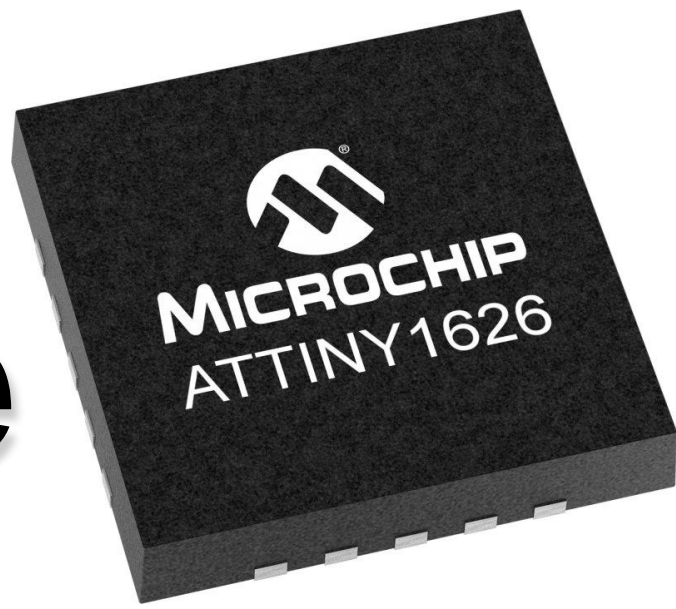


**How to
Use the
ADC**



**On the MCU
ATtiny1626
with Code**

Table of Content

1. Overview
2. MCU ATtiny1626 Block Diagram
3. ADC Block Diagram
4. MCU ATtiny1626 (20-Pin VQFN) Pinout
5. Project Schematic
6. Atmel Studio 7 IDP
7. Create a New Project using Atmel Start Project
8. Use Device Filter to Search for MCU
9. Select the Desired MCU
10. Add the Drivers (ADC, etc)
11. Create New Project
12. Check Project Software Components
13. Setup the ADC Peripheral

Table of Content (continue)

14. Select ADC Signals
15. Setup the VREF Peripheral
16. Setup the Clock Settings
17. Use the Pinmux Configurator to Mux Pins
18. Generate the Project Files
19. Name the Project and Save it
20. Use the Solution Explorer to Browse the Project
21. Entry Point to the Program Code
22. ADC Pins Initialization Code
23. ADC Peripheral Initialization Code
24. Clock Initialization Code
25. Function to Measure Voltage with ADC Code
26. Conclusion
27. References

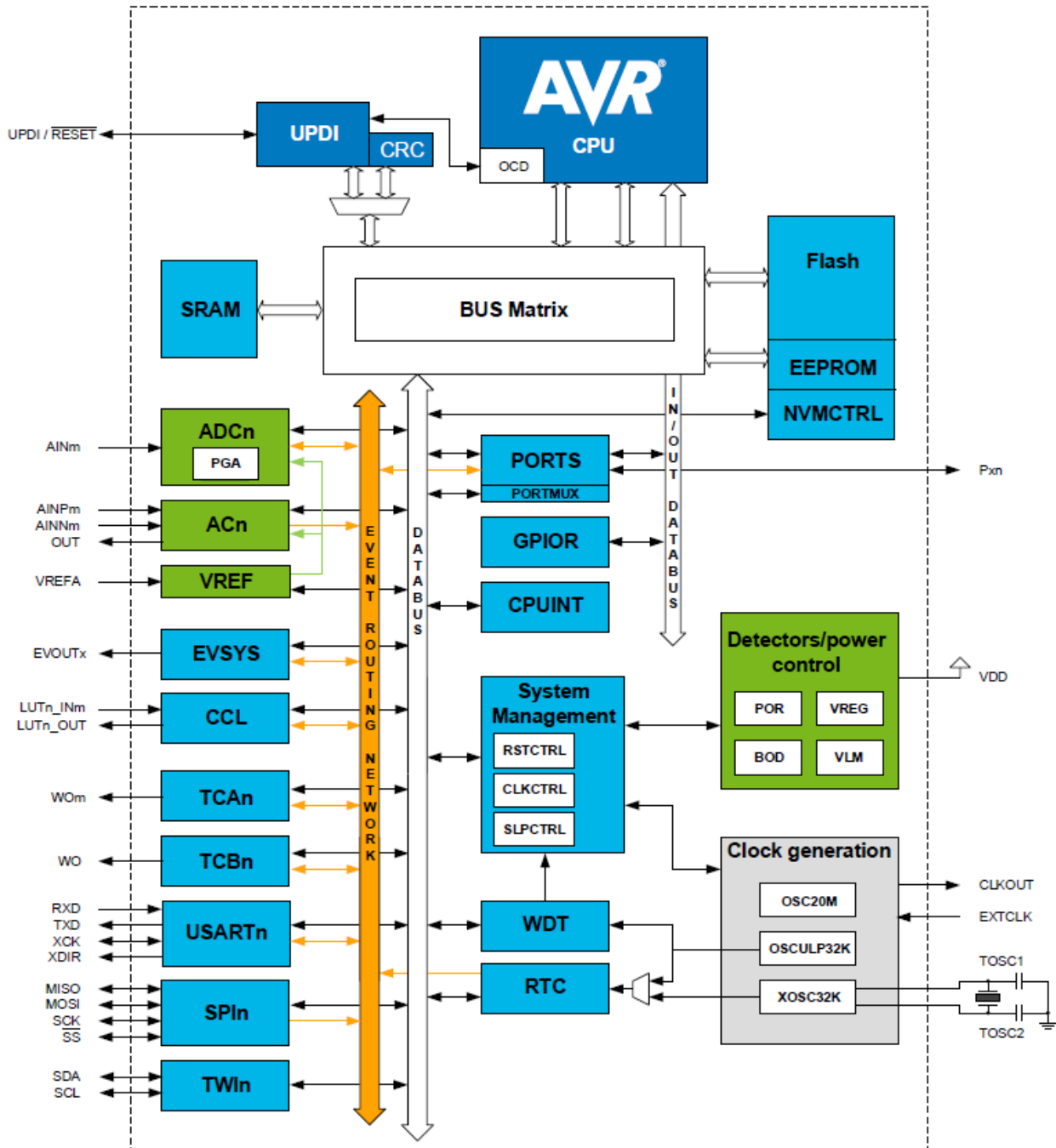
Overview

This tutorial will guide you through the process of create a project using [Atmel Studio 7 IDP](#).

