; // ADC resolution is 12 bits
  ADCCON1bits.STRGSR = 0; // No scan trigger.
  ADCCON1bits.AICMPFEN = 0;
  ADCCON1bits.FRACT = 0; // Integer format

  /* Configure ADCCON2 */
  ADCCON2bits.SAMC = 32; // ADC7 sampling time
  ADCCON2bits.ADCDIV = 4; // ADC7 clock freq is half of control clock = TAD7
  ADCCON2bits.CVDCPL = 2; // 5 pF is the CVD capacitor value selected

  /* Initialize warm up time register */
  ADCANCON = 0;
  ADCANCONbits.WKUPCLKCNT = 5; // Wakeup exponent = 32 * TADx

  /* Clock setting */
  ADCCON3 = 0;
  ADCCON3bits.ADCSEL = 0; // Select input clock source
  ADCCON3bits.CONCLKDIV = 1; // Control clock frequency is half of input clock
  ADCCON3bits.VREFSEL = 0; // Select AVDD and AVSS as reference source

  /* No selection for dedicated ADC modules, no presync trigger, not sync sampling */
  ADCTRGMODE = 0;
  /* Select ADC input mode */
  ADCIMCON2bits.SIGN21 = 1; // signed data format
  ADCIMCON2bits.DIFF21 = 0; // Single ended mode

  /* Configure ADCGIRQENx */
  ADCGIRQEN1 = 0; // No interrupts are used.
  ADCGIRQEN2 = 0;

  /* Configure ADCCSSx */
  ADCCSS1 = 0; // No scanning is used
  ADCCSS2 = 0;
  ADCCSS1bits.CSS21 = 1; // AN21 (Class 3) set for CVD

  /* Configure ADCCMPCONx */
  ADCCMP1 = 0; // Clear the register
  ADCCMP1bits.DCMPHI = 1600; // High limit, depends on touch pad layout and selected
  ADCCMP1bits.DCMPL0 = 0; // Low limit, not very important for sensing touch event
  ADCCMP1bits.IEHIHI = 1; // When touched, the CVDDATA > HI_LIMIT(DCMPHI)
```
Example 22-7: ADC CVD Mode (Continued)

```c
ADCCMPCON1bits.IEBTWN = 0;
ADCCMPCON1bits.IEHILO = 0;
ADCCMPCON1bits.IELOHI = 0;
ADCCMPCON1bits.IELOLO = 0;
ADCCMPCON1bits.DCMPGIEN = 1;  // enable interrupt
ADCCMPCON1bits.ENDCMP = 1;  // enable comparator
ADCMPEN1 = 0;  // Clear all enable bits, as it is irrelevant in CVD mode
ADCCMPCON2 = 0;
ADCCMPCON3 = 0;
ADCCMPCON4 = 0;
ADCCMPCON5 = 0;
ADCCMPCON6 = 0;

/* Configure ADCFLTRx */
ADCFLTR1 = 0;  // No oversampling filters are used.
ADCFLTR2 = 0;
ADCFLTR3 = 0;
ADCFLTR4 = 0;
ADCFLTR5 = 0;
ADCFLTR6 = 0;

/* Set up the trigger sources */
ADCTRGSNS = 0;
ADCTRG1 = 0;
ADCTRG2 = 0;
ADCTRG3 = 0;

/* Early interrupt */
ADCEIEN1 = 0;  // No early interrupt
ADCEIEN2 = 0;

/* Turn the ADC on */
ADCCON1bits.ON = 1;

/* Wait for voltage reference to be stable */
while(!ADCCON2bits.BGVRRDY); // Wait until the reference voltage is ready
while(ADCCON2bits.REFFLT); // Wait if there is a fault with the reference voltage

/* Enable clock to analog circuit */
ADCCANCONbits.ANEN7 = 1;  // Enable the clock to analog bias

/* Wait for ADC to be ready */
while(ADCCANCONbits.WKRDY7);  // Wait until the ADC7 is ready

/* Enable interrupt setting for digital comparator 1 */
IFS1bits.ADCDC1IF = 0;
IER1bits.ADCDC1IE = 1;
IPC11bits.ADCDC1IP = 3;
IPC11bits.ADCDC1IS = 3;

INTCONbits.MVEC = 1;
__builtin_enable_interrupts();

