AN3382

ADC with Computation and Context Switching Using DMA

Introduction

Author: Keith Curtis, Microchip Technology Inc.

The ADC with Computation and Context Switching peripheral combines an ADC conversion with post-conversion operations, and allows the register configurations to be saved as contexts. These post-conversion operations can include averaging, accumulation, filter and comparison to preset thresholds. In addition, the ADC has the capability to do single or dual conversion and CVD capacitive measurements. These post-conversion operations significantly enhance the ADC's operation, but they become problematic when more than one channel of operation is required. Each channel will likely have its own specific configuration, average value, accumulated value, filter value and/or thresholds. Therefore, to efficiently operate more than one channel in a given application, the ADC register configurations for multiple input channels can be saved as contexts. The contexts can be read using the ADC module, or through the Direct Memory Access (DMA) module.

This application note will cover several examples of a context switching system for the ADC. This includes channels with different configurations, and channels with different average, accumulate and filter values, with and without DMA assistance.

Table of Contents

Introduction....1

1. ADC Overview....4
   - 1.1. 10/12-Bit ADC....4
   - 1.2. Computation....5
   - 1.3. Context Switching....5

2. DMA Overview....8
   - 2.1. Implementing ADC Context Saving Recovery....8

3. Dual DMA System....9
   - 3.1. Use Cases....10
   - 3.2. ADC Configuration....10
   - 3.3. DMA Channel Read Configuration....10
   - 3.4. DMA Channel Write Configuration....10
   - 3.5. Final System Configuration....10
   - 3.6. System Operation – Dual DMA....12

4. Three-Channel DMA....13
   - 4.1. Use Cases....14
   - 4.2. ADC Configuration....14
   - 4.3. DMA Channel Read Configuration....14
   - 4.4. DMA Write Channel Configuration....14
   - 4.5. Final System Configuration....15
   - 4.6. System Operation – Three Channels....15

5. Single-Channel DMA using ADC with Context Switching....17
   - 5.1. Use Cases....17
   - 5.2. ADC Configuration....18
   - 5.3. DMA Channel Read Configuration....18
   - 5.4. Final System Configuration....18
   - 5.5. System Operation – ADC with Context Sequencer....18

6. Conclusion....20

The Microchip Website....21

Product Change Notification Service....21

Customer Support....21

Product Identification System....22

Microchip Devices Code Protection Feature....22

Legal Notice....22

Trademarks....22

Quality Management System....23

Worldwide Sales and Service.....24

1. ADC Overview

To start, it is useful to review the general capabilities of the ADC, post-conversion computation, and the new context switching feature. This is not meant to be an in-depth discussion of how to configure and use the ADC; rather, it is to identify the modes and registers utilized for configuration, conversion, status, computational functions and context switching.

1.1 10/12-Bit ADC

The ADC is a 10-bit or 12-bit Successive Approximation Register (SAR) converter with some additions. These additions include:

- Multichannel analog input multiplexer
- Multichannel analog reference multiplexer
- Multisource input clock multiplexer
- Multisource conversion trigger multiplexer
- Acquisition timer
- Precharge timer
- Variable size Sample-and-Hold (S/H) capacitor
- Double conversion sequencer
- Capacitive Voltage Divider (CVD) sequencer (capacitive sensor converter)
- Guard ring converter for CVD

Together, these additions form the basic ADC front end for the ADC. These options are configured in the ADC Configuration registers listed in Table 1-1.

Table 1-1. Configuration Registers

Configuration Register	Description
ADPCH	ADC Input Select
ADCON0	ADC Configuration 0
ADCON1	ADC Configuration 1
ADREF	ADC Reference Select
ADCLK	ADC Clock Select
ADCAP	Sample Capacitance Select
ADACQ	ADC Acquisition Time Select
ADPRE(16)	ADC Precharge Time Select
ADACT	ADC Conversion Trigger Select

In addition to the Configuration registers, there are two Conversion Result registers listed in Table 1-2.

Table 1-2. Data Registers

Data Register	Description
ADRES(16)	16-Bit Result Register
ADPREV(16)	16-Bit Previous Result Register

The ADRES is the result of the most recent conversion and the ADPREV register data are the result of the previous conversion. Two registers are required to hold the results of a double conversion.

Note: The Conversion Result registers are 16-bit to hold the 10-/12-bit result of the ADC conversion. Both right and left justification/format are available by configuring the FM bit in the ADCON0 register.

1.2 Computation

The computational portion of the ADC is capable of the following:

- Basic: conversion with no post-conversion processing
- Accumulate: which accumulates the results of all subsequent conversions
- Average: averages each sample and compares to the thresholds' every X sample
- Burst Average: repeatedly samples X values and compares to the thresholds
- Filter: Low-pass filters the samples, and after X samples, compares with thresholds

These computational options are configured by the registers listed in Table 1-3.

Table 1-3. Configuration Registers

Configuration Register	Description
ADCON2	Computation Configuration 2 Register
ADCON3	Computation Configuration 3 Register
ADRPT	ADC Repeat Setting Register
ADSTPT(16)	ADC Threshold Set Point Register
ADLTH(16)	Lower Threshold Register
ADUTH(16)	Upper Threshold Register

In addition to the Configuration registers, there are context and status registers for computation listed in Tables 1-4 and 1-5.

Table 1-4. Status Registers

Status Register	Description
ADERR(16)	16-Bit Error Status Register
ADSTAT	Computation Status Register

Table 1-5. Data Registers

Data Register	Description
ADACC(16)	16-Bit Accumulation Register
ADCNT	Repetition Count Register
ADFLTR(16)	Current Filter Value Register

1.3 Context Switching

The context switching feature of the ADC allows the configuration of multiple channels with differing computational features, status, input channels, accumulated values, filter values, thresholds, plus current and previous results. This context switching feature gives designers the equivalent of multiple ADC peripherals with individual computational functions. Further, the actual context switch can be triggered at the end of the ADC conversion, allowing the polling of the multiple inputs in response to a hardware CIP trigger event.

The CIP trigger automated polling is a combination of the existing hardware ADC trigger and the context sequencer. The sequencer manages both software access to all the shadowed Configuration registers and the automated context switching between ADC trigger events. Additional access is possible using DMA to access the Special Function Register map (DMA Access Only).

Note: Information on accessing the context registers using the DMA peripheral can be found in the device data sheet under the DMA peripheral chapter.

Figure 1-1. Special Function Register Map (DMA Access Only)