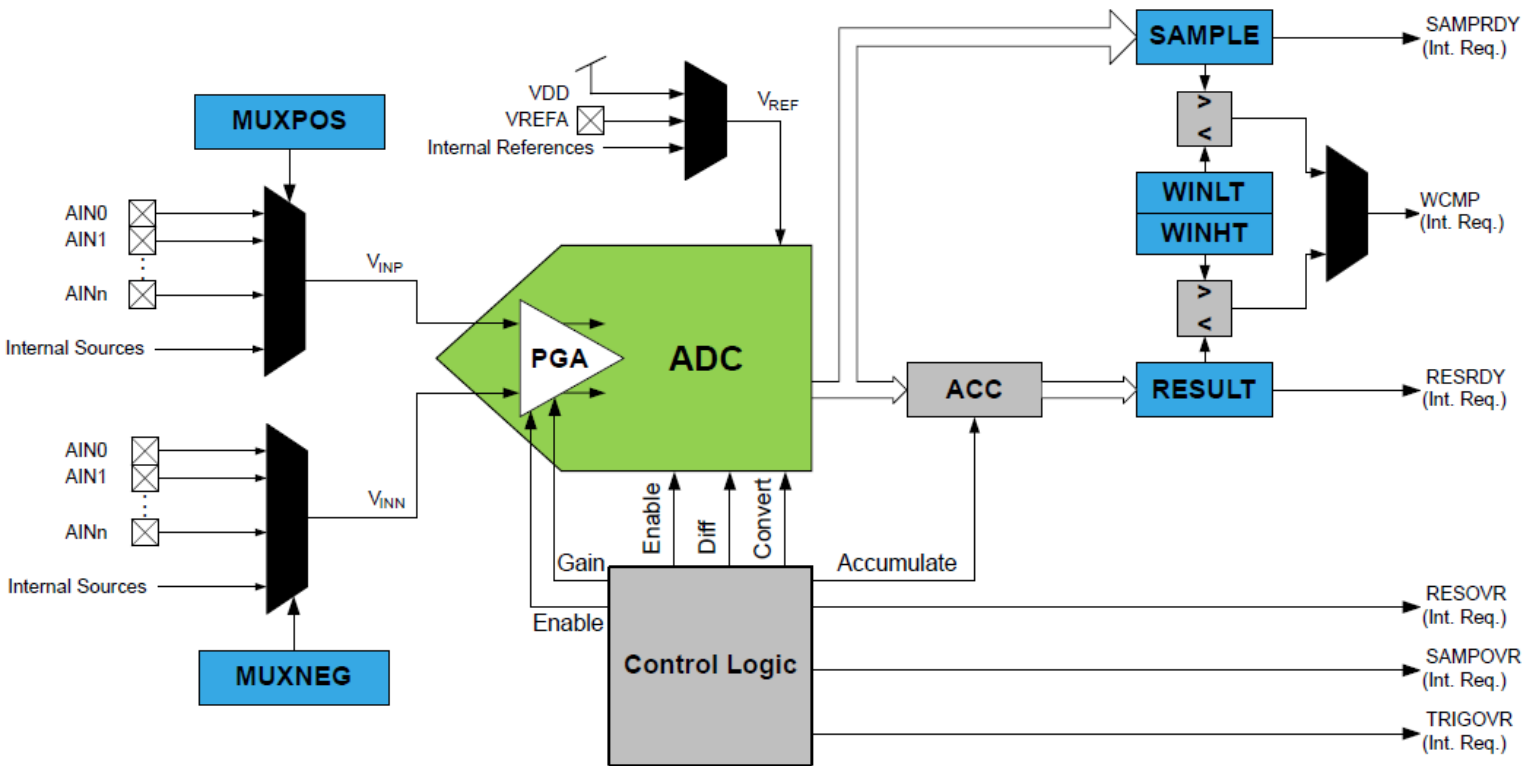
Atmel® Studio 7 is the integrated development platform (IDP) for developing and debugging SMART ARM®-based and AVR® microcontroller (MCU) applications. Studio 7 supports all AVR and SMART MCUs. The Atmel Studio 7 IDP gives you a seamless and easy-to-use environment to write, build and debug your applications written in C/C++ or assembly code. It also connects seamlessly to Atmel debuggers and development kits.

The project is a simple, but intuitive example designed to shows how to configure and use the ADC of the [ATtiny1626](#) MCU to measure the voltage from three different channels.