/* Enable the ADC module */
ADCCON3bits.DIGEN7 = 1;  // Enable ADC7

/* Turn the CVD mode on */
ADCCON1bits.CVDEN = 1;
while (1);
return (1);
```
22.5.6 Double Fast Turbo Channel

In certain applications, the user may require to achieve a data throughput from the ADC that is higher than what can be possible from a single dedicated ADC module. In such a case, the Turbo mode can be used, which interleaves two dedicated ADC modules. By interleaving two dedicated modules, one ADC module would be sampling, while the other ADC module would be converting. Therefore, the throughput could be increased to a double rate.

22.5.6.1 CONFIGURING TURBO CHANNEL

The Turbo mode consists of two dedicated ADC modules, called master and slave, which are selected by the TRBMST<2:0> bits (ADCCON1<29:27>) and the TRBSLV<2:0> bits (ADCCON1<26:24>), respectively. When the master ADC module is triggered by selection of the TRGSRCx<4:0> bits (ADCTRGMODE), the master ADC completes the sampling time and undergoes conversion. At the same time, the master ADC triggers the slave in the middle of ($t_{SAMC} + t_{CONV}$), which begins its sampling. The discrete steps, which follow, explain Turbo mode operation:

1. Master ADC receives trigger from external source (selected by TRGSRCx<4:0>).
2. Master ADC goes to Sampling mode.
3. After sampling, conversion begins. Also, master ADC triggers the slave ADC in the middle of ($t_{SAMC} + t_{CONV}$). Therefore, the slave ADC enters Sampling mode.
4. While master ADC is converting data, the slave ADC completes sampling and starts conversion.
5. Master ADC ends conversion and the slave ADC also ends conversion after some time.

Overall, at the end of conversion, the data is available at a much higher rate, as compared to data rate from a single channel.

At the end of conversion, the output data goes to two separate data buffers (ADCDATAx) associated with the analog inputs of each dedicated ADC module or to a FIFO or system RAM.

Note: Even if two dedicated ADC modules are configured as master and slave, they still maintain their separate analog input pins and it is the user responsibility to short these two inputs (or two and two if in differential mode) at the application board level to present the same input signal to both master and slave for sampling and conversion.

22.5.6.2 TURBO CHANNEL ERROR

There are several configuration conditions that must be met before the turbo channel can function. If turbo channel is not configured appropriately, the turbo channel error bit, TRBERR (ADCCON1<30>), is set by hardware. When the TRBERR bit (ADCCON1<30>) is set, the hardware will disable the functioning of the turbo channel despite the fact the user code had set the TRBEN bit (ADCCON1<31>) to ‘1’. The turbo channel error makes sure that the user chooses the right setup for the turbo channel. The following conditions must be matched for proper turbo channel operation; otherwise, the TRBERR bit will be set by hardware:

- ADC module clock divisor: The ADCDIV<6:0> bits (ADCxTIME<22:16>) of master and slave ADC modules should be the same
- ADC resolution: The SELRES<1:0> bits (ADCxTIME<25:24>) of master and slave ADC modules should be same. Also, 6-bit resolution (SELRES<1:0> = 00) is not supported in Turbo mode. Therefore, the value of the SELRES<1:0> bits should be other than '00'.
- Sampling time: The SAMC<9:0> bits (ADCxTIME<9:0>) of master and slave ADC modules should be same
- The STRGEN bit (ADCTRGMODE) for both master and slave ADC modules should be set
- The SSAMPEN bit (ADCTRGMODE) for both master and slave ADC modules should be set
22.5.6.3 EFFECTIVENESS OF TURBO CHANNEL

Turbo channel will only be effective if, the sample time (tsAMC) is less than the conversion time (tCONV). This means that one ADC module will finish sampling before the other ADC module finishes conversion. Otherwise, both ADC modules will end up sampling at the same time, which defeats the purpose of turbo channel. For a single SAR ADC, the ADC conversion time is the time in which no output data is produced. Using the turbo channel, the conversion time is interleaved with sampling of other channel.

The sample time of ADC depends on the input resistance of analog source. To reduce the sample time, the input resistance of analog source should be reduced.