40FFh	-	40DFh	CX4_ADPREH_M1	40BFh	CX3_ADPREH_M1	409Fh	CX2_ADPREH_M1	407Fh	CX1_ADPREH_M1	405Fh	-	403Fh	-	401Fh	PWM4S1P2H_M1
40FEh	-	40DEh	CX4_ADPREL_M1	40BEh	CX3_ADPREL_M1	409Eh	CX2_ADPREL_M1	407Eh	CX1_ADPREL_M1	405Eh	-	403Eh	-	401Eh	PWM4S1P2L_M1
40FDh	-	40DDh	CX4_ADRESH_M1	40BDh	CX3_ADRESH_M1	409Dh	CX2_ADRESH_M1	407Dh	CX1_ADRESH_M1	405Dh	-	403Dh	-	401Dh	PWM4S1P1H_M2
40FCh	-	40DCh	CX4_ADRESL_M1	40BCh	CX3_ADRESL_M1	409Ch	CX2_ADRESL_M1	407Ch	CX1_ADRESL_M1	405Ch	-	403Ch	-	401Ch	PWM4S1P1L_M2
40FBh	-	40DBh	CX4_ADPCH_M1	40BBh	CX3_ADPCH_M1	409Bh	CX2_ADPCH_M1	407Bh	CX1_ADPCH_M1	405Bh	-	403Bh	-	401Bh	PWM3S1P2H_M1
40FAh	-	40DAh	CX4_ADCLK_M1	40BAh	CX3_ADCLK_M1	409Ah	CX2_ADCLK_M1	407Ah	CX1_ADCLK_M1	405Ah	-	403Ah	-	401Ah	PWM3S1P2L_M1
40F9h	-	40D9h	CX4_ADACT_M1	40B9h	CX3_ADACT_M1	4099h	CX2_ADACT_M1	4079h	CX1_ADACT_M1	4059h	-	4039h	-	4019h	PWM3S1P1H_M2
40F8h	-	40D8h	CX4_ADREF_M1	40B8h	CX3_ADREF_M1	4098h	CX2_ADREF_M1	4078h	CX1_ADREF_M1	4058h	-	4038h	-	4018h	PWM3S1P1L_M2
40F7h	-	40D7h	CX4_ADCON3_M1	40B7h	CX3_ADCON3_M1	4097h	CX2_ADCON3_M1	4077h	CX1_ADCON3_M1	4057h	-	4037h	PWM4PRH_M1	4017h	PWM2S1P2H_M1
40F6h	ADRESH_M1	40D6h	CX4_ADCON2_M1	40B6h	CX3_ADCON2_M1	4096h	CX2_ADCON2_M1	4076h	CX1_ADCON2_M1	4056h	-	4036h	PWM4PRL_M1	4016h	PWM2S1P2L_M1
40F5h	ADRESL_M1	40D5h	CX4_ADCON1_M1	40B5h	CX3_ADCON1_M1	4095h	CX2_ADCON1_M1	4075h	CX1_ADCON1_M1	4055h	-	4035h	PWM4S1P2H_M2	4015h	PWM2S1P1H_M2
40F4h	ADPCH_M1	40D4h	CX4_ADCON0_M1	40B4h	CX3_ADCON0_M1	4094h	CX2_ADCON0_M1	4074h	CX1_ADCON0_M1	4054h	-	4034h	PWM4S1P2L_M2	4014h	PWM2S1P1L_M2
40F3h	ADCAP_M1	40D3h	CX4_ADCAP_M1	40B3h	CX3_ADCAP_M1	4093h	CX2_ADCAP_M1	4073h	CX1_ADCAP_M1	4053h	-	4033h	PWM4S1P1H_M3	4013h	PWM1S1P2H_M1
40F2h	ADACQH_M1	40D2h	CX4_ADACQH_M1	40B2h	CX3_ADACQH_M1	4092h	CX2_ADACQH_M1	4072h	CX1_ADACQH_M1	4052h	-	4032h	PWM4S1P1L_M3	4012h	PWM1S1P2L_M1
40F1h	ADACQL_M1	40D1h	CX4_ADACQL_M1	40B1h	CX3_ADACQL_M1	4091h	CX2_ADACQL_M1	4071h	CX1_ADACQL_M1	4051h	-	4031h	PWM3PRH_M1	4011h	PWM1S1P1H_M2
40F0h	ADPREVH_M1	40D0h	CX4_ADPREVH_M1	40B0h	CX3_ADPREVH_M1	4090h	CX2_ADPREVH_M1	4070h	CX1_ADPREVH_M1	4050h	-	4030h	PWM3PRL_M1	4010h	PWM1S1P1L_M2
40EFh	ADPREVL_M1	40CFh	CX4_ADPREVL_M1	40AFh	CX3_ADPREVL_M1	408Fh	CX2_ADPREVL_M1	406Fh	CX1_ADPREVL_M1	404Fh	-	402Fh	PWM3S1P2H_M2	400Fh	-
40EEh	ADRPT_M1	40CEh	CX4_ADRPT_M1	40AEh	CX3_ADRPT_M1	408Eh	CX2_ADRPT_M1	406Eh	CX1_ADRPT_M1	404Eh	-	402Eh	PWM3S1P2L_M2	400Eh	-
40EDh	ADCNT_M1	40CDh	CX4_ADCNT_M1	40ADh	CX3_ADCNT_M1	408Dh	CX2_ADCNT_M1	406Dh	CX1_ADCNT_M1	404Dh	-	402Dh	PWM3S1P1H_M3	400Dh	PWM4S1P1H_M1
40ECh	ADACCU_M1	40CCh	CX4_ADACCU_M1	40ACh	CX3_ADACCU_M1	408Ch	CX2_ADACCU_M1	406Ch	CX1_ADACCU_M1	404Ch	-	402Ch	PWM3S1P1L_M3	400Ch	PWM4S1P1L_M1
40CBh	ADACCH_M1	40CBh	CX4_ADACCH_M1	40ABh	CX3_ADACCH_M1	408Bh	CX2_ADACCH_M1	406Bh	CX1_ADACCH_M1	404Bh	-	402Bh	PWM2PRH_M1	400Bh	PWM3S1P1H_M1
40EAh	ADACCL_M1	40CAh	CX4_ADACCL_M1	40AAh	CX3_ADACCL_M1	408Ah	CX2_ADACCL_M1	406Ah	CX1_ADACCL_M1	404Ah	-	402Ah	PWM2PRL_M1	400Ah	PWM3S1P1L_M1
40E9h	ADFLTRH_M1	40C9h	CX4_ADFLTRH_M1	40A9h	CX3_ADFLTRH_M1	4089h	CX2_ADFLTRH_M1	4069h	CX1_ADFLTRH_M1	4049h	-	4029h	PWM2S1P2H_M2	4009h	PWM2S1P1H_M1
40E8h	ADFLTRL_M1	40C8h	CX4_ADFLTRL_M1	40A8h	CX3_ADFLTRL_M1	4088h	CX2_ADFLTRL_M1	4068h	CX1_ADFLTRL_M1	4048h	T6PR_M1	4028h	PWM2S1P2L_M2	4008h	PWM2S1P1L_M1
40E7h	ADSTPTH_M1	40C7h	CX4_ADSTPTH_M1	40A7h	CX3_ADSTPTH_M1	4087h	CX2_ADSTPTH_M1	4067h	CX1_ADSTPTH_M1	4047h	CCPR3H_M2	4027h	PWM2S1P1H_M3	4007h	PWM1S1P1H_M1
40E6h	ADSTPTL_M1	40C6h	CX4_ADSTPTL_M1	40A6h	CX3_ADSTPTL_M1	4086h	CX2_ADSTPTL_M1	4066h	CX1_ADSTPTL_M1	4046h	CCPR3L_M2	4026h	PWM2S1P1L_M3	4006h	PWM1S1P1L_M1
40E5h	ADERRH_M1	40C5h	CX4_ADERRH_M1	40A5h	CX3_ADERRH_M1	4085h	CX2_ADERRH_M1	4065h	CX1_ADERRH_M1	4045h	T4PR_M1	4025h	PWM1PRH_M1	4005h	CCPR3H_M1
40E4h	ADERRL_M1	40C4h	CX4_ADERRL_M1	40A4h	CX3_ADERRL_M1	4084h	CX2_ADERRL_M1	4064h	CX1_ADERRL_M1	4044h	CCPR2H_M2	4024h	PWM1PRL_M1	4004h	CCPR3L_M1
40E3h	ADUTHH_M1	40C3h	CX4_ADUTHH_M1	40A3h	CX3_ADUTHH_M1	4083h	CX2_ADUTHH_M1	4063h	CX1_ADUTHH_M1	4043h	CCPR2L_M2	4023h	PWM1S1P2H_M2	4003h	CCPR2H_M1
40E2h	ADUTHL_M1	40C2h	CX4_ADUTHL_M1	40A2h	CX3_ADUTHL_M1	4082h	CX2_ADUTHL_M1	4062h	CX1_ADUTHL_M1	4042h	T2PR_M1	4022h	PWM1S1P2L_M2	4002h	CCPR2L_M1
40E1h	ADLTHH_M1	40C1h	CX4_ADLTHH_M1	40A1h	CX3_ADLTHH_M1	4081h	CX2_ADLTHH_M1	4061h	CX1_ADLTHH_M1	4041h	CCPR1H_M2	4021h	PWM1S1P1H_M3	4001h	CCPR1H_M1
40E0h	ADLTHL_M1	40C0h	CX4_ADLTHL_M1	40A0h	CX3_ADLTHL_M1	4080h	CX2_ADLTHL_M1	4060h	CX1_ADLTHL_M1	4040h	CCPR1L_M2	4020h	PWM1S1P1L_M3	4000h	CCPR1L_M1