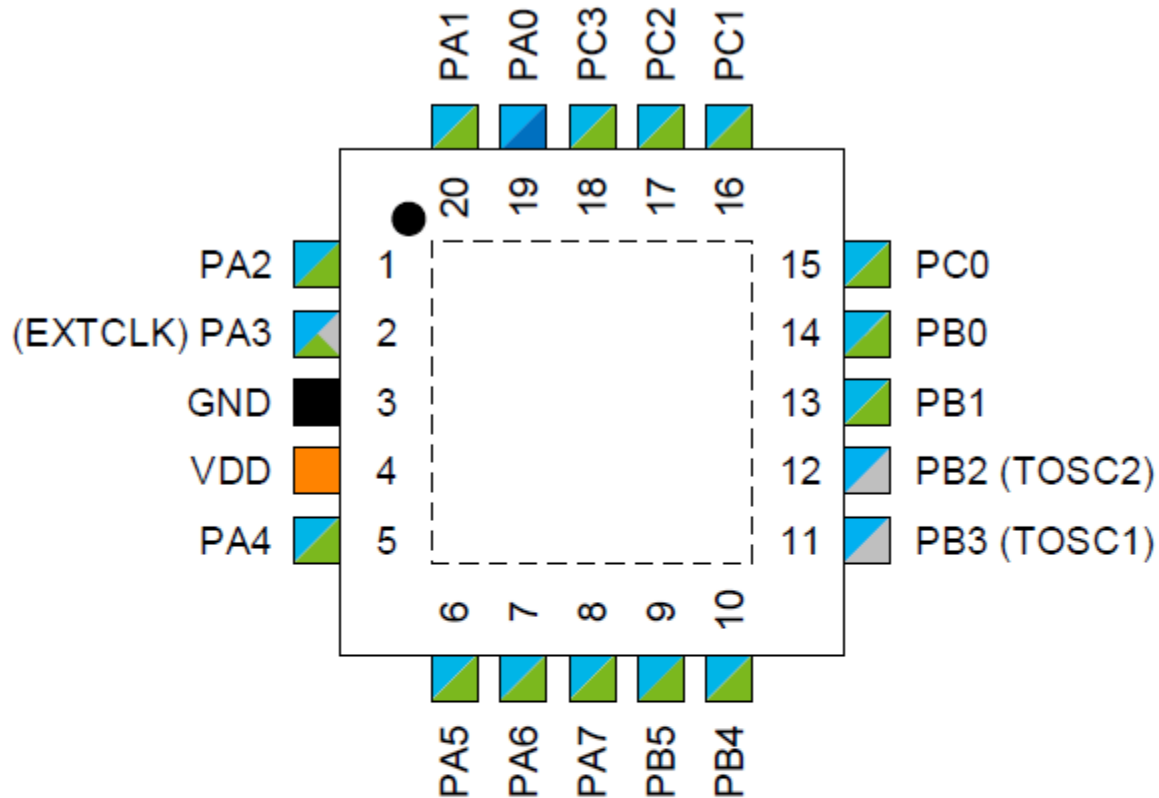
MCU ATtiny1626 Block Diagram




ADC Block Diagram





MCU ATtiny1626 (20-Pin VQFN) Pinout




Power


 Power Supply

 Ground

 Pin on VDD Power Domain

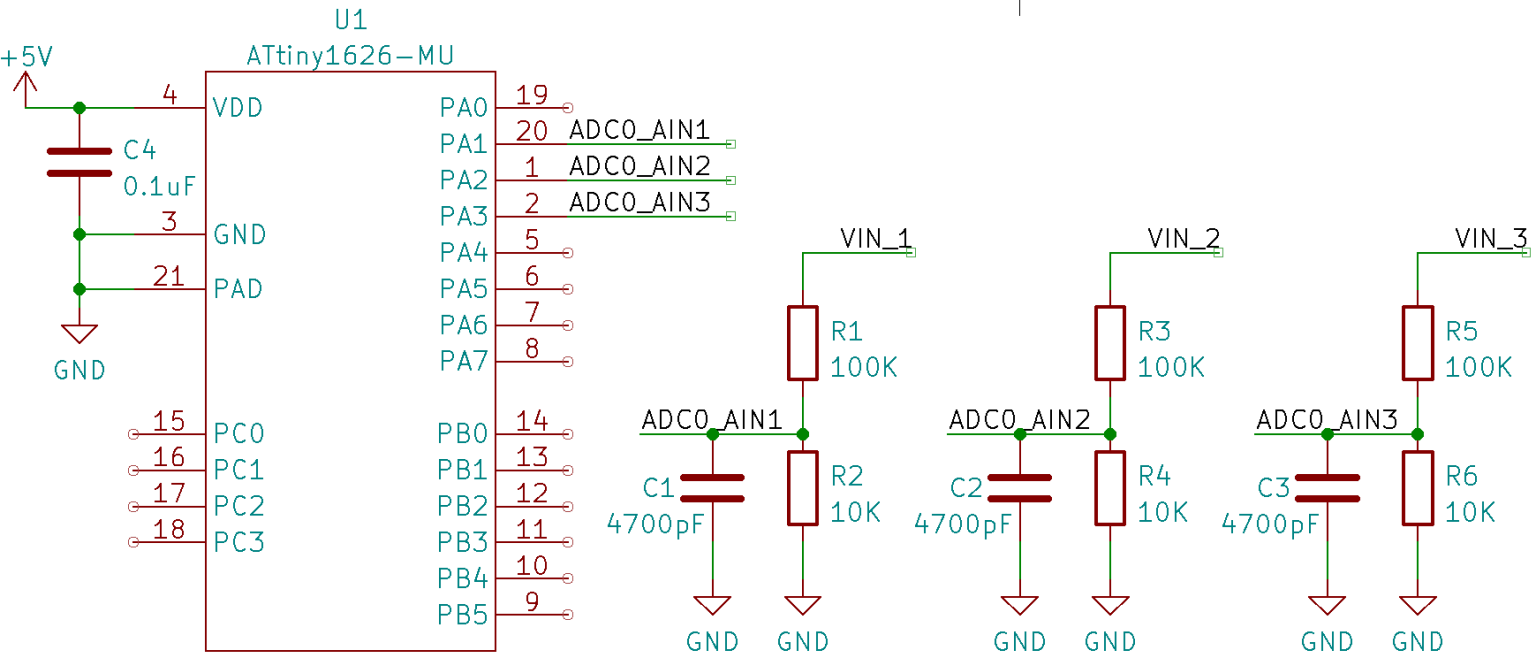
Functionality

 Programming/Debug

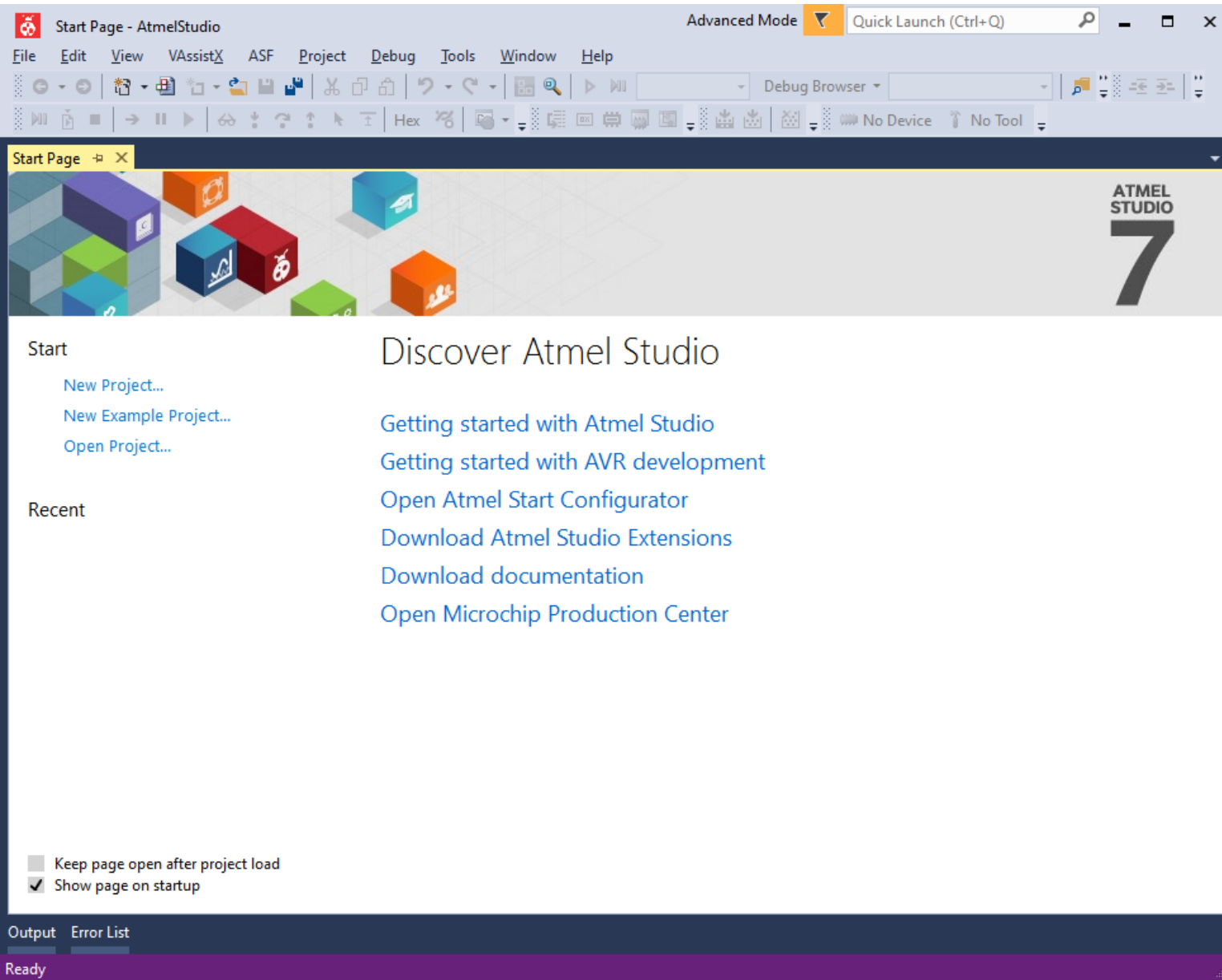
 Clock/Crystal

 Analog Function

Project Schematic

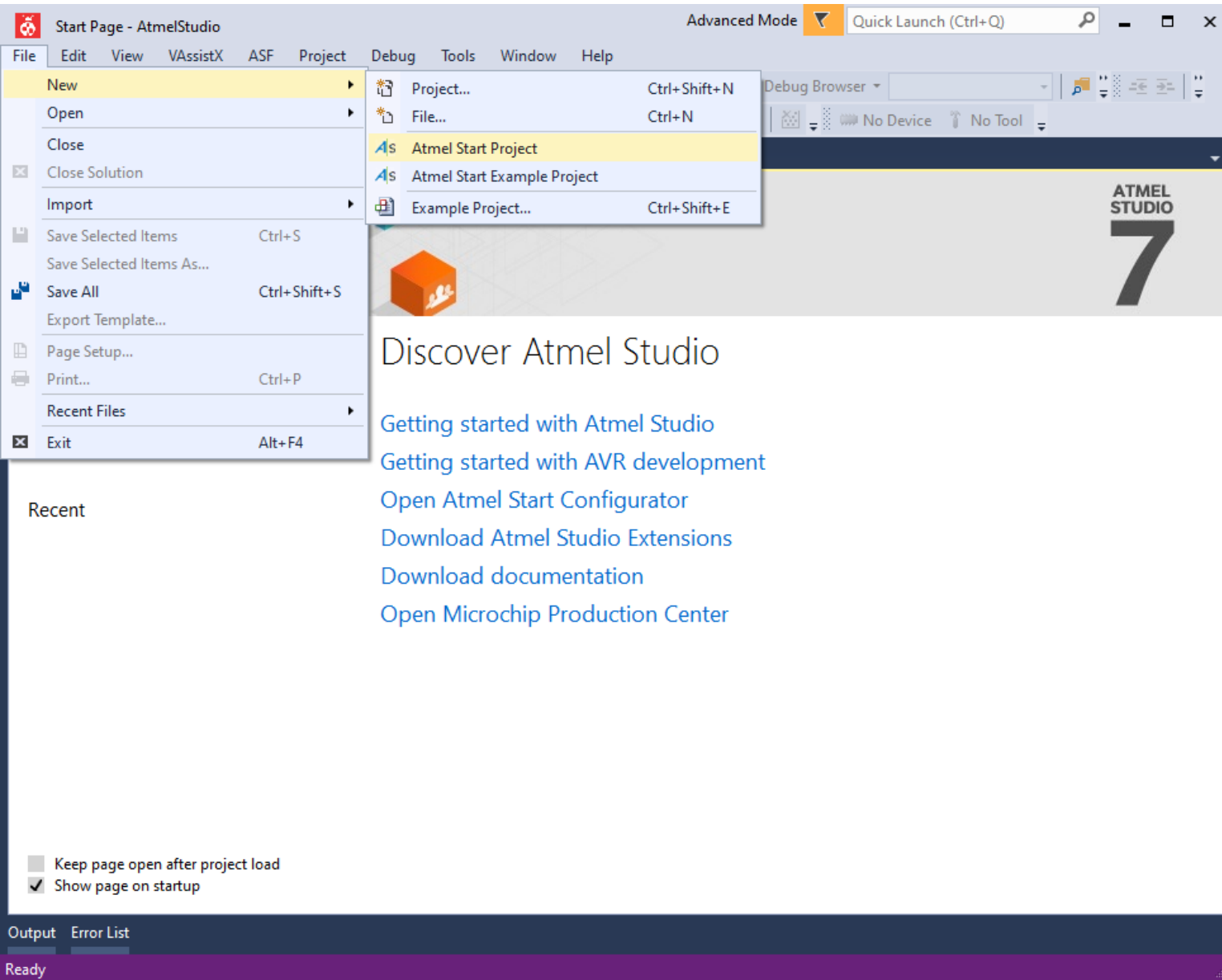


Atmel Studio 7 IDP



Atmel® Studio 7 is the integrated development platform (IDP) for developing and debugging microcontroller (MCU) applications.

Create a New Project using Atmel Start Project



Use Device Filter to Search for MCU

Atmel | START - AtmelStudio

Advanced Mode Quick Launch (Ctrl+Q)

File Edit View VAssistX ASF Project Debug Tools Window Help

Atmel | START Start Page

Atmel | START

What's New | Help

CREATE NEW PROJECT