If the user is presented with an application where (tsAMC) is greater than the conversion time (tCONV) (as then the source impedance cannot be reduced any further due to design constraints or productized hardware etc.), the clock frequency can be reduced by increasing the ADCDIV<6:0> bits (ADCxTIME<22:16>) in such a way that the conversion time (counts * TAD) is the same as the sampling time (not counts). Please note that the counts for the sampling time SAMC<9:0> bits (ADCxTIME<9:0>) should be made smaller due to the reduced clock frequency.

If the sampling time is very large compared to the fastest conversion time (14 * TAD) supported, the turbo channel will not achieve a significant increase in conversion bandwidth. When this occurs, it is better to run in single channel mode at the fastest conversion rate supported by the hardware.

Figure 22-24 illustrates the operation of turbo channel mode with the master trigger in edge and level sensitivity mode.

![Figure 22-24: Turbo Mode Operation](image-url)

While in Interrupt mode, the user application should enable the AGIENx bit of the analog inputs associated with both the master and slave ADC modules.
## 22.6 INTERRUPTS

Each ADC module supports interrupts triggered from a variety of sources, which can be processed individually or globally. An early interrupt feature is also available to compensate for interrupt servicing latency.

After an enabled interrupt is generated, the CPU will jump to the vector assigned to that interrupt. The CPU will then begin executing code at the vector address. The user software at this vector address should perform the required operations, such as processing the data results, clearing the interrupt flag, and then exit. For more information on interrupts and the vector address details, refer to the Section 8. “Interrupts” (DS60001108) in the “PIC32 Family Reference Manual” and the “Interrupt Controller” chapter in the specific device data sheet.

### 22.6.1 Interrupt Sources

Each ADC is capable of generating interrupts from the events listed in Table 22-8.

#### Table 22-8: ADC Interrupt Sources

<table>
<thead>
<tr>
<th>Interrupt Event</th>
<th>Description</th>
<th>Interrupt Enable bit</th>
<th>Interrupt Status bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANx Data Ready Event</td>
<td>Interrupt is generated upon a completion of a conversion from an analog input source (ANx). Each of the ARDYx bits is capable of generating a unique interrupt when set, using the ADCBASE register.</td>
<td>AGIENx of ADCGIROQEN1 or ADCGIROQEN2 register</td>
<td>ARDYx of ADCDSTATx register</td>
</tr>
<tr>
<td>Digital Comparator Event</td>
<td>When a conversion's comparison criteria are met by a configured and enabled digital comparator. Each of the digital comparators is capable of generating a unique interrupt when its DCMPED bit is set.</td>
<td>DCMPGIEN of ADCCMPCONx register</td>
<td>DCMPED of ADCCMPCONx register</td>
</tr>
<tr>
<td>Oversampling Filter Data Ready Event</td>
<td>When an oversampling filter has completed the accumulation/decimation process and has stored the result.</td>
<td>AFGIEN of ADCFLTRx register</td>
<td>AFRDY of ADCFLTRx register</td>
</tr>
<tr>
<td>Both Band Gap Voltage and ADC Reference Voltage Ready Event</td>
<td>Interrupt is generated when both band gap voltage and ADC reference voltage are ready.</td>
<td>BGVRIEN of ADCCON2 register</td>
<td>BGVRRDY of ADCCON2 register</td>
</tr>
<tr>
<td>End of Scan Event</td>
<td>Interrupt is generated when all the selected inputs have completed scan.</td>
<td>EOSIEN of ADCCON2 register</td>
<td>EOSRDY of ADCCON2 register</td>
</tr>
<tr>
<td>FIFO Data Ready Event</td>
<td>Interrupt is generated when data is ready to be read from FIFO.</td>
<td>FIEN of ADCFRSTAT register</td>
<td>FRDY of ADCFRSTAT register</td>
</tr>
<tr>
<td>DMA Buffer B Full Event</td>
<td>Interrupt is generated when DMA Buffer B is full.</td>
<td>RBFIEN6:RBFIEN0 of ADCDMASTAT register</td>
<td>RBF6:RBF0 of ADCDMASTAT register</td>
</tr>
<tr>
<td>DMA Buffer A Full Event</td>
<td>Interrupt is generated when DMA Buffer A is full.</td>
<td>RAIFIEN6:RAIFIEN0 of ADCDMASTAT register</td>
<td>RAF6:RAF0 of ADCDMASTAT register</td>
</tr>
<tr>
<td>ADC Module Wake-up Event</td>
<td>Interrupt is generated when ADC wakes up after being enabled.</td>
<td>WKIEN6:WKIEN0, WKIEN7 of ADCANCON register</td>
<td>WKRDY6:WKRDY0, WKRDY7 of ADCANCON register</td>