41FFh	-	41DFh	-	41BFh	-	419Fh	DMAnAIRQ_DMA7	417Fh	DMAnSPTRH_DMA6	415Fh	DMAnDPTRL_DMA5	413Fh	DMAnSSAH_DMA3	411Fh	DMAnDSAH_DMA2
41FEh	-	41DEh	-	41BEh	-	419Eh	DMAnCON1_DMA7	417Eh	DMAnSPTRL_DMA6	415Eh	DMAnDCNTH_DMA5	413Eh	DMAnSSAL_DMA3	411Eh	DMAnDSAL_DMA2
41FDh	-	41DDh	-	41BDh	-	419Dh	DMAnCON0_DMA7	417Dh	DMAnSCNTH_DMA6	415Dh	DMAnDCNTL_DMA5	413Dh	DMAnSSZH_DMA3	411Dh	DMAnDSZH_DMA2
41FCh	-	41DCh	-	41BCh	-	419Ch	DMAnSSAU_DMA7	417Ch	DMAnSCNTL_DMA6	415Ch	DMAnBUF_DMA5	413Ch	DMAnSSZL_DMA3	411Ch	DMAnDSZL_DMA2
41FBh	TMR5H_M1	41DBh	-	41BBh	-	419Bh	DMAnSSAH_DMA7	417Bh	DMAnDSAH_DMA6	415Bh	DMAnSIRQ_DMA4	413Bh	DMAnSPTRU_DMA3	411Bh	DMAnDPTRH_DMA2
41FAh	TMR5L_M1	41DAh	-	41BAh	-	419Ah	DMAnSSAL_DMA7	417Ah	DMAnDSAL_DMA6	415Ah	DMAnAIRQ_DMA4	413Ah	DMAnSPTRH_DMA3	411Ah	DMAnDPTRL_DMA2
41F9h	TMR3H_M1	41D9h	-	41B9h	-	4199h	DMAnSSZH_DMA7	4179h	DMAnDSZH_DMA6	4159h	DMAnCON1_DMA4	4139h	DMAnSPTRL_DMA3	4119h	DMAnDCNTH_DMA2
41F8h	TMR3L_M1	41D8h	-	41B8h	-	4198h	DMAnSSZL_DMA7	4178h	DMAnDSZL_DMA6	4158h	DMAnCON0_DMA4	4138h	DMAnSCNTH_DMA3	4118h	DMAnDCNTL_DMA2
41F7h	TMR1H_M1	41D7h	-	41B7h	DMAnSIRQ_DMA8	4197h	DMAnSPTRU_DMA7	4177h	DMAnDPTRH_DMA6	4157h	DMAnSSAU_DMA4	4137h	DMAnSCNTL_DMA3	4117h	DMAnBUF_DMA2
41F6h	TMR1L_M1	41D6h	-	41B6h	DMAnAIRQ_DMA8	4196h	DMAnSPTRH_DMA7	4176h	DMAnDPTRL_DMA6	4156h	DMAnSSAH_DMA4	4136h	DMAnDSAH_DMA3	4116h	DMAnSIRQ_DMA1
41F5h	-	41D5h	-	41B5h	DMAnCON1_DMA8	4195h	DMAnSPTRL_DMA7	4175h	DMAnDCNTH_DMA6	4155h	DMAnSSAL_DMA4	4135h	DMAnDSAL_DMA3	4115h	DMAnAIRQ_DMA1
41F4h	-	41D4h	-	41B4h	DMAnCON0_DMA8	4194h	DMAnSCNTH_DMA7	4174h	DMAnDCNTL_DMA6	4154h	DMAnSSZH_DMA4	4134h	DMAnDSZH_DMA3	4114h	DMAnCON1_DMA1
41F3h	-	41D3h	-	41B3h	DMAnSSAU_DMA8	4193h	DMAnSCNTL_DMA7	4173h	DMAnBUF_DMA6	4153h	DMAnSSZL_DMA4	4133h	DMAnDSZL_DMA3	4113h	DMAnCON0_DMA1
41F2h	-	41D2h	-	41B2h	DMAnSSAH_DMA8	4192h	DMAnDSAH_DMA7	4172h	DMAnSIRQ_DMA5	4152h	DMAnSPTRU_DMA4	4132h	DMAnDPTRH_DMA3	4112h	DMAnSSAU_DMA1
41F1h	-	41D1h	-	41B1h	DMAnSSAL_DMA8	4191h	DMAnDSAL_DMA7	4171h	DMAnAIRQ_DMA5	4151h	DMAnSPTRH_DMA4	4131h	DMAnDPTRL_DMA3	4111h	DMAnSSAH_DMA1
41F0h	-	41D0h	-	41B0h	DMAnSSZH_DMA8	4190h	DMAnDSZH_DMA7	4170h	DMAnCON1_DMA5	4150h	DMAnSPTRL_DMA4	4130h	DMAnDCNTH_DMA3	4110h	DMAnSSAL_DMA1
41EFh	-	41CFh	-	41AFh	DMAnSSZL_DMA8	418Fh	DMAnDSZL_DMA7	416Fh	DMAnCON0_DMA5	414Fh	DMAnSCNTH_DMA4	412Fh	DMAnDCNTL_DMA3	410Fh	DMAnSSZH_DMA1
41EEh	-	41CEh	-	41AEh	DMAnSPTRU_DMA8	418Eh	DMAnDPTRH_DMA7	416Eh	DMAnSSAU_DMA5	414Eh	DMAnSCNTL_DMA4	412Eh	DMAnBUF_DMA3	410Eh	DMAnSSZL_DMA1
41EDh	-	41CDh	-	41ADh	DMAnSPTRH_DMA8	418Dh	DMAnDPTRL_DMA7	416Dh	DMAnSSAH_DMA5	414Dh	DMAnDSAH_DMA4	412Dh	DMAnSIRQ_DMA2	410Dh	DMAnSPTRU_DMA1
41ECh	-	41CCh	-	41ACh	DMAnSPTRL_DMA8	418Ch	DMAnDCNTH_DMA7	416Ch	DMAnSSAL_DMA5	414Ch	DMAnDSAL_DMA4	412Ch	DMAnAIRQ_DMA2	410Ch	DMAnSPTRH_DMA1
41CBh	-	41CBh	-	41ABh	DMAnSCNTH_DMA8	418Bh	DMAnDCNTL_DMA7	416Bh	DMAnSSZH_DMA5	414Bh	DMAnDSZH_DMA4	412Bh	DMAnCON1_DMA2	410Bh	DMAnSPTRL_DMA1
41EAh	-	41CAh	-	41AAh	DMAnSCNTL_DMA8	418Ah	DMAnBUF_DMA7	416Ah	DMAnSSZL_DMA5	414Ah	DMAnDSZL_DMA4	412Ah	DMAnCON0_DMA2	410Ah	DMAnSCNTH_DMA1
41E9h	-	41C9h	-	41A9h	DMAnDSAH_DMA8	4189h	DMAnSIRQ_DMA6	4169h	DMAnSPTRU_DMA5	4149h	DMAnDPTRH_DMA4	4129h	DMAnSSAU_DMA2	4109h	DMAnSCNTL_DMA1
41E8h	-	41C8h	-	41A8h	DMAnDSAL_DMA8	4188h	DMAnAIRQ_DMA6	4168h	DMAnSPTRH_DMA5	4148h	DMAnDPTRL_DMA4	4128h	DMAnSSAH_DMA2	4108h	DMAnDSAH_DMA1
41E7h	-	41C7h	-	41A7h	DMAnDSZH_DMA8	4187h	DMAnCON1_DMA6	4167h	DMAnSPTRL_DMA5	4147h	DMAnDCNTH_DMA4	4127h	DMAnSSAL_DMA2	4107h	DMAnDSAL_DMA1
41E6h	-	41C6h	-	41A6h	DMAnDSZL_DMA8	4186h	DMAnCON0_DMA6	4166h	DMAnSCNTH_DMA5	4146h	DMAnDCNTL_DMA4	4126h	DMAnSSZH_DMA2	4106h	DMAnDSZH_DMA1
41E5h	-	41C5h	-	41A5h	DMAnDPTRH_DMA8	4185h	DMAnSSAU_DMA6	4165h	DMAnSCNTL_DMA5	4145h	DMAnBUF_DMA4	4125h	DMAnSSZL_DMA2	4105h	DMAnDSZL_DMA1
41E4h	-	41C4h	-	41A4h	DMAnDPTRL_DMA8	4184h	DMAnSSAH_DMA6	4164h	DMAnDSAH_DMA5	4144h	DMAnSIRQ_DMA3	4124h	DMAnSPTRU_DMA2	4104h	DMAnDPTRH_DMA1
41E3h	IOCEF_M1	41C3h	-	41A3h	DMAnDCNTH_DMA8	4183h	DMAnSSAL_DMA6	4163h	DMAnDSAL_DMA5	4143h	DMAnAIRQ_DMA3	4123h	DMAnSPTRH_DMA2	4103h	DMAnDPTRL_DMA1
41E2h	IOCCF_M1	41C2h	-	41A2h	DMAnDCNTL_DMA8	4182h	DMAnSSZH_DMA6	4162h	DMAnDSZH_DMA5	4142h	DMAnCON1_DMA3	4122h	DMAnSPTRL_DMA2	4102h	DMAnDCNTH_DMA1
41E1h	IOCBF_M1	41C1h	-	41A1h	DMAnBUF_DMA8	4181h	DMAnSSZL_DMA6	4161h	DMAnDSZL_DMA5	4141h	DMAnCON0_DMA3	4121h	DMAnSCNTH_DMA2	4101h	DMAnDCNTL_DMA1
41E0h	IOCAF_M1	41C0h	-	41A0h	DMAnSIRQ_DMA7	4180h	DMAnSPTRU_DMA6	4160h	DMAnDPTRH_DMA5	4140h	DMAnSSAU_DMA3	4120h	DMAnSCNTL_DMA2	4100h	DMAnBUF_DMA1