Select device or board before creating a new project. You can filter devices and boards by what software you need and also with hardware requirements such as memory sizes.

FILTERS

HARDWARE

SEARCH FOR SOFTWARE

Find software...

MIDDLEWARE

Atmel Data Protocol 0

+ Audio and Voice Click

DRIVERS

AC 0

ADC 0

Analog Glue Function 0

CAN 0

RESULTS

Device filter

Filter on device... ☒ Show all ☐ Show only boards ☐ Show only dev

Name	Architecture	Package	Pins	Flash	SRAM
ATmega128L-8MU	AVR	QFN64	64	128 KB	4 KB
ATmega128L-8MN	AVR	QFN64	64	128 KB	4 KB
ATmega128L-8AU	AVR	TQFP64	64	128 KB	4 KB
ATmega128L-8AN	AVR	TQFP64	64	128 KB	4 KB
ATmega128-16MU	AVR	QFN64	64	128 KB	4 KB
ATmega128-16MN	AVR	QFN64	64	128 KB	4 KB
ATmega128-16AU	AVR	TQFP64	64	128 KB	4 KB
ATmega128-16AN	AVR	TQFP64	64	128 KB	4 KB
ATmega1280V-8CU	AVR	CBGA100	100	128 KB	8 KB
ATmega1280V-8AU	AVR	TQFP100	100	128 KB	8 KB

1049 of 1049 boards and devices

CREATE NEW PROJECT

Output Error List

Ready

Use the device filter to search for the desired MCU.