</tr>
<tr>
<td>Update Ready Event</td>
<td>Interrupt is generated when ADC SFRs are ready to be (and can be safely) updated with new values.</td>
<td>UPDIEN of ADCCON3 register</td>
<td>UPDRDY of ADCCON3 register</td>
</tr>
<tr>
<td>Early Interrupt Event</td>
<td>Interrupt is generated earlier than certain ADC clocks (prior to end-of-conversion) as set by ADCEIS&lt;2:0&gt; bits (ADCXTIME&lt;28:26&gt;) and ADCEIS&lt;2:0&gt; bits (ADCCON2&lt;10:8&gt;).</td>
<td>EIIEN6:EIIEN0 of ADCEIENx register</td>
<td>EIRDY6:EIRDY0 of ADCEIESTAfx register</td>
</tr>
</tbody>
</table>
22.6.2 ADC Base Register (ADCBASE) Usage

After conversion of ADC is complete, if the interrupt is vectored to a function which is common to all analog inputs, it takes some significant time to find the ADC input by evaluating the ARDYx bits in the ADCDSTATx. To avoid this time spent, ADCBASE register is provided, which contains the base address of the user's ADC ISR jump table. When read, the ADCBASE register will provide a sum of the contents of the ADCBASE register plus an encoding of the ARDYx bits set in the ADCDSTATx registers. This use of the ADCBASE register supports the creation of an interrupt vector address that can be used to improve the performance of an ISR.

The ARDYx bits are binary priority encoded with ARDY0 being the highest priority, and A63RDY being the lowest priority. The encoded priority result is then shifted left the amount specified by the number of bit positions specified by the IRQVS<2:0> bits in the ADCCON1 register, and then added to the contents of the ADCBASE register. If there are no ARDYx bits set, then reading the ADCBASE register will equal the value written into the ADCBASE register.

The ADCBASE register will typically be loaded with the base address of a jump table that will contain the address of appropriate ISR. The k\textsuperscript{th} interrupt request is enabled via the AGIENx bit (0 ≤ x ≤ 63) in one of ADCGIRQENx SFRs (x = 1 or 2). Example 22-8 shows the usage of ADCBASE register. In addition, the code in Example 22-9 shows how to use the ADCBASE register for vectoring interrupts for different ADC inputs.

Example 22-8: ADCBASE Register Usage

<table>
<thead>
<tr>
<th>Case 1:</th>
<th>ADCBASE = 0x1234; // Set the address</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ADCCON1bits.IRQVS = 2; // left shift by 2</td>
</tr>
<tr>
<td></td>
<td>ADCGIRQEN1bits.AGIEN0 = 1; // enable interrupt when AN0 completion is done.</td>
</tr>
</tbody>
</table>

Once the ADC conversion for AN0 is complete, bit 0 of ADCDSTAT1 = ARDY0 is set. Read value of ADCBASE = 0x1234 + (0 << 2) = 0x1234.

Therefore, the ISR should be placed at address 0x1234 for AN0.

<table>
<thead>
<tr>
<th>Case 2:</th>
<th>ADCBASE = 0x1234; // Set the address</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ADCCON1bits.IRQVS = 2; // left shift by 2</td>
</tr>
<tr>
<td></td>
<td>ADCGIRQEN1bits.AGIEN0 = 2; // enable interrupt when AN2 completion is done.</td>
</tr>
</tbody>
</table>

Once the ADC conversion for AN2 is complete, bit 2 of ADCDSTAT1 = ARDY2 is set. Read value of ADCBASE = 0x1234 + (2 << 2) = 0x123C.

Therefore, the ISR should be placed at address 0x123C for AN2.

Note: The contents of the ADCBASE register are not altered. Summation is performed when ADCBASE register is read and the summation result is the returned read value from the ADCBASE SFR.
Example 22-9: Vectoring Interrupts for Different ADC Inputs

/* Number of ADC modules doing conversion */
#define ADC_MODULES 3

/* Declare functions for each ADC handler */
void ADC0Handler(void);
void ADC1Handler(void);
void ADC2Handler(void);

void(*jumpTable[ADC_MODULES * 2])(void);

int ADC0Result;
int ADC1Result;
int ADC2Result;

int main(int argc, char** argv) {
    jumpTable[0] = &ADC0Handler; // Set up jump table
    jumpTable[2] = &ADC1Handler;
    jumpTable[4] = &ADC2Handler;

    /* Configure ADCCON1 */
    ADCCON1 = 0; // No ADCCON1 features are enabled including: Stop-in-Idle, turbo,
                // CVD mode, Fractional mode and scan trigger source.

    /* Configure ADCCON2 */
    ADCCON2 = 0; // Since, we are using only the Class 1 inputs, no setting is
                // required for ADCDIV

    /* Initialize warm up time register */
    ADCANCON = 0;
    ADCANCONbits.WKUPCLKCNT = 5; // Wakeup exponent = 32 * TADx