The sequencer is configured using the ADCTX register and the four ADCSELx registers. The ADCTX register handles software access to the individual context registers. The ADCSELx registers determine if the specific configuration is part of the polling sequence, and if a threshold interrupt will halt the sequencer. The start of the sequencer is initiated by setting the GO bit in the ADCCON0 register or by a selected hardware trigger. See the ADC peripheral chapter of the device data sheet for additional configuration and control information concerning the ADC context switching sequencer.

Table 1-6. Register Summary for PIC18F57Q84 ADC

Address	Name	Bit Pos.								
0x00-0x03D7	Reserved									
0x03D8	ADCP	7:0	CPON							CPRDY
0x03D9	ADLTH	7:0	LTH[7:0]							
		15:8	LTH[15:8]							
0x03DB	ADUTH	7:0	UTH[7:0]							
		15:8	UTH[15:8]							

.....continued

Address	Name	Bit Pos.								
0x03DD	ADERR	7:0	ERR[7:0]							
		15:8	ERR[15:8]							
0x03DF	ADSTPT	7:0	STPT[7:0]							
		15:8	STPT[15:8]							
0x03E1	ADFLTR	7:0	FLTR[7:0]							
		15:8	FLTR[15:8]							
0x03E3	ADACC	7:0	ACC[7:0]							
		15:8	ACC[15:8]							
		23:16							ACC[17:16]	
0x03E6	ADCNT	7:0	CNT[7:0]							
0x03E7	ADRPT	7:0	RPT[7:0]							
0x03E8	ADPREV	7:0	PREV[7:0]							
		15:8	PREV[15:8]							
0x03EA	ADRES	7:0	RES[7:0]							
		15:8	RES[15:8]							
0x03EC	ADPCH	7:0			PCH[5:0]					
0x03ED	Reserved									
0x03EE	ADACQ	7:0	ACQ[7:0]							
		15:8				ACQ[12:8]				
0x03F0	ADCAP	7:0				CAP[4:0]				
0x03F1	ADPRE	7:0	PRE[7:0]							
						PRE[12:8]				
0x03F3	ADCON0	7:0	ON	CONT		CS		FM		GO
0x03F4	ADCON1	7:0	PPOL	IPEN	GPOL					DSEN
0x03F5	ADCON2	7:0	PSIS	CRS[2:0]			ACLR	MD[2:0]		
0x03F6	ADCON3	7:0		CALC[2:0]			SOI	TMD[2:0]		
0x03F7	ADSTAT	7:0	AOV	UTHR	LTHR	MATH		STAT[2:0]		
0x03F8	ADREF	7:0				NREF			PREF[1:0]	
0x03F9	ADACT	7:0			ACT[5:0]					
0x03FA	ADCLK	7:0			CS[5:0]					
0x03FB	ADCTX	7:0	CTXSW						CTX[1:0]	
0x03FC	ADCSEL1	7:0	CHEN	SSI						
0x03FD	ADCSEL2	7:0	CHEN	SSI						
0x03FE	ADCSEL3	7:0	CHEN	SSI						
0x03FF	ADCSEL4	7:0	CHEN	SSI						

2. DMA Overview

The Direct Memory Access (DMA) peripheral is a generic data transfer peripheral capable of moving data from the program memory, EEPROM memory, GPR memory and SFR memory to the GPR and SFR memory. This transfer can be triggered by software or by a hardware trigger.

To configure the DMA, the following configurations must be made:

1. The source type of memory must be configured in the DMAxCON1 register using the SMR[4:3] bits.
2. Load the DMAxSSAL, DMAxSSAH and DMAxSSAU registers with the source address.
3. Load the DMAxDSAL and DMAxDSAH registers with the destination address.
4. Load the DMAxSSZL and DMAxSSZH registers with the size of the source.
5. Load the DMAxDSZL and DMAxDSZH registers with the size of the destination.
6. Select the transfer trigger source by loading the DMAxSIRQ register.
7. If needed, select an abort trigger by loading the DMAxAIRQ register.
8. Configure the DMA automation using the SMODE[1:0] and DMODE[1:0] bits in the DMAxCON1 register. This will select fixed, post-increment or post-decrement of the Address Pointer.
9. Set the appropriate DMA termination select using the SSTP or DSTP bits in the DMAxCON1 register.
10. Enable any interrupts required. Note that this application note will use the interrupts to trigger additional actions.
11. Configure the transfer priority by configuring the arbiter system in the microcontroller. Remember to set the PRLOCK bit to enable DMA operation.
12. Enable the DMA by setting the EN bit in the DMAxCON0 register.

Note: For in-depth configuration information, refer to the device data sheet and the *"Configuring the DMA Peripheral"* Technical Brief (DS90003242).

2.1 Implementing ADC Context Saving Recovery

For this application note, we will look at three configurations:

- The simplest system that uses only two DMA channels and a larger context storage array.
- A more robust system that uses three DMA channels and a smaller context storage array.
- A system that uses the new ADC with Computation and Context Switching and a single DMA channel with a data only array.

The two DMA system is based on a system with multiple ADC channels with completely different configurations for computation. The larger context array allows the firmware to modify the configuration of the individual ADC channels on the fly. This is useful if thresholds or modes need to change based on external requirements. The firmware can just modify the context array and results in a modification of the computational configuration. Examples of this would include: mTouch® systems requiring variable thresholds, sensor filtering applications that need to respond to changing noise conditions, or systems with different modes of operation requiring changes into ADC modes such as accumulate or filtering.

