MCC porting: AVR-DA ADC triggered via EVSYS Code Example

Design Specification

Revision History

Revision Number	Revision Date	Changes	Revision Author
1	16-Jun-2020	Initial revision	Grig Barbulescu
2	17-Jun-2020	• Added first page • Added Revision History Chapter • Added comments to previous author's work	Marius Nicolae
3	18-Jun-2020	• Add header and footer • Reorganize chapters 3 and 4	Marius Nicolae
4	18-Jun-2020	• Changed title, minor edits	Grig Barbulescu

Table of Contents

1. Project Repository Name

The project repo name is: **avr128da48-cnano-adc-evsys-mcc**

2. Jira Project Details

Jira project task: https://jira.microchip.com/browse/MCU8APPS-32234

Jira Epic: https://jira.microchip.com/browse/MCU8APPS-26379

3. Project Description

This project describes the usage of the ADC peripheral triggered via Event System. Using the event system, the on-board user button (PC7) state change will trigger the ADC0 to start a conversion and read the analog signal on PD1 which comes from a 10 kohm potentiometer. With this setup, when the button on the Curiosity Nano board is pressed, a single conversion of the ADC will be triggered on Analog Input 1 (PD1). An interrupt is set to be activated when the ADC conversion cycle is over. After this interrupt the result of the conversion is saved and sent via serial communication (USART1) and the on-board user LED is toggled to visualize the and of conversion.

3.1. Project requirements

3.1.1. Parameters

- The value of the sharedAdcFlag parameter is updated inside the ADC Result Ready interrupt routine. Inside the while(1) infinite loop, this flag is polled.
- When the flag is set, the ADC conversion result is stored (atomic read) in the adcValue variable.
- The variable adcValue is printed via serial communication (both in raw and computed format) and the sharedAdcFlag is cleared.
- When printed, adcValue should be between 0 : 4095 (raw format) and 0 : 3.33 (computed format – VDD = 3.3V being the ADC voltage reference).

3.1.2. Functionality

- Pressing the user-button (PC7) will trigger a single conversion of the ADC via Event System.
- When the conversion is over, inside the interrupt routine, a flag is set.
- Inside the while(1) loop, when this flag is set, the on-board LED (PC6) is toggled and the conversion result is printed via USART1.
- The flag is cleared, and the program waits for another button press.
-

3.2. Hardware Used

- AVR128DA48 Curiosity Nano board (DM164151)
- 1x 10kohm potentiometer
- 3 jumper wires

3.3. Hardware Configuration

- CPU speed is 4 MHz.
- ADC0 is configured in single conversion mode (default mode), with AIN/1 (PD1) as analog input, 12-bit resolution and input clock is the system clock divided by 2. Event System and Result Ready interrupt are enabled.
- In the VREF peripheral, VDD is selected as reference for the ADC0 with the Always On feature selected.
- In the Event System, PC7 (user-button) is selected as channel 3 event generator, and ADC0 Start is selected as channel 3 event user.
- USART1 is used with 9600 baud rate, with only the TX enabled, no parity, 1 stop bit and 8-bit character size. Also, the printf support is enabled.

3.4. Software Used

- MPLAB X IDE v5.40
- XC8 v2.20
- MCC v3.95.0
- AVR-Dx_DFP 1.2.52
- 8-bit AVR MCUs Lib version 2.3.0
- MPLAB Data Visualizer v2.20

4. Software Specification

4.1. Description

The main software blocks are:

- MCC generated code for peripherals initialization (ADC0, VREF, USART1, EVSYS, PINMUX).

Beside the MCC drivers (for ADC, VREF, EVSYS, USART and port config), for this example is necessary to add the following code snippets:

1. In the "adc0.c" file, at the beginning, after the include section:

```
#include <stdbool.h>

extern volatile bool sharedAdcFlag;
extern volatile uint16_t sharedAdcValue;
```

2. In the "adc0.c" file, in the ADC0 Result Ready Interrupt:

```
/* This flag marks the end of an ADC conversion cycle */
sharedAdcFlag = true;
/* Reading the conversion result */
sharedAdcValue = ADC0_GetConversionResult();
```