    /* Clock setting */
    ADCCON3 = 0;
    ADCCON3bits.ADCSEL = 0; // Select input clock source
    ADCCON3bits.CONCLKDIV = 1; // Control clock frequency is half of input clock
    ADCCON3bits.VREFSEL = 0; // Select AVDD and AVSS as reference source

    /* Select ADC sample time and conversion clock */
    ADC0TIMEbits.ADCDIV = 1; // ADC0 clock frequency is half of control clock = TAD0
    ADC0TIMEbits.SAMC = 5; // ADC0 sampling time = 5 * TAD0
    ADC0TIMEbits.SELRES = 3; // ADC0 resolution is 12 bits
    ADC1TIMEbits.ADCDIV = 1; // ADC1 clock frequency is half of control clock = TAD1
    ADC1TIMEbits.SAMC = 5; // ADC1 sampling time = 5 * TAD1
    ADC1TIMEbits.SELRES = 3; // ADC1 resolution is 12 bits
    ADC2TIMEbits.ADCDIV = 1; // ADC2 clock frequency is half of control clock = TAD2
    ADC2TIMEbits.SAMC = 5; // ADC2 sampling time = 5 * TAD2
    ADC2TIMEbits.SELRES = 3; // ADC2 resolution is 12 bits

    /* Select analog input for ADC modules, no presync trigger, not sync sampling */
    ADCTRGMODEbits.SH0ALT = 0; // ADC0 = AN0
    ADCTRGMODEbits.SH1ALT = 0; // ADC1 = AN1
    ADCTRGMODEbits.SH2ALT = 0; // ADC2 = AN2

    /* Select ADC input mode */
    ADCIMCON1bits.SIGN0 = 0; // unsigned data format
    ADCIMCON1bits.DIFF0 = 0; // Single ended mode
    ADCIMCON1bits.SIGN1 = 0; // unsigned data format
    ADCIMCON1bits.DIFF1 = 0; // Single ended mode
    ADCIMCON1bits.SIGN2 = 0; // unsigned data format
    ADCIMCON1bits.DIFF2 = 0; // Single ended mode

    /* Configure ADCGIRQENx */
    ADCGIRQEN1 = 0;
    ADCGIRQEN2 = 0;
    ADCGIRQEN1bits.AGIN0 = 1; // Enable data ready interrupt for AN0
    ADCGIRQEN1bits.AGIN1 = 1; // Enable data ready interrupt for AN1
    ADCGIRQEN1bits.AGIN2 = 1; // Enable data ready interrupt for AN2

    /* Configure ADBASE */
    ADBASE = (int)(&jumpTable[0]); // Initialize ADBASE with starting address of jump table
    ADBASEbits.IRQVS = 0; // No left shift of address

    /* Configure ADCCSSx */
    ADCCSS1 = 0; // No scanning is used
    ADCCSS2 = 0;
Example 22-9: Vectoring Interrupts for Different ADC Inputs (Continued)

/* Configure ADCCMPCONx */
ADCCMPCON1 = 0; // No digital comparators are used. Setting the ADCCMPCONx
ADCCMPCON2 = 0; // register to '0' ensures that the comparator is disabled.
ADCCMPCON3 = 0; // Other registers are "don't care".
ADCCMPCON4 = 0;
ADCCMPCON5 = 0;
ADCCMPCON6 = 0;

/* Configure ADCFLTRx */
ADCFLTR1 = 0; // No oversampling filters are used.
ADCFLTR2 = 0;
ADCFLTR3 = 0;
ADCFLTR4 = 0;
ADCFLTR5 = 0;
ADCFLTR6 = 0;

/* Set up the trigger sources */
ADCTRGSNSbits.LVL0 = 0; // Edge trigger
ADCTRGSNSbits.LVL1 = 0; // Edge trigger
ADCTRGSNSbits.LVL2 = 0; // Edge trigger
ADCTRGRbits.TRGSRC0 = 1; // Set AN0 to trigger from software.
ADCTRGRbits.TRGSRC1 = 1; // Set AN1 to trigger from software.
ADCTRGRbits.TRGSRC2 = 1; // Set AN2 to trigger from software.

/* Early interrupt */
ADCEIEN1 = 0; // No early interrupt
ADCEIEN2 = 0;
ADCCON2bits.ADCEIOVR = 1; // Override early interrupt

/* Turn the ADC on */
ADCCON1bits.ON = 1;

/* Wait for voltage reference to be stable */
while(!ADCCON2bits.BGVRDY); // Wait until the reference voltage is ready
while(ADCCON2bits.REFFLT); // Wait if there is a fault with the reference voltage

/* Enable clock to analog circuit */
ADCCONbits.ANEN0 = 1; // Enable the clock to analog bias and digital control
ADCCONbits.ANEN1 = 1; // Enable the clock to analog bias and digital control
ADCCONbits.ANEN2 = 1; // Enable the clock to analog bias and digital control

/* Wait for ADC to be ready */
while(!ADCCONbits.WKRDY0); // Wait until ADC0 is ready
while(!ADCCONbits.WKRDY1); // Wait until ADC1 is ready
while(!ADCCONbits.WKRDY2); // Wait until ADC2 is ready

/* Enable the ADC module */
ADCCONbits.DIGEN0 = 1; // Enable ADC0
ADCCONbits.DIGEN1 = 1; // Enable ADC1
ADCCONbits.DIGEN2 = 1; // Enable ADC2

/* Trigger a conversion */
ADCCONbits.GSWTRG = 1;
while (1);
return (1);

/* Handler for the ADC interrupt */
void __ISR(_ADC_VECTOR, ipl3) ADCHandler1(void)
{
    /* call the corresponding ADC module handler */
    ((void(*)(void))*(int *)ADCBASE)();
}

void ADC0Handler(void)
{
    /* Verify if data for AN0 is ready. This bit is self cleared upon data read */
    if(ADCDSTAT1bits.ARDY0)
    {
        ADC0Result = ADCDATA0;
    }
}
Example 22-9: Vectoring Interrupts for Different ADC Inputs (Continued)