The three DMA system is also based on a system with multiple ADC channels with different a configuration for computation. The smaller context array minimizes the GPR memory requirements, but locks the system concerning the configuration of the ADC channels. This is useful if noise or potential data corruption is a problem in that the configuration of the ADC is fixed by program memory.

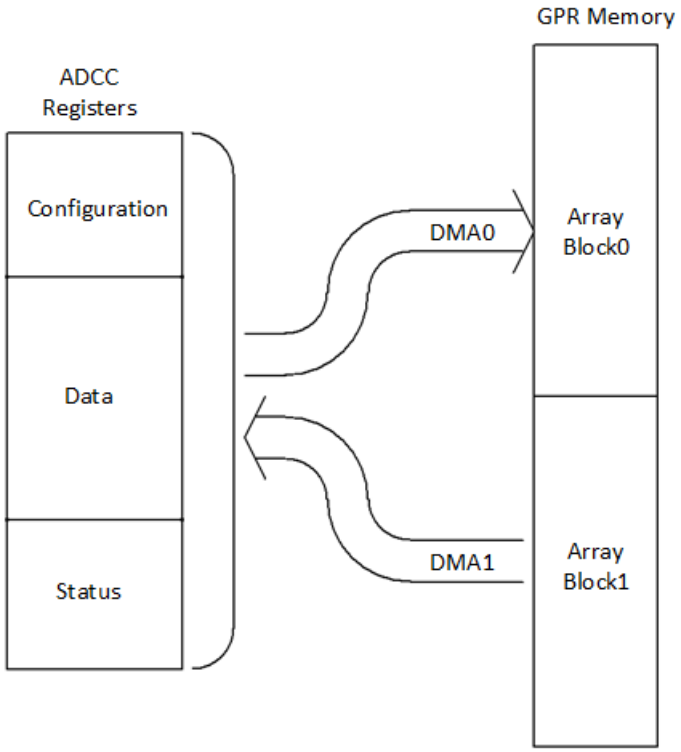
The ADC with Context Switching is based on a system with multiple ADC channels with different configurations for computation as well. The tiny data array is the minimum possible GPR memory requirement, but it does not lock the system concerning the configuration of the ADC channels, like the three DMA system. The configuration of the individual channels can still be modified on the fly, but it does require that the firmware access the registers through the ADCTX register. While this is not as secure against noise as the three DMA system, it is still more secure than the two DMA system and uses the smallest number of GPR registers.

3. Dual DMA System

The simplest system is a design that uses two DMA channels; the first copies out the complete current register map for the ADC (context, data and status) and the second, which copies in the next register map for the ADC. The transfers are triggered by the ADC interrupt signaling the end of the conversion or computation. Arbitration priority is given to the DMA channel handling the register read (Priority 0), followed by the DMA channel handling the next context write (Priority 1). The system interrupt and main code follow as Priority 2 and 3. See Figure 3-1.

The attached project demonstrates this system using a PIC18F57Q43, a Curiosity Nano Adapter and three sensor Click boards™. The three sensor Click boards are a force sensing click, an ambient light sensor click and a UV sensor click. Each sensor board has its own ADC configuration and the dual DMA system cycles between the sensors, swapping in and out of the context for each. TMR0 drives the ADC conversion input and the ADC completion interrupt triggers the two DMA channels. The first DMA copies out the previous context and ADC configuration. The second DMA copies in the next context ADC configuration. Both DMA channels are triggered by the ADC done signal, with the arbiter giving priority to the first DMA channel, next priority to the second DMA, with third and forth priority assigned to the interrupt and main line code.

Figure 3-1. Data Flow for the Dual DMA Context Save System



The read DMA has the highest priority so that when triggered, it will complete the register read first. Once it completes, priority will fall to the write DMA, which will then complete the write of the next context. Then finally, priority will be given back to the system interrupts and main code. This simplifies the sequential nature of the read and write DMA channels, while making sure the transfers occur prior to the next ADC trigger.

The system will require 34 memory locations for each channel of the ADC (configuration, context and data) if computational and threshold features are used. If just a simple ADC is desired, then only 18 registers (configuration and data) per channel are required. Only 13 memory locations are required if the ADC is configured for single conversion.

Note: The ADCON2, ADCON3 and ADSTAT registers are included in the register count for non-computational context saving, even though they provide configuration for computation operation. Their location in memory is interspersed with the ADC Configuration registers and it is not possible to get the DMA to skip over random registers in a series, so they are included.

3.1 Use Cases

The two DMA system does not allow on the fly changes to the ADC configuration while the sampling system is operating. However, it does protect the system from potential modifications to the ADC by noise corruption of the GPR memory, making it more robust. The system also uses a smaller array of GPR memory locations to store the ADC data and status.

3.2 ADC Configuration

To configure the system, start with the ADC:

1. Determine the number of registers to be saved and the base address of the registers.
2. Setup an array of variables for the context save/restore:
   – The number of bytes of storage required will be: (number of channels) x (number of registers).
   – This will require that the variable space be defined in RAM banks that contain contiguous GPR memory.
3. Create a routine to load the variable space with the configuration values for each ADC channel:
   – Include the ADC trigger.
4. Configure the ADC trigger source.

3.3 DMA Channel Read Configuration

Once the ADC is configured, configure the read DMA channel:

1. Configure the source for GPR/SFR memory and the starting address of the ADC registers.
2. Configure the source size for the number of registers to be read.
3. Set the SSTP bit in the DMAxCON1 register to terminate the DMA when the source counter reloads.
4. Configure the source for post-increment.
5. Configure the destination with the starting address of the starting address of the array variable.
6. Configure the destination size for the size of the array variable.
7. Clear the DSTP bit in the DMAxCON1 register.
8. Configure the source for post-increment.
9. Set the DMA trigger source for the ADC conversion complete flag.
10. Configure the system arbitrator to give the read DMA the highest priority (0).

3.4 DMA Channel Write Configuration

Next, configure the DMA write channel:

1. Configure the source for GPR/SFR memory and the starting address of the starting address of the array variable.
2. Configure the source size for the size of the array variable.
3. Clear the SSTP bit in the DMAxCON1 register.
4. Configure the source for post-increment.
5. Configure the destination for the starting address of the ADC registers.
6. Configure the destination size for the number of registers to be written.
7. Set the DSTP bit in the DMAxCON1 register to terminate the DMA when the destination counter reloads.
8. Configure the destination for post-increment.
9. Set the DMA trigger source for the ADC conversion complete flag.
10. Configure the system arbitrator to give the read DMA the highest priority (1).

3.5 Final System Configuration

For the final system configuration:

1. Set the system arbiter for interrupt at level (2) and main for level (3).
2. Set the PRLOCK in the system arbiter to prevent inadvertent modification of system priority. See Examples 3-1 and 3-2.
3. Manually (in software) trigger the first DMA write channel to preload the ADC for the first conversion.
4. Start the peripheral supplying the ADC trigger.

Example 3-1. Priority Lock Sequence

```
INTCON0bits.GIE = 0;         // Disable Interrupts;
PRLOCK = 0x55;
PRLOCK = 0xAA;
PRLOCKbits.PRLOCKED = 1;     // Grant memory access to peripherals;
INTCON0bits.GIE = 1;         // Enable Interrupts;
```

Example 3-2. Priority Unlock Sequence

```
INTCON0bits.GIE = 0;         // Disable Interrupts;
PRLOCK = 0x55;
PRLOCK = 0xAA;
PRLOCKbits.PRLOCKED = 0;     // Allow changing priority settings;
INTCON0bits.GIE = 1;         // Enable Interrupts;
```

3.6 System Operation – Dual DMA

Once started, the system should perform an ADC conversion, read the registers of the ADC and store the data in GPR memory. The system will then read the next set of data from GPR memory and write them to the ADC registers. At this point, the system is waiting until the next ADC trigger event. This sequence should continue until all of the channels have been converted. Then, the DMA will roll over and start through the ADC input channels in a continuous cycle until the DMA is disabled, the ADC is disabled or the system is placed in Sleep.