Select the Desired MCU

Atmel | START - AtmelStudio

Advanced Mode Quick Launch (Ctrl+Q)

File Edit View VAssistX ASF Project Debug Tools Window Help

Atmel | START Start Page

Atmel | START

What's New | Help

CREATE NEW PROJECT

Select device or board before creating a new project. You can filter devices and boards by what software you need and also with hardware requirements such as memory sizes.

FILTERS

HARDWARE

SEARCH FOR SOFTWARE

Find software...

MIDDLEWARE

- + Audio and Voice Click
- Buck 2 Click

DRIVERS

- AC
- ADC
- CRC
- Digital Glue Logic

RESULTS

attiny1626 ☒ Show all ☐ Show only boards ☐ Show only dev

Name	Architecture	Package	Pins	Flash	SRAM
ATtiny1626	AVR	VQFN20	20	16 KB	2 KB

1 of 1049 boards and devices

CREATE NEW PROJECT

Output Error List

Ready

ATtiny1626 MCU has been selected

Add the Drivers (ADC, etc)

Atmel | START - AtmelStudio

Advanced Mode Quick Launch (Ctrl+Q)

File Edit View VAssistX ASF Project Debug Tools Window Help

Atmel | START Start Page

Atmel | START

What's New | Help

CREATE NEW PROJECT

Select device or board before creating a new project. You can filter devices and boards by what software you need and also with hardware requirements such as memory sizes.

FILTERS

HARDWARE

SEARCH FOR SOFTWARE

Find software...

MIDDLEWARE

DRIVERS

- AC 0
- ADC 1
- CRC 0
- Digital Glue Logic 0
- Event System 0
- External DAC 0

RESULTS

attiny1626 Show all Show only boards Show only dev

Name	Architecture	Package	Pins	Flash	SRAM
ATtiny1626	AVR	VQFN20	20	16 KB	2 KB

1 of 1049 boards and devices

CREATE NEW PROJECT

Output Error List

Ready

Once drivers have been added, click on **CREATE NEW PROJECT** to create the project.

Create New Project

Atmel | START - AtmelStudio

Advanced Mode Quick Launch (Ctrl+Q)

File Edit View VAssistX ASF Project Debug Tools Window Help

Atmel | START Start Page

CREATE NEW PROJECT

Select device or board before creating a new project. You can filter devices and boards by what software you need and also with hardware requirements such as memory sizes.

FILTERS

HARDWARE

SEARCH FOR SOFTWARE

Find software...

MIDDLEWARE

DRIVERS

- AC 0
- ADC 1
- CRC 0
- Digital Glue Logic 0
- Event System 0
- External DAC 0

RESULTS

ADC (1) x VREF (1) x Timer (1) x

attiny1626 ☒ Show all ☐ Show only boards ☐ Show only dev

Name	Architecture	Package	Pins	Flash	SRAM
ATtiny1626	AVR	VQFN20	20	16 KB	2 KB

1 of 1049 boards and devices

Create project button

CREATE NEW PROJECT

Output Error List

Ready

Once drivers have been added for desired MCU, click on button **CREATE NEW PROJECT** to create the project.

Check Project Software Components

Atmel | START - AtmelStudio

Advanced Mode Quick Launch (Ctrl+Q)