```c
void ADC1Handler(void)
{
    /* Verify if data for AN1 is ready. This bit is self cleared upon data read */
    if(ADCDSTAT1bits.ARDY1)
    {
        ADC1Result = ADCDATA1;
    }
}
void ADC2Handler(void)
{
    /* Verify if data for AN2 is ready. This bit is self cleared upon data read */
    if(ADCDSTAT2bits.ARDY2)
    {
        ADC2Result = ADCDATA2;
    }
}
```

### 22.6.3 Interrupt Enabling, Priority and Vectoring

Each of the ADC events previously mentioned will generate an interrupt when its associate Interrupt Enable bit, IE, is set. An Interrupt Flag bit, IF, priority bits, IP<2:0>, and sub-priority bits IS<1:0> are also associated with each of the events. For more information on how to enable and prioritize interrupts, refer to Section 8, “Interrupts” (DS60001108). Each of the ADC events previously listed also has an associated interrupt vector. Refer to the “Interrupt Controller” chapter in the specific device data sheet for more information on the vector location and control/status bits associated with each individual interrupt.

### 22.6.4 Individual and Global Interrupts

The use of the individual interrupts previously listed can significantly optimize the servicing of multiple ADC events, by keeping each ISR focused on efficiently handling a specific event. In addition, different ISRs can be easily segregated according to the tasks performed, thereby making user software easier to implement and maintain. There may be cases where it is desirable to have a single ISR service multiple interrupt events. To facilitate this, each ADC event can be logically “ORed” to create a single global ADC interrupt. When an ADC event is enabled for a global interrupt, it will vector to a single interrupt routine. It will be the responsibility of this single global ISR to determine the source of the interrupt through polling and process it accordingly.

Use of the Global Interrupt requires configuration of its own unique IE, IF, IP and IS bits as well as configuration of its interrupt vector as described in 22.6.3 “Interrupt Enabling, Priority and Vectoring”.

Interrupts for the ADC can be configured as individual or global, or utilize both where some are processed individually and others in the global ISR.
22.6.5 Early Interrupts

The early interrupt feature lets ADC module generate interrupt before the conversion is complete. Even though the input is still in the conversion process, the processor application software can use the “head-start” to begin execution of the entry into the ISR. The early interrupt can improve the throughput of a system by overlapping the completion of the ADC conversion with the processor overhead associated with an interrupt. The ADCEIS<2:0> bits (ADCxTIME<28:26>) and ADCEIS<2:0> bits (ADCCON2<10:8>) sets the number of ADC clock prior to which, the interrupt should occur (for dedicated and shared inputs respectively). Suitably configuring these bits can reduce the latency from the moment the analog signal was sampled until the point in time when the user application software can use the data.

Once the set number of ADC clocks reached prior to end-of-conversion (i.e., prior to data actually being available in the ADCDATAx register), the corresponding early interrupt ready bit, EIRDYx, in the ADCEISTAT1 or ADCEISTAT2 register is set. If the respective interrupt enable bit, EIENx, in the ADCEIEN1 or ADCEIEN2 register is set, the interrupt will be generated.

The early interrupt feature can be overridden by setting the Early Interrupt Override bit, ADCEIOVR (ADCCON2<12>). Once this bit is set, setting the bits in the ADCEIEN1 or ADCEIEN2 register has no effect on interrupt generation. The interrupt generation is then controlled by the setting of ADCGIRQEN1 and ADCGIRQEN2 registers.

**Note:** The ADCEIS<2:0> bits setting does not apply to the Oversampling Filter Data Ready signal, AFRDY.
22.7 OPERATION DURING POWER-SAVING MODES

The power-saving modes, Sleep and Idle, are useful for reducing the conversion noise by minimizing the digital activity of the CPU, buses and other peripherals.

22.7.1 Sleep Mode

When device enters Sleep mode, the system clock (SYCCLK) is halted. If an ADC module selects SYSCLK as its clock source or selects REFCLK3 as its clock source (REFCLK3 is generated from SYSCLK), the ADC will enter the Sleep mode.

When the SYSCLK is the source, (directly or indirectly) and Sleep mode occurs during a conversion, the conversion is aborted. The converter will not resume a partially completed conversion on exiting from Sleep mode. The ADC register contents are not affected by the device entering or leaving Sleep mode. The ADC module can operate during Sleep mode if the ADC clock source is derived from a source other than SYSCLK that is active during Sleep mode. The FRC clock source is a logical choice for operation during Sleep; however, the REFCLK3 clock source can also be used, provided it has an input clock that is operational during Sleep mode.

ADC operation during Sleep mode reduces the digital switching noise from the conversion. When the conversion is completed, the ARDYx status bit for that analog input will be set and the result will be loaded into the corresponding ADC Result register (ADCDATAx).