To access the conversion data, the user simply reads the appropriate locations within the context array. If needed, the ADC conversion complete interrupt can be enabled, and when the conversion is complete, the ISR can access the data in the context array.
Note: Because the ISR has a lower priority than the DMA in the system arbiter, the most recent conversion data will no longer be present in the ADRES registers and can only be accessed in the context array memory.

Threshold data can also be accessed in the context array memory in response to a threshold interrupt.

Another option is to set the interrupt to the highest priority in the system arbiter. This will allow both ADC conversion complete and threshold interrupts to access the conversion data directly in the ADC registers. Then, when the interrupt completes, the DMA will automatically take control and perform the read and write context function.

Note: It is not possible to differentiate ADC interrupts from other system interrupts, so using the system arbiter to set interrupts at a higher priority than DMA could potentially delay the completion of the read and store of the ADC context due to a stack up of ISR calls. This could delay the context switch, such that the next trigger of the ADC could fire before the completion of the context switch, so care should be taken with this approach.

Project `Dual_DMA.x` contains the configuration for this example. The reader is directed to the example and the ADC/DMA peripheral configuration information in the device data sheet for additional setup information.

4. Three-Channel DMA

A more robust system is a design that uses three DMA channels; the first copies out the data register map for the ADC. The second DMA copies in the complete Configuration register map for next ADC channel from a constant array in program Flash memory. The third DMA copies in the data register map for the next ADC channel. The DMA transfers are triggered by the ADC interrupt, signaling the end of the conversion or computation.

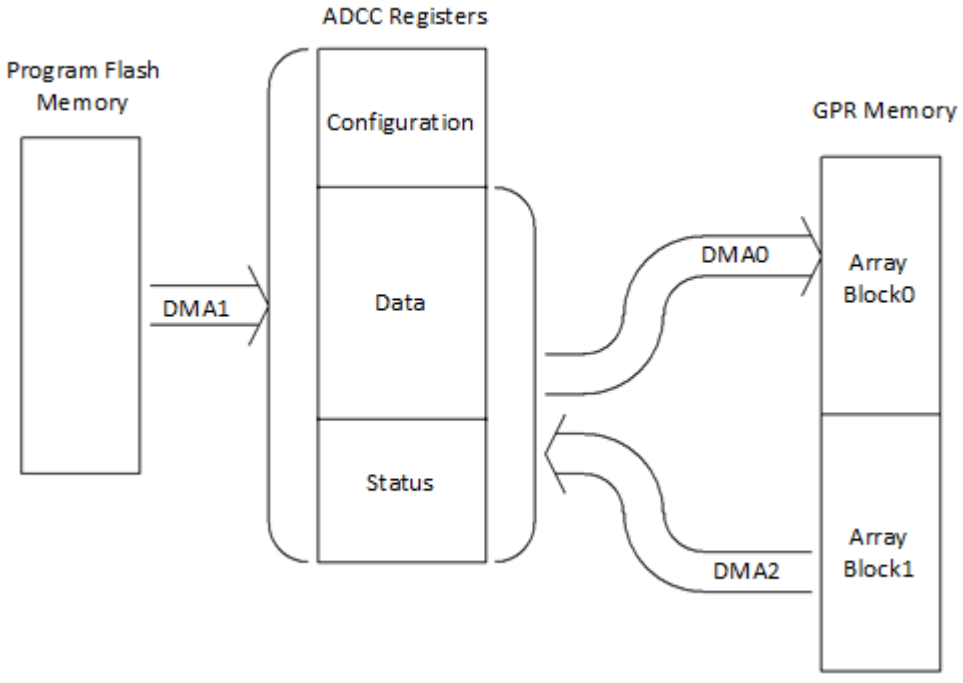
Arbitration priority is given to the DMA channel handling the register data read (Priority 0), followed by the DMA channel handling the Configuration register write for the next ADC conversion (Priority 1), and then the final DMA, which copies in the data register context for the next channel, is Priority 2. The system interrupt and main code follow as Priority 3 and 4.

The design is more robust in that the Configuration registers of the ADC are copied from Flash memory prior to each conversion. The previous design copied the configuration data from RAM, which could be at risk from corruption by runaway code.

This design also has lower GPR requirements as the context save only saves off the data registers for the computational section of the ADC. This accounts for 12 registers per channel, as opposed to the 34 registers per channel in the previous design. In fact, if the computation section is not used, there is no need for the context save and the only DMA required is the DMA that copies in the Configuration Data registers from program Flash memory. See Figure 4-1.

The attached project demonstrates this system using a PIC18F57Q43, a Curiosity Nano Adapter and three sensor Click boards™. The three sensor Click boards are a force-sensing click, an ambient light sensor click and a UV sensor click. Each sensor board has its own ADC configuration and the three channel DMA system cycles between the sensors, swapping in and out of the context for each. TMR0 drives the ADC conversion input and the ADC completion interrupt triggers the three DMA channels. The first DMA copies out the previous context, the second copies in the next context and the third copies in the ADC configuration for the new channel. All three DMA channels are triggered by the ADC done signal with the arbiter giving priority to the first DMA channel, second priority to the second DMA, third priority to the ADC configuration DMA, with fourth and fifth priority assigned to the interrupt and main line code.

Figure 4-1. Data Flow for Triple DMA Context Save System



Interrupts can be enabled, as mentioned in the previous design, including the same warnings about overrunning the ADC timing in the case of an interrupt stack up.

As in the previous design, the configuration of the DMA channels is straight forward with the ADC GODONE interrupt triggering all three interrupts. The system arbiter is again used to sequentially gate the DMA channels with the data context read first, the configuration write next and finally, the data context write last.

4.1 Use Cases

The three DMA system allows on the fly changes to the ADC configuration while the sampling system is operating. This allows the system to modify the computational aspect of the ADC sampler system as needed to respond to system changes. While this has value, it does open the system to potential problems if the configuration data is corrupted. The system also uses a very large number of GPR memory locations to store potentially redundant configuration information.

4.2 ADC Configuration

As in the previous design, start with the ADC:

1. Determine the number of data registers to be saved and the base address of the registers.
2. Setup an array of variables for the context save/restore:
   – The number of bytes of storage required will be: (number of channels) x (number of registers).
   – This will require that the variable space be defined in RAM banks that contain contiguous GPR memory.
3. Create a routine to clear the variable space.

4.3 DMA Channel Read Configuration

Once the ADC is configured, configure the ADC data read DMA channel:

1. Configure the source for GPR/SFR memory and the starting address of the ADC registers to be saved.
2. Configure the source size for the number of registers to be read.
3. Set the SSTP bit in the DMAxCON1 register to terminate the DMA when the source counter reloads.
4. Configure the source for post-increment.
5. Configure the destination with the starting address of the starting address of the array variable.
6. Configure the destination size for the size of the array variable.
7. Clear the DSTP bit in the DMAxCON1 register.
8. Configure the source for post-increment.
9. Set the DMA trigger source for the ADC conversion complete flag.
10. Configure the system arbitrator to give the read DMA the highest priority (0).

4.4 DMA Write Channel Configuration

Next, configure the DMA write channel:

1. Configure the source for program Flash memory containing the constant array and the starting address of the starting address of the constant array.
2. Configure the source size for the size of the array variable.
3. Clear the SSTP bit in the DMAxCON1 register.
4. Configure the source for post-increment.
5. Configure the destination for the starting address of the ADC Configuration registers.
6. Configure the destination size for the number of registers to be written.
7. Set the DSTP bit in the DMAxCON1 register to terminate the DMA when the destination counter reloads.
8. Configure the destination for post-increment.
9. Set the DMA trigger source for the ADC conversion complete flag.
10. Configure the system arbitrator to give the read DMA the highest priority (1).