File Edit View VAssistX ASF Project Debug Tools Window Help

Atmel | START Start Page

Atmel | START ATtiny1626

What's New | Help

MY SOFTWARE COMPONENTS

- Application
- Middleware
- Driver
- System driver

+ Add software component

Show system drivers Show hardware

MY PROJECT

ADC_0 VREF_0 TIMER_0

CLKCTRL CPUINT BOD SLPCTRL

MY PROJECT

GENERAL

Rename component

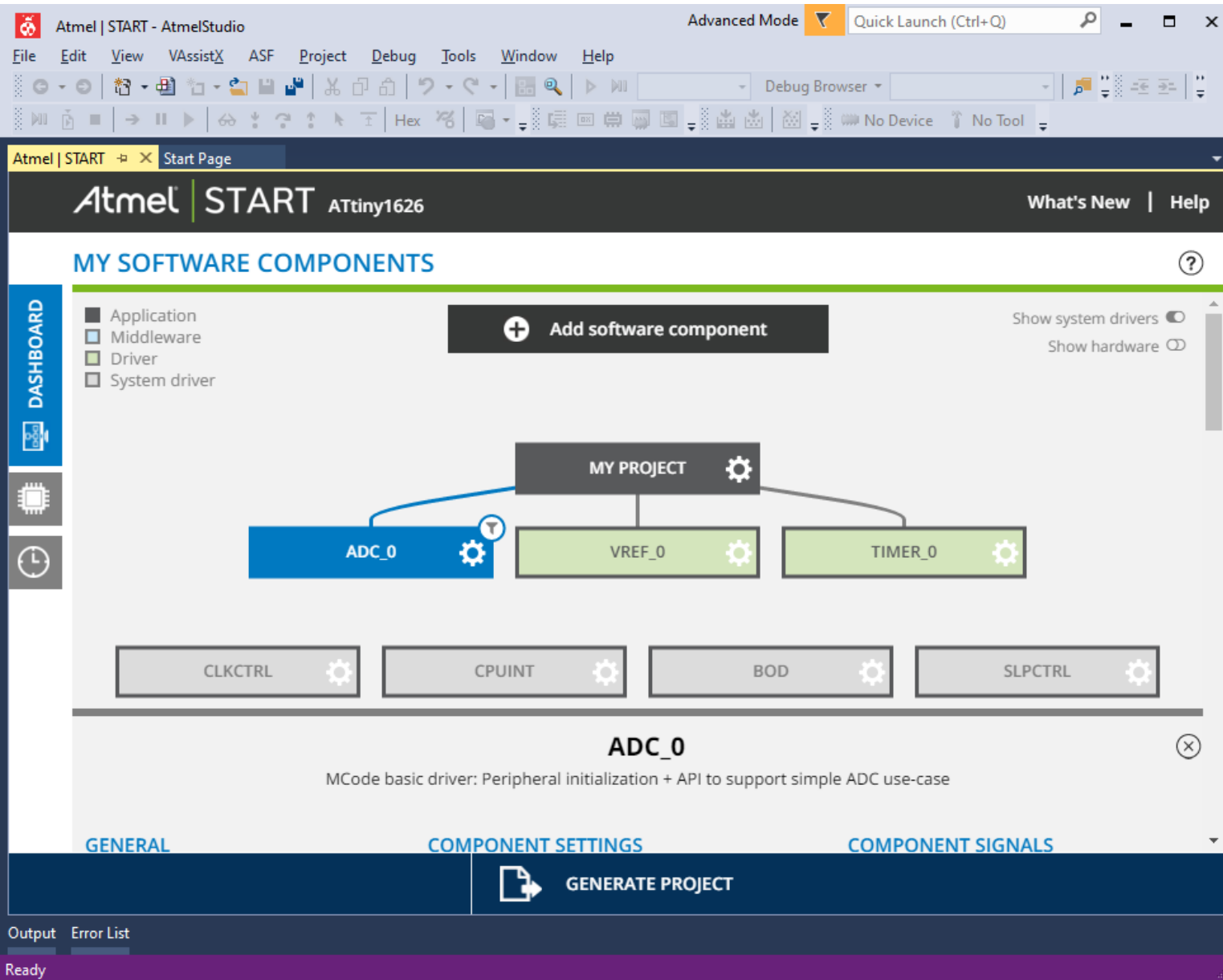
GENERATE PROJECT

Output Error List

Ready

```
graph TD; MP[MY PROJECT] --- ADC_0[ADC_0]; MP --- VREF_0[VREF_0]; MP --- TIMER_0[TIMER_0]; MP --- CLKCTRL[CLKCTRL]; MP --- CPUINT[CPUINT]; MP --- BOD[BOD]; MP --- SLPCTRL[SLPCTRL];
```

Setup the ADC Peripheral



The screenshot displays the Atmel Studio IDE interface. The top menu bar includes File, Edit, View, VAssistX, ASF, Project, Debug, Tools, Window, and Help. The toolbar contains various icons for file operations, editing, and debugging. The main workspace shows the 'MY PROJECT' component tree. The 'ADC_0' component is selected and highlighted in blue. Below the component tree, the 'ADC_0' configuration options are visible, including 'MCode basic driver: Peripheral initialization + API to support simple ADC use-case'. The bottom status bar shows 'Output', 'Error List', and 'Ready'.

Atmel | START - AtmelStudio

Advanced Mode Quick Launch (Ctrl+Q)

File Edit View VAssistX ASF Project Debug Tools Window Help

Atmel | START Start Page

Atmel | START ATtiny1626 What's New | Help

MY SOFTWARE COMPONENTS

Application
Middleware
Driver
System driver

Add software component

MY PROJECT

ADC_0 VREF_0 TIMER_0

CLKCTRL CPUINT BOD SLPCTRL

ADC_0

MCode basic driver: Peripheral initialization + API to support simple ADC use-case

GENERAL COMPONENT SETTINGS COMPONENT SIGNALS

GENERATE PROJECT

Output Error List

Ready

Click on **ADC_0** and scroll down to show the ADC configuration options.

Select ADC Signals

The screenshot shows the Atmel Studio interface with the 'ADC_0' component selected. The 'COMPONENT SIGNALS' section is expanded, showing a list of ADC input channels (AIN/1 through AIN/12) with checkboxes for selection. The 'COMPONENT SETTINGS' section shows the 'Driver' set to 'Drivers:ADC:Basic' and the 'Clock' set to 'Main Clock (CLK_MAIN) (3.33 MHz)'. The 'GENERAL' section shows options for 'User guide', 'Rename component', and 'Remove component'. The 'DASHBOARD' sidebar is visible on the left, and the 'GENERATE PROJECT' button is at the bottom.

Atmel | START - AtmelStudio

Advanced Mode Quick Launch (Ctrl+Q)

File Edit View VAssistX ASF Project Debug Tools Window Help

Atmel | START Start Page

Atmel | START ATtiny1626 What's New | Help

MY SOFTWARE COMPONENTS

ADC_0

MCode basic driver: Peripheral initialization + API to support simple ADC use-case

GENERAL

- User guide
- Rename component
- Remove component

COMPONENT SETTINGS

Driver: Drivers:ADC:Basic

CLOCKS

ADC: Main Clock (CLK_MAIN) (3.33 MHz)

COMPONENT SIGNALS

AIN/1:	<input checked="" type="checkbox"/>	PA1
AIN/2:	<input checked="" type="checkbox"/>	PA2
AIN/3:	<input checked="" type="checkbox"/>	PA3
AIN/4:	<input type="checkbox"/>	PA4
AIN/5:	<input type="checkbox"/>	PA5
AIN/6:	<input type="checkbox"/>	PA6
AIN/7:	<input type="checkbox"/>	PA7
AIN/8:	<input type="checkbox"/>	PB5
AIN/9:	<input type="checkbox"/>	PB4
AIN/12:	<input type="checkbox"/>	PC0

GENERATE PROJECT

Output Error List

Ready

Select the ADC signals as per the schematic. Also select the driver and the clock.

Setup the VREF Peripheral

The screenshot displays the Atmel Studio environment. The top menu bar includes File, Edit, View, VAssistX, ASF, Project, Debug, Tools, Window, and Help. The toolbar contains various icons for file operations, development, and debugging. The main window shows the 'Atmel | START' project for 'ATtiny1626'. The 'MY SOFTWARE COMPONENTS' section is active, showing a tree structure where 'VREF_0' is selected. A legend on the left identifies component types: Application (dark grey), Middleware (light blue), Driver (green), and System driver (grey). Below the tree, a row of component boxes includes CLKCTRL, CPUINT, BOD, and SLPCTRL. The 'VREF_0' component is expanded, showing its 'VREF driver' configuration options. The bottom of the interface features tabs for 'GENERAL', 'COMPONENT SETTINGS', and 'COMPONENT SIGNALS', along with a 'GENERATE PROJECT' button. The status bar at the bottom indicates 'Ready'.

Atmel | START - AtmelStudio

Advanced Mode Quick Launch (Ctrl+Q)

File Edit View VAssistX ASF Project Debug Tools Window Help

Atmel | START Start Page

Atmel | START ATtiny1626 What's New | Help

MY SOFTWARE COMPONENTS

Application
Middleware
Driver
System driver

Add software component

MY PROJECT

ADC_0 VREF_0 TIMER_0

CLKCTRL CPUINT BOD SLPCTRL

VREF_0
VREF driver

GENERAL COMPONENT SETTINGS COMPONENT SIGNALS

GENERATE PROJECT

Output Error List

Ready

Click on VREF_0 and scroll down to show the VREF driver configuration options.