If any of the ADC interrupts is enabled, the device will wake-up from Sleep mode when the ADC interrupt occurs. The program execution will resume at the ADC ISR, if the ADC interrupt is greater than the current CPU priority. Otherwise, execution will continue from the instruction after the \texttt{WAIT} instruction that placed the device in Sleep mode.

To minimize the effects of digital noise on the ADC module operation, the user must select a conversion trigger source that ensures that the analog-to-digital conversion will take place in Sleep mode. For example, the external interrupt pin (INT0) conversion trigger option (TRGSRC<4:0> = 00100) can be used for performing sampling and conversion while the device is in Sleep mode.

\begin{center}
\textbf{Note:} For the ADC module to operate in Sleep mode, the ADC clock source must be set to Internal FRC (ADCSEL<1:0> bits (ADCCON2<31:30>) = 01). Alternately, the REFCLK3 source can be used; however, the clock source used for REFCLK3 must operate during Sleep. Any changes to the ADC clock configuration require that the ADC be disabled.
\end{center}

22.7.2 ADC Operation During Idle Mode

For the ADC, the Stop in Idle Mode bit, SIDL (ADCCON1<13>), specifies whether the ADC module will stop on Idle or continue on Idle. If SIDL = 0, the ADC module will continue normal operation when the device enters Idle mode. If any of the ADC interrupts are enabled, the device will wake-up from Idle mode when the ADC interrupt occurs. The program execution will resume at the ADC ISR if the ADC interrupt is greater than the current CPU priority. Otherwise, execution will continue from the instruction after the \texttt{WAIT} instruction that placed the device in Idle mode.

If SIDL = 1, the ADC module will stop in Idle mode. If the device enters Idle mode during a conversion, the conversion is aborted. The converter will not resume a partially completed conversion on exiting from Idle mode.
22.7.3 ADC Low-power Mode

The ADC module can be placed in a low-power state by disabling the digital circuit for individual ADC modules, which are not running. This is possible by clearing the DIGENx bits and the DIGEN7 bit in the ADCCON3 register (see Register 22-3).

An even lower power state is possible by disabling the analog and bias circuit for individual ADC modules, which are not running. This is possible by clearing the ANENx bits and the ANEN7 bit in the ADCANCON register (see Register 22-41). Disabling the digital circuit to achieve Low-power mode provides a significantly faster module restart compared to disabling and re-enabling the analog and bias circuit of the ADC module. This is because disabling and re-enabling the analog and bias circuit using the ANENx bits and the ANEN7 bit requires a wake-up time (typical minimum wake-up time of 20 µs) for the ADC module, before it can be used. Refer to the “Electrical Characteristics” chapter in the specific device data sheet for more information on the stabilization time.

Once the analog and bias circuit for an ADC module is enabled, the wake-up should be polled (or through an interrupt) using the wake-up ready bits, WKRDY6:WKRDY0 and WKRDY7, which should be equal to ‘1’.
22.8  Effects of Reset

Following any Reset event, all the ADC control and status registers are reset to their default values with control bits in a non-active state. This disables the ADC module and sets the analog input pins to Analog Input mode. Any conversion that was in progress will terminate and the result will not be written to the result buffer. The values in the ADCDATAx registers are initialized to 0x00000000 during a device Reset. The bias circuits are also turned off, so the ADC resuming operations will have to wait for the bias circuits to stabilize by polling (or requesting to be interrupted by) the BGVRRDY bit (ADCCON2 register).

22.9  Transfer Function

A typical transfer function of the 12-bit ADC is illustrated in Figure 22-25. The difference of the input voltages (VinH - VinL) is compared with the reference (VrefH - VrefL).

- The first code transition (A) occurs when the input voltage is (VrefH - VrefL/8192) or 0.5 LSb
- The 00 0000 0001 code is centered at (VrefH - VrefL/4096) or 1.0 LSb (B)
- The 10 0000 0000 code is centered at (2048 * (VrefH - VrefL)/4096) (C)
- An input voltage less than (1 * (VrefH - VrefL)/8192) converts as 00 0000 0000 (D)
- An input greater than (8192 * (VrefH - VrefL)/8192) converts as 11 1111 1111 (E)

Figure 22-25: Analog-to-Digital Transfer Function
22.10 ADC SAMPLING REQUIREMENTS

The analog input model of the 12-bit ADC is illustrated in Figure 22-26. The total acquisition time for the analog-to-digital conversion is a function of the internal circuit settling time and the holding capacitor charge time.

For the ADC module to meet its specified accuracy, the charge holding capacitor (CHOLD) must be allowed to fully charge to the voltage level on the analog input pin. The analog output source impedance (Rs), the interconnect impedance (Ric), and the internal sampling switch (Rss) impedance combine to directly affect the time required to charge the CHOLD. The combined impedance of the analog sources must therefore be small enough to fully charge (to within one-fourth LSB of the desired voltage) the holding capacitor within the selected sample time. The internal holding capacitor will be in the discharged state prior to each sample operation.