4.5 Final System Configuration

For the final system configuration:

1. Set the system arbiter for interrupt at level (3) and main for level (4).
2. Set the PRLOCK in the system arbiter to prevent inadvertent modification of system priority. See Examples 4-1 and 4-2.
3. Manually (in software) trigger the first ADC configuration DMA write channel to preload the ADC for the first conversion.
4. Manually (in software) trigger the first ADC data DMA write channel to preload the ADC Data registers for the first conversion.
5. Start the peripheral supplying the ADC trigger.

Example 4-1. Priority Lock Sequence

```
INTCON0bits.GIE = 0;          // Disable Interrupts;
PRLOCK = 0x55;
PRLOCK = 0xAA;
PRLOCKbits.PRLOCKED = 1;      // Grant memory access to peripherals;
INTCON0bits.GIE = 1;          // Enable Interrupts;
```

Example 4-2. Priority Unlock Sequence

```
INTCON0bits.GIE = 0;          // Disable Interrupts;
PRLOCK = 0x55;
PRLOCK = 0xAA;
PRLOCKbits.PRLOCKED = 0;      // Allow changing priority settings;
INTCON0bits.GIE = 1;          // Enable Interrupts;
```

Note: On start-up, the data context write and the configuration write should be triggered manually to preload the Data and Configuration registers of the ADC for the first transfer. This will also move their pointers to the next channel configuration and data in preparation for the completion of the initial ADC conversion.

4.6 System Operation – Three Channels

Once started, the system should perform an ADC conversion, read the data context registers of the ADC and store the data in GPR memory, then load the ADC Configuration registers from Flash for the next ADC channel. It will then read the next set of data from GPR memory and write them to the ADC registers. At this point, the system is waiting until the next ADC trigger event. This sequence should continue until all the channels have been converted, then the DMA will roll over and start through the ADC input channels again, in a continuous loop, until the DMA is disabled, the ADC is disabled or the system is placed in Sleep.

To access the conversion data, the user simply reads the appropriate locations within the context array. If needed, the ADC conversion complete interrupt can be enabled and when the conversion is complete, the ISR can access the data in the context array.
Note: Because the ISR has a lower priority than the DMA in the system arbiter, the most recent conversion data will no longer be present in the ADRES registers and can only be accessed in the context array memory.

Threshold data can also be accessed in the context array memory in response to a threshold interrupt.

As in the previous example, the interrupt can be moved to the highest priority in the system arbiter. This will allow both ADC conversion complete and threshold interrupts to access the conversion data directly in the ADC registers. Then, when the interrupt completes, the DMA will automatically take control, and perform the read and write context function.

Note: The same timing issues mentioned in the previous design are still present, so using the system arbiter to set interrupts at a higher priority than DMA could still potentially delay the completion of the read and store of the ADC context due to a stack up of ISR calls. This could delay the context, switch such that the next trigger of the ADC could fire before the completion of the context switch, so care should be taken with this approach.

Project `Tripple_DMA.x` contains the configuration for this example. The reader is directed to the example and the ADC/DMA peripheral configuration information in the device data sheet for additional setup information.

5. Single-Channel DMA using ADC with Context Switching

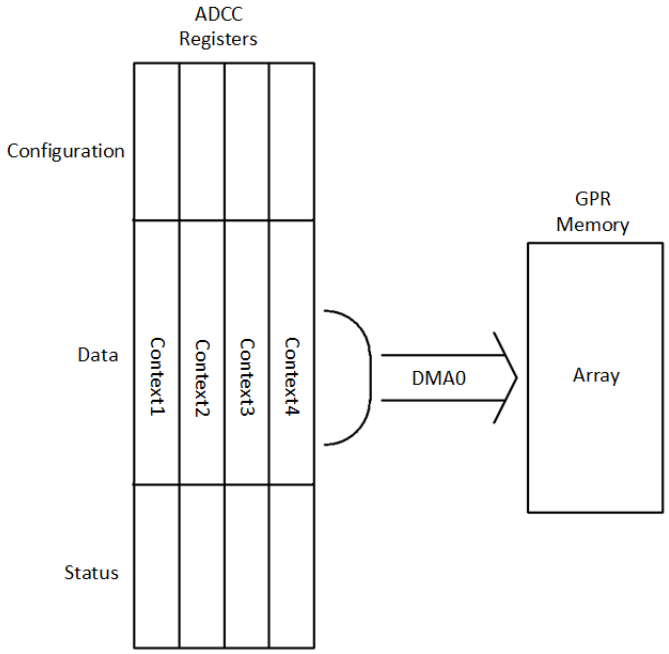
The final example combines a single DMA channel with the new ADC with Computation and Context Switching. Context switching is handled by the DMA context sequencer, with the DMA peripheral copying the resulting data out of the Result registers and into an external data memory. The DMA transfer is triggered by the ADC interrupt, signaling the end of the conversion or computation.

Arbitration priority (Priority 0) is given to the DMA channel handling the transfer of the ADC data from the Shadow register, while the system interrupt and main code follow as Priority 1 and 2.

This design has the lowest GPR requirements, as the context save handles the swapping of configuration, while GPR memory is only used to hold the conversion results. This accounts for between two and nine registers per channel, depending upon the amount of data saved. See Figure 5-1.

The attached project demonstrates this system using a PIC18F57Q84, a Curiosity HPC board and two sensor Click boards™. The two sensor Click boards are an ambient light sensor click and a UV sensor click. In addition, the internal temperature indicator will be sampled. Each sensor has its own ADC configuration and the ADC context sequencer cycles the ADC between the sensors, swapping in and out of the context for each. TMR0 drives the ADC conversion input and the ADC completion interrupt triggers the DMA channel. The DMA copies out the previous data and stores it into a GPR buffer for formatting and transmission to the host. The DMA channel is triggered by the ADC done signal with the arbiter giving priority to the DMA channel, second priority to the interrupt function and the third priority to the main line code.

Figure 5-1. Data Flow for Single ADC Context Save System



The read DMA has the highest priority so that when triggered, it will complete the register read in a burst. Once it completes, priority will fall back to the Interrupt Service Routines and main function code. This ensures the read is completed before the next ADC trigger.

5.1 Use Cases

The single DMA system with ADC Computation and Context Switching does allow on the fly changes to the ADC configuration while the sampling system is operating. This allows the system to modify the computational aspect of the ADC sampler system, as needed to respond to system changes. In addition, because the context switching is based on registers that are hidden from the system, it is also more robust than the three DMA system, though not as

secure as the two DMA system. The Single DMA system also uses the smallest number of GPR memory locations to store ADC results.

5.2 ADC Configuration

To configure the system, start with the ADC:

1. Determine the configuration for each ADC channel.
2. Create a routine to load the Configuration registers for each channel with the configuration values:
   – Include the ADC trigger.

Notes:

1. The ADCCON0 and ADCCLK registers are common for all context channels, so the values must be the same for all configurations.
2. The CTXSW bit in the ADCTX register must be cleared to write the starting context number in the ADCTX register.

5.3 DMA Channel Read Configuration

Once the ADC is configured, configure the read DMA channel:

1. Configure the source for GPR/SFR memory and the starting address of the ADC required data registers.
2. Configure the source size for the number of registers to be read.
3. Set the SSTP bit in the DMAxCON1 register to terminate the DMA when the source counter reloads.
4. Configure the source for post-increment.
5. Configure the destination with the starting address of the starting address of the GPR output array variable.
6. Configure the destination size for the size of the array variable.
7. Clear the DSTP bit in the DMAxCON1 register.
8. Configure the source for post-increment.
9. Set the DMA trigger source for the ADC conversion complete flag.
10. Configure the system arbitrator to give the read DMA the highest priority (0).