3. In the "main.c" file, at the beginning, after the include section:

```
#include <stdbool.h>

volatile bool sharedAdcFlag = false;
volatile uint16_t sharedAdcValue = 0;
uint16_t adcValue = 0;

/* macro defined to toggle the on-board LED */
#define LED_Toggle() PORTC.OUTTGL = PIN6_bm
```

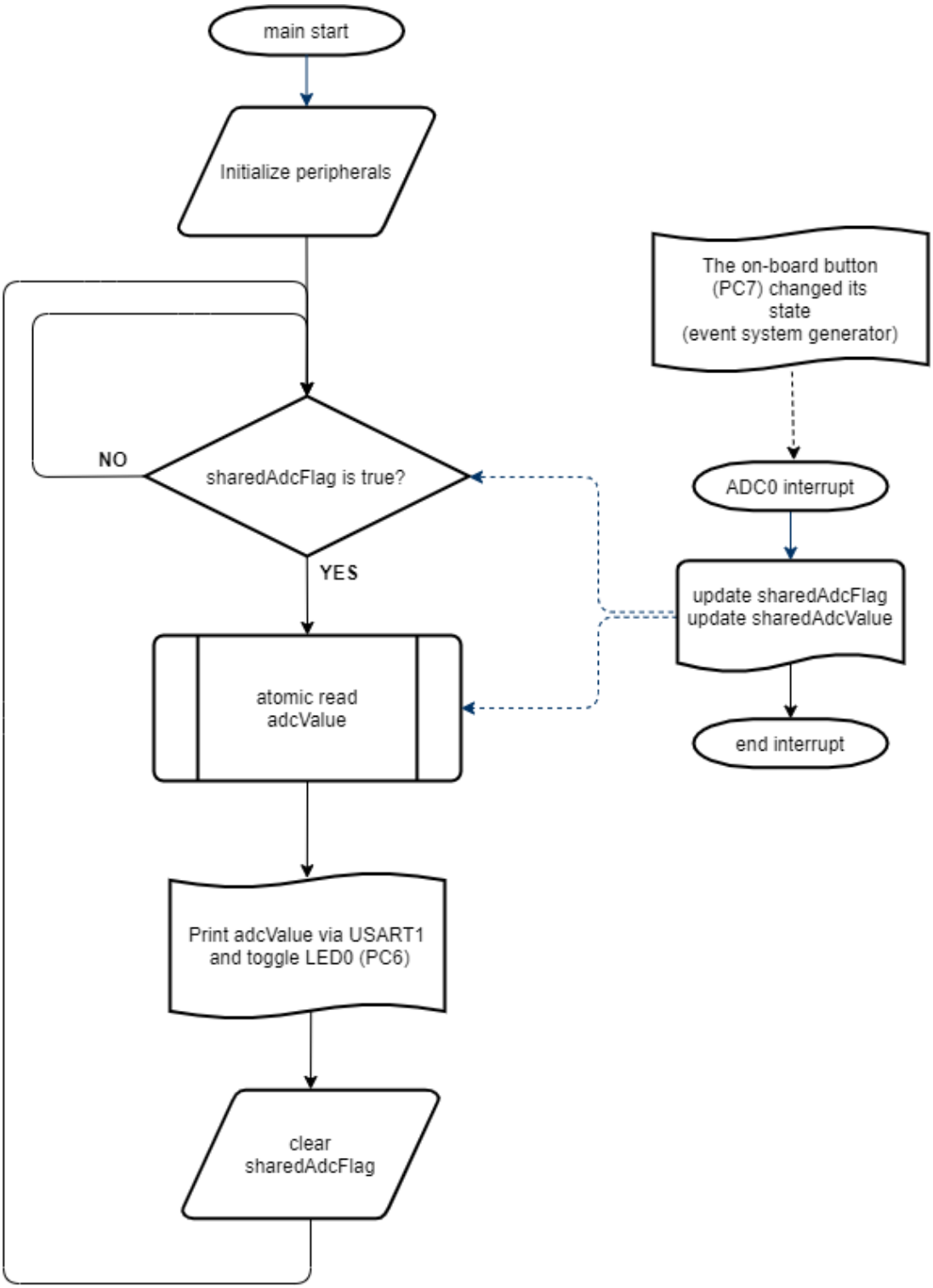
4. Also in the "main.c" file, add the following code snippet inside the while(1) loop.

```
/* if an ADC conversion has ended and the flag was set */
if(sharedAdcFlag == 1)
  {
      /* atomic read of the ADC conversion result */
      DISABLE_INTERRUPTS();
      adcValue = sharedAdcValue;
      ENABLE_INTERRUPTS();

      /* the LED on PC6 is toggled to visualize the end of the
conversion cycle */
      LED_Toggle();

      /* printing both the raw and computed results */
      printf("ADC Conversion Raw Result = %d\r\n", adcValue);
      printf("ADC Conversion Result [V] = %.2fV\r\n", adcValue
/ 4096.0 * 3.3);

      /* clearing the flag */
      sharedAdcFlag = false;
  }
```