Setup the VREF Peripheral (continue)

The screenshot shows the Atmel Studio interface for configuring the VREF peripheral. The top bar includes the Atmel logo, 'START - AtmelStudio', 'Advanced Mode', and a 'Quick Launch (Ctrl+Q)' search bar. The menu bar contains File, Edit, View, VAssistX, ASF, Project, Debug, Tools, Window, and Help. The toolbar has various icons for file operations, editing, and debugging. The main window title is 'Atmel | START - Start Page'. The header area displays 'Atmel | START ATtiny1626' and 'What's New | Help'. The main content area is titled 'MY SOFTWARE COMPONENTS' and features a sidebar with 'DASHBOARD' and icons for a circuit board and a clock. The central panel shows the 'VREF_0' component, a 'VREF driver'. It has three tabs: 'GENERAL', 'COMPONENT SETTINGS', and 'COMPONENT SIGNALS'. Under 'GENERAL', there are buttons for 'User guide', 'Rename component', and 'Remove component'. Under 'COMPONENT SETTINGS', the 'Driver:' dropdown is set to 'Drivers:VREF:Init'. Below this, the section 'DRIVERS:VREF:INIT (VREF) CONFIGURATION ON VREF' is shown, followed by 'VOLTAGE REFERENCE FOR ADC AND AC/DAC'. This section contains four configuration items: 'ADCCOREFEN: ADC0 reference enable:' with an unchecked checkbox, 'ACCCOREFEN: AC0 DACREF reference enable:' with an unchecked checkbox, 'ACCCOREFSEL: AC0 reference select:' with a dropdown menu set to 'Voltage reference at 2.5V', and 'NVMREFEN: NVM reference enable:' with an unchecked checkbox. At the bottom right, there is a 'GENERATE PROJECT' button with a document icon. The bottom status bar shows 'Output Error List' and 'Ready'.

Atmel | START - AtmelStudio Advanced Mode Quick Launch (Ctrl+Q)

File Edit View VAssistX ASF Project Debug Tools Window Help

Atmel | START - Start Page

Atmel | START ATtiny1626 What's New | Help

MY SOFTWARE COMPONENTS

VREF_0

VREF driver

GENERAL

- User guide
- Rename component
- Remove component

COMPONENT SETTINGS

Driver: Drivers:VREF:Init

COMPONENT SIGNALS

DRIVERS:VREF:INIT (VREF) CONFIGURATION ON VREF

VOLTAGE REFERENCE FOR ADC AND AC/DAC

ADCCOREFEN: ADC0 reference enable: ☐

ACCCOREFEN: AC0 DACREF reference enable: ☐

ACCCOREFSEL: AC0 reference select: Voltage reference at 2.5V

NVMREFEN: NVM reference enable: ☐

GENERATE PROJECT

Output Error List

Ready

Setup the Clock Settings

The screenshot shows the Atmel Studio IDE interface. The top menu bar includes File, Edit, View, VAssistX, ASF, Project, Build, Debug, Tools, Window, and Help. The toolbar contains various icons for file operations, debugging, and development. The main window displays the 'Atmel | START ATtiny1626' project. The 'MY SOFTWARE COMPONENTS' section shows a tree structure with 'MY PROJECT' at the top, branching into 'VREF_0', 'ADC_0', and 'TIMER_0'. Below this, a row of components includes 'BOD', 'CLKCTRL' (highlighted in blue), 'CPUINT', and 'SLPCTRL'. The 'CLKCTRL' component is selected, and its configuration window is open, showing the 'Clock Configuration' options. The bottom status bar indicates 'Ready'.

ADC_Example - AtmelStudio Advanced Mode Quick Launch (Ctrl+Q)

File Edit View VAssistX ASF Project Build Debug Tools Window Help

Debug Debug Browser

Hex ATtiny1626 None on

adc_basic.c clkctrl.c driver_init.c main.c Atmel | START iotn1626.h

Atmel | START ATtiny1626 What's New | Help

MY SOFTWARE COMPONENTS

Application
Middleware
Driver
System driver

Add software component

MY PROJECT

VREF_0 ADC_0 TIMER_0

BOD CLKCTRL CPUINT SLPCTRL

CLKCTRL
Clock Configuration

GENERAL COMPONENT SETTINGS COMPONENT SIGNALS

GENERATE PROJECT

Output Error List

Ready

Click on CLKCTRL and scroll down to show clock configuration options.

Setup the Clock Settings (continue)

The screenshot shows the Atmel Studio IDE with the 'CLKCTRL' component configuration page open. The page is titled 'CLKCTRL Clock Configuration' and has three tabs: 'GENERAL', 'COMPONENT SETTINGS', and 'COMPONENT SIGNALS'. The 'GENERAL' tab is active, showing the 'DRIVERS:CLKCTRL:INIT (CLKCTRL) CONFIGURATION ON CLKCTRL' section. This section is divided into two columns. The left column contains the 'MAIN CLOCK CONFIGURATION' and '20MHZ OSCILLATOR CONFIGURATION'. The right column contains 'OSCULP32K DIV 32', '16MHZ OSCILLATOR CONFIGURATION', '32.768KHZ CRYSTAL OSCILLATOR CONFIGURAT...', and 'TCD0 CLOCK CONFIGURATION'. Each configuration section has an 'Enable' checkbox and a 'CLKSEL: Main Clock Source' dropdown menu. The 'MAIN CLOCK CONFIGURATION' is enabled, and its clock source is set to '20MHz Internal Oscillato'. The '20MHZ OSCILLATOR CONFIGURATION' is also enabled, and its 'RUNSTDBY: Run standby' checkbox is unchecked. The 'OSCULP32K DIV 32' is enabled, and its clock source is set to '32KHz Internal Ultra Low'. The '16MHZ OSCILLATOR CONFIGURATION' is disabled. The '32.768KHZ CRYSTAL OSCILLATOR CONFIGURAT...' is disabled. The 'TCD0 CLOCK CONFIGURATION' is enabled, and its clock source is set to '20MHz Internal Oscillato'. The 'SYNCPRES: Synchronization prescaler' is set to '1'. At the bottom of the page, there is a 'GENERATE PROJECT' button. The status bar at the bottom indicates 'Ready'.