5.4 Final System Configuration

For the final system configuration:

1. Set the system arbiter for DMA1 at level (0).
2. Set the system arbiter for interrupt at level (1) and main for level (2).
3. Set the PRLOCK in the system arbiter to prevent inadvertent modification of system priority. See Examples 4-1 and 4-2.
4. Manually (in software) trigger the first DMA write channel to preload the ADC for the first conversion.
5. Start the peripheral supplying the ADC trigger.

5.5 System Operation – ADC with Context Sequencer

Once started, the system should perform an ADC conversion, read the results registers of the ADC and store the data in GPR memory. The ADC will then switch to the next context. At this point, the system is waiting until the next ADC trigger event. This sequence should continue until all of the channels have been converted. Then, the DMA and ADC sequencer will rollover and start through the ADC input channels in a continuous cycle until the DMA is disabled, the ADC is disabled, or the system is placed in Sleep.

To access the conversion data, the user reads the appropriate locations within the output array. If needed, the ADC conversion complete interrupt can be enabled, and when the conversion is complete, the ISR can access the data in the context array.

Note: Because the ISR has a lower priority than the DMA in the system arbiter, the most recent conversion data will no longer be present in the ADRES registers, and can only be accessed in the context array memory.

Threshold data can also be accessed in the context array memory in response to a threshold interrupt. Another option is to set the interrupt to the highest priority in the system arbiter. This will allow both ADC conversion complete and threshold interrupts to access the conversion data directly in the ADC registers. Then, when the interrupt completes, the DMA will automatically take control and perform the read and write context function.

Note: It is not possible to differentiate ADC interrupts from other system interrupts, so using the system arbiter to set interrupts at a higher priority than DMA could potentially delay the completion of the read and store of the ADC context due to a stack up of ISR calls. This could delay the context switch, such that the next trigger of the ADC could fire before the completion of the context switch, so care should be taken with this approach. Project `SINGLE_DMA.x` contains the configuration for this example. Refer to the example and the ADC/DMA peripheral configuration information in the device data sheet for additional setup information.

6. Conclusion

The purpose of this application note has been to demonstrate how to use the computational feature on multiple input channels. It accomplishes this by switching in and out the context of each channel's configuration and computation registers. The design methods included systems using two and three channels of DMA, as well as a system using a single DMA channel, plus a context switch and sequencer built into specialized versions of the ADC. Each design method has its own advantages and disadvantages.

The Dual DMA system is the simplest to configure and allows modification to each channel's configuration and computation on the fly. It does require a relatively large block of GPR memory for storage of both configuration and data context, and its configuration and data storage have no protection.

The Three-Channel DMA system is more robust due to the storage of configuration data in program memory. It uses three DMA channels, but with a smaller GPR memory block for just status and data storage.

The Single-Channel DMA uses an ADC with Computation and Context Switching, which allows on the fly modification, has a protected configuration and data memory, and uses minimal GPR memory to hold results.

See Table 6-1 for comparison of the three context switching methods.

Table 6-1. Context Switching Methods

Design Solutions	Channel Numbers	DMA Required	On the Fly Changes	Robustness	GPR Data Requirements
2-DMA System	Any	2	Yes	Medium	High
3-DMA System	Any	3	No	High	Low
ADC with Computation and Context Switching	≤ 4 channels	1	No	High	Zero

Depending upon the reliability needs of the system, one of the three design methods should be able to balance the requirements for reliability, mix of peripherals and GPR memory requirements.

Note: The specific mix of peripherals required for each of the three systems may not be available in every PIC18 device.

The Microchip Website

Microchip provides online support via our website at www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to www.microchip.com/pcn and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: www.microchip.com/support

Product Identification System

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.

Microchip Devices Code Protection Feature

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.

Legal Notice

Trademarks

ISBN: 978-1-5224-6186-9

Quality Management System

For information regarding Microchip's Quality Management Systems, please visit www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 Technical Support: www.microchip.com/support Web Address: www.microchip.com **Atlanta** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 **Austin, TX** Tel: 512-257-3370 **Boston** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 **Chicago** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 **Dallas** Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 **Detroit** Novi, MI Tel: 248-848-4000 **Houston, TX** Tel: 281-894-5983 **Indianapolis** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **Los Angeles** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 **Raleigh, NC** Tel: 919-844-7510 **New York, NY** Tel: 631-435-6000 **San Jose, CA** Tel: 408-735-9110 Tel: 408-436-4270 **Canada - Toronto** Tel: 905-695-1980 Fax: 905-695-2078	**Australia - Sydney** Tel: 61-2-9868-6733 **China - Beijing** Tel: 86-10-8569-7000 **China - Chengdu** Tel: 86-28-8665-5511 **China - Chongqing** Tel: 86-23-8980-9588 **China - Dongguan** Tel: 86-769-8702-9880 **China - Guangzhou** Tel: 86-20-8755-8029 **China - Hangzhou** Tel: 86-571-8792-8115 **China - Hong Kong SAR** Tel: 852-2943-5100 **China - Nanjing** Tel: 86-25-8473-2460 **China - Qingdao** Tel: 86-532-8502-7355 **China - Shanghai** Tel: 86-21-3326-8000 **China - Shenyang** Tel: 86-24-2334-2829 **China - Shenzhen** Tel: 86-755-8864-2200 **China - Suzhou** Tel: 86-186-6233-1526 **China - Wuhan** Tel: 86-27-5980-5300 **China - Xian** Tel: 86-29-8833-7252 **China - Xiamen** Tel: 86-592-2388138 **China - Zhuhai** Tel: 86-756-3210040	**India - Bangalore** Tel: 91-80-3090-4444 **India - New Delhi** Tel: 91-11-4160-8631 **India - Pune** Tel: 91-20-4121-0141 **Japan - Osaka** Tel: 81-6-6152-7160 **Japan - Tokyo** Tel: 81-3-6880- 3770 **Korea - Daegu** Tel: 82-53-744-4301 **Korea - Seoul** Tel: 82-2-554-7200 **Malaysia - Kuala Lumpur** Tel: 60-3-7651-7906 **Malaysia - Penang** Tel: 60-4-227-8870 **Philippines - Manila** Tel: 63-2-634-9065 **Singapore** Tel: 65-6334-8870 **Taiwan - Hsin Chu** Tel: 886-3-577-8366 **Taiwan - Kaohsiung** Tel: 886-7-213-7830 **Taiwan - Taipei** Tel: 886-2-2508-8600 **Thailand - Bangkok** Tel: 66-2-694-1351 **Vietnam - Ho Chi Minh** Tel: 84-28-5448-2100	**Austria - Wels** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **Denmark - Copenhagen** Tel: 45-4485-5910 Fax: 45-4485-2829 **Finland - Espoo** Tel: 358-9-4520-820 **France - Paris** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **Germany - Garching** Tel: 49-8931-9700 **Germany - Haan** Tel: 49-2129-3766400 **Germany - Heilbronn** Tel: 49-7131-72400 **Germany - Karlsruhe** Tel: 49-721-625370 **Germany - Munich** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **Germany - Rosenheim** Tel: 49-8031-354-560 **Israel - Ra'anana** Tel: 972-9-744-7705 **Italy - Milan** Tel: 39-0331-742611 Fax: 39-0331-466781 **Italy - Padova** Tel: 39-049-7625286 **Netherlands - Drunen** Tel: 31-416-690399 Fax: 31-416-690340 **Norway - Trondheim** Tel: 47-72884388 **Poland - Warsaw** Tel: 48-22-3325737 **Romania - Bucharest** Tel: 40-21-407-87-50 **Spain - Madrid** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **Sweden - Gothenberg** Tel: 46-31-704-60-40 **Sweden - Stockholm** Tel: 46-8-5090-4654 **UK - Wokingham** Tel: 44-118-921-5800 Fax: 44-118-921-5820