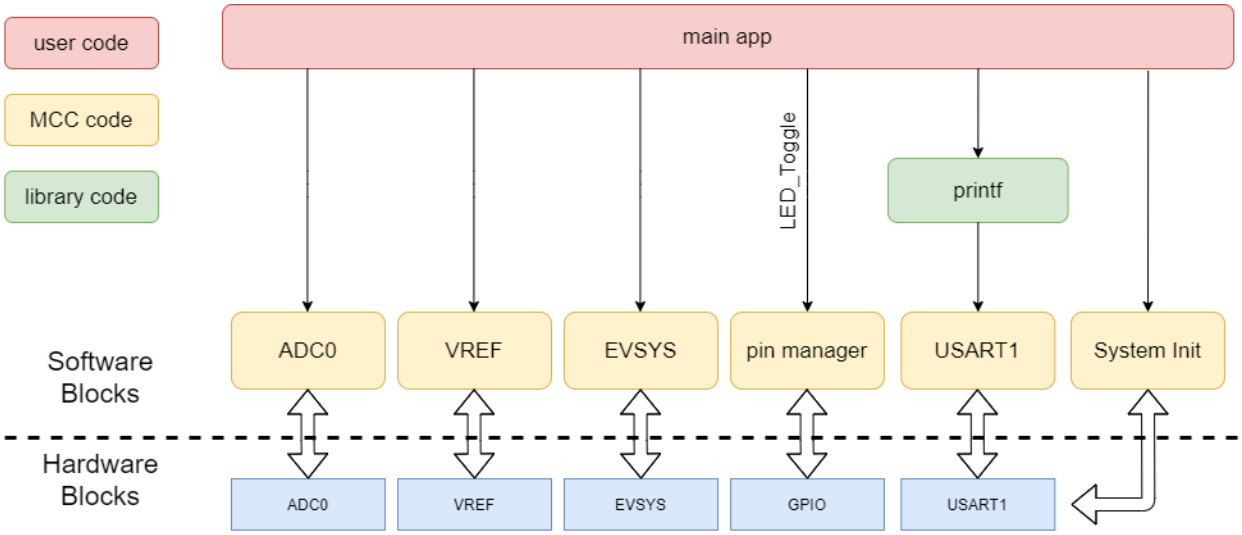
- The 1st and 3rd code snippets are variable and macro declaration.
- The 2nd code snippet represents the line needed to set the sharedAdcFlag and read the conversion result inside the ADC Result Ready interrupt.
- The 4th code listing executes an atomic read of the conversion result, toggles the on-board LED and prints the conversion result when the sharedAdcFlag is set.

4.2. Software diagrams

4.2.1. Main application flowchart



4.3. Software modules architecture



4.4. Detailed specification

This section of the document covers the APIs and function used in this SW project.

4.4.1. Pre-existing APIs Used

4.4.1.1. MCC-generated APIs used by this project

- `ADC0_GetConversionResult()`
- `ENABLE_INTERRUPTS()`
- `DISABLE_INTERRUPTS()`

4.4.1.2. Compiler and DFP related APIs used by this project

- `printf()`

4.4.2. Custom Functions

If you consider it would be helpful to further break-down the following functions to categories, feel free to create multiple sub-chapters (e.g. temp module, display module etc).

4.4.2.1. `main`

Name	`main`
Prototype	`void main(void)`
Summary	Main functions
Description	Initializes the system. Inside the while(1) loop, when the sharedAdcFlag is set in the interrupt routine, the conversion result is read and printed, and on-board LED is toggled.
Preconditions	none
Parameters	none
Return	none
Example use	`Called by default`