At least 1 TAD time period should be allowed between conversions for the acquisition time. Refer to the “Electrical Characteristics” chapter in the specific device data sheet for more information.

Figure 22-26: 12-bit ADC Analog Input Model

Note: The CPIN value depends on the device package and is not tested. The effect of the CPIN is negligible if Rs ≤ 5 kΩ.

Legend:
CPIN = Input capacitance
VT = Threshold voltage
Rss = Sampling switch resistance
Rs = Source resistance
CHOLD = Sample/hold capacitance
Ileakage = Leakage current at the pin due to various junctions

22.11 CONNECTION CONSIDERATIONS

Because the analog inputs employ Electrostatic Discharge (ESD) protection, they have diodes to VDD and VSS; therefore, the analog input must be between VDD and VSS. The presence of diodes is the reason why the analog pins cannot be 5V tolerant. If the input voltage exceeds this range by greater than 0.3V (either direction), one of the diodes becomes forward biased and it may damage the device if the input current specification exceeds.

An external RC filter is sometimes added for anti-aliasing of the input signal. The R (resistive) component should be selected to ensure that the acquisition time is met. Any external components connected (through high-impedance) to an analog input pin (capacitor, Zener diode, and so on) should have very little leakage current at the pin.
22.12 RELATED APPLICATION NOTES

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the PIC32 device family, but the concepts are pertinent and could be used with modification and possible limitations. The current application notes related to the 12-bit High-Speed Successive Approximation Register (SAR) Analog-to-Digital Converter (ADC) module are:

<table>
<thead>
<tr>
<th>Title</th>
<th>Application Note #</th>
</tr>
</thead>
<tbody>
<tr>
<td>No related application notes at this time.</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**Note:** Please visit the Microchip web site ([www.microchip.com](http://www.microchip.com)) for additional application notes and code examples for the PIC32 family of devices.
22.13  REVISION HISTORY

Revision A (June 2015)
This is the initial released version of the document.

Revision B (April 2016)
This revision includes the following updates:
• The ADINSEL<5:0> bits in the ADCCON3 register were updated (see Register 22-7)
• The DFMODE bit in the ADCFLTRx register was updated (see Register 22-17)
• Figure 22-18: “ADC Filter Comparisons Example” was added
• The ADC CVD Mode code example was updated (see Example 22-7)
• Minor updates to text and formatting were incorporated throughout the document

Revision C (September 2017)
This revision includes the following updates:
• Section 22.4.11 “Only-the-fly Update of SFRs” was removed.
• Minor updates to text and formatting were incorporated throughout the document

Revision D (November 2017)
This revision includes the following updates:
• The bit values for the ADC Clock Source (TCLK) bits, ADCSEL<1:0> (ADCCON3<31:30>),
were removed and a referral to the specific device data sheet for the ADC clock source
selections was added (see Register 22-3)
• The Sample and Conversion Time equations were updated (see Equation 22-1 and
Equation 22-2)

Revision E (September 2019)
This revision includes the following updates:
• Added missing mathematical symbols in Equation 22-1 and Equation 22-2
• Removed ADC6CFG configuration in Example 22-1
NOTES:
Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip’s Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable.”

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip’s code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications.

MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer’s risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Trademarks

The Microchip name and logo, the Microchip logo, Adaptec, AnyRate, AVR, AVR logo, AVR Freaks, BessTime, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, HELDO, IGLOO, JukeBlox, KeeLog, Kleer, LANcheck, LinkMD, maxSTylist, maxTXtouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, PackeTime, PIC, picoPower, PICSTART, PIC32 logo, PolarFire, Prochip Designer, QTouch, SAM-BA, SenGi, SpyNIC, SST, SST Logo, SuperFlash, Symmetricom, SymmServer, Tachyon, TempTrackr, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

APT, ClockWorks, The Embedded Control Solutions Company, EtherSynch, FlashTec, Hyper Speed Control, HyperLight Load, IntellIMOS, Libero, motorBench, mTouch, Powermite 3, Precision Edge, ProASIC, ProASIC Plus, ProASIC Plus logo, Quiet-Wire, SmartFusion, SyncWorld, Temux, TimeCesium, TimeHub, TimePictra, TimeProvider, Vite, WinPath, and ZL are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.


SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

The Adaptec logo, Frequency on Demand, Silicon Storage Technology, and Symmcom are registered trademarks of Microchip Technology Inc. in other countries.

GestC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

© 2015-2019, Microchip Technology Incorporated, All Rights Reserved.


For information regarding Microchip’s Quality Management Systems, please visit www.microchip.com/quality.