Atmel | START ATtiny1626

What's New | Help

MY SOFTWARE COMPONENTS

CLKCTRL

Clock Configuration

GENERAL **COMPONENT SETTINGS** **COMPONENT SIGNALS**

User guide

Driver: Drivers:CLKCTRL:Init

DRIVERS:CLKCTRL:INIT (CLKCTRL) CONFIGURATION ON CLKCTRL

MAIN CLOCK CONFIGURATION

Enable: ☒

CLKSEL: Main Clock Source: 20MHz Internal Oscillato

PEN: Prescaler enable: ☒

PDIV: Prescaler division: 4

LOCKEN: lock enable: ☐

CLKOUT: System clock out: ☐

20MHZ OSCILLATOR CONFIGURATION

Enable: ☒

RUNSTDBY: Run standby: ☐

OSCULP32K DIV 32

Enable: ☒

CLKSEL: Main Clock Source: 32KHz Internal Ultra Low

16MHZ OSCILLATOR CONFIGURATION

Enable: ☐

32.768KHZ CRYSTAL OSCILLATOR CONFIGURAT...

Enable: ☐

TCD0 CLOCK CONFIGURATION

Enable: ☒

CLKSEL: clock select: 20MHz Internal Oscillato

SYNCPRES: Synchronization prescaler: 1

GENERATE PROJECT

Output Error List

Ready

Setup the clock as required.

Setup the Clock Settings (continue)

ADC_Example - AtmelStudio Advanced Mode Quick Launch (Ctrl+Q)

File Edit View VAssistX ASF Project Build Debug Tools Window Help

clock_config.h clkctrl.c driver_init.c main.c adc_basic.c Atmel | START iotn1626.h

Atmel | START ATtiny1626

What's New | Help

CLOCK CONFIGURATOR

Zoom in Zoom out Reset

OSCILLATORS

- External Clock (EXTCLK) 32.768 kHz
- 20MHz Internal Oscillator (OSC20M) 20 MHz
- 16MHz Internal Oscillator (OSC16M) 16 MHz
- 32KHz Internal Ultra Low Power Oscillator (OSCULP32K) 32.768 kHz
- ☐ 32.768kHz External Crystal Oscillator

SOURCES

- ☒ Main Clock (CLK_MAIN) 5 MHz
- ☒ 32KHz divided by 32 1.024 kHz
- ☒ TCD0 Clock (CLK_TCD0) 20 MHz
- ☐ RTC Clock
Component RTC not loaded

Reset clock settings

COMPONENTS

- CPU 5 MHz
- RAM 5 MHz
- NVM 5 MHz
- ADC_0 ADC 5 MHz
- WDT_0 WDT 1.024 kHz
- TIMER_0 TCA 5 MHz

GENERATE PROJECT

Output Error List

Ready

Use the Clock Configurator to easily setup the clock.

Use the Pinmux Configurator to Mux Pins

Atmel | START - AtmelStudio

Advanced Mode Quick Launch (Ctrl+Q)

File Edit View VAssistX ASF Project Debug Tools Window Help

Atmel | START Start Page

Atmel | START ATtiny1626

What's New | Help

PINMUX CONFIGURATOR

Pin			Board		Mode	SW c...
# ↑	Pad	Use..	Header	Label		
<input type="checkbox"/> ADC_0						
1	PA2				Analog	AIN/2
2	PA3				Analog	AIN/3
20	PA1				Analog	AIN/1
<input type="checkbox"/> No software components						
3	GND					
4	VCC					
5	PA4					
6	PA5					
7	PA6					
8	PA7					
9	PB5					
10	PB4					
11	PB3					
...	...					

Show labels... Zoom in Zoom out Auto fit

Diagram of ATtiny1626 pinmux configuration showing pins PA1, PA2, PA3, PA4, PA5, PA6, PA7, PA0, PC3, PC2, PC1, PB0, PB1, PB2, PB3, PB4, PB5, and PC0.

GENERATE PROJECT

Output Error List

Ready

Use the Pinmux Configurator to check the pins mux and to label them.

Generate the Project Files

The screenshot shows the Atmel Studio interface. The top menu bar includes File, Edit, View, VAssistX, ASF, Project, Debug, Tools, Window, and Help. The toolbar contains various icons for file operations, editing, and debugging. The main window displays the 'MY SOFTWARE COMPONENTS' section for an ATtiny1626 project. A legend on the left identifies component types: Application (dark grey), Middleware (light blue), Driver (green), and System driver (light grey). The 'MY PROJECT' node is expanded, showing three sub-components: ADC_0, VREF_0, and TIMER_0, all marked as Drivers. Below these are four System driver components: CLKCTRL, CPUINT, BOD, and SLPCTRL. A 'GENERATE PROJECT' button is located at the bottom of the main window. The status bar at the bottom indicates 'Ready'.

Atmel | START - AtmelStudio Advanced Mode Quick Launch (Ctrl+Q)

File Edit View VAssistX ASF Project Debug Tools Window Help

Atmel | START Start Page

Atmel | START ATtiny1626

What's New | Help

MY SOFTWARE COMPONENTS

- Application
- Middleware
- Driver
- System driver

+ Add software component

Show system drivers ☐

Show hardware ☐

MY PROJECT

ADC_0 VREF_0 TIMER_0

CLKCTRL CPUINT BOD SLPCTRL

SELECTED BOARD: CUSTOM BOARD

You are using a custom board. Predefined boards can be selected from the front page when you create a new project.

GENERATE PROJECT

Output Error List

Ready

Click on GENERATE PROJECT to generate the project files.

Name the Project and Save it

The screenshot shows the Atmel Studio 7.0 interface. The top menu bar includes File, Edit, View, VAssistX, ASF, Project, Debug, Tools, Window, and Help. The toolbar contains various icons for file operations, debugging, and development. The main window displays the 'Atmel | START' page for the ATtiny1626 board. A 'DASHBOARD' sidebar is visible on the left. The 'MY SOFTWARE COMPONENTS' section is active, showing a list of components. A 'New Atmel Start Project' dialog box is open, prompting for project details. The dialog box has fields for Project Name, Location, Solution, and Solution Name. The 'Project Name' field is filled with 'ADC_Example'. The 'Location' field is filled with 'C:\Users\YankiMainDesk\Documents\Atmel Studio\7.0'. The 'Solution' dropdown is set to 'Create New Solution'. The 'Solution Name' field is filled with 'ADC_Example'. There is a 'Browse' button next to the 'Location' field. At the bottom of the dialog box are 'OK' and 'Cancel' buttons. Below the dialog box, the 'SELECTED BOARD: CUSTOM BOARD' section is visible, with a message: 'You are using a custom board. Predefined boards can be selected from the front page when you create a new project.' A 'GENERATE PROJECT' button is located at the bottom right of the main window.

Atmel | START - AtmelStudio Advanced Mode Quick Launch (Ctrl+Q)

File Edit View VAssistX ASF Project Debug Tools Window Help

Atmel | START Start Page

Atmel | START ATtiny1626 What's New | Help

MY SOFTWARE COMPONENTS

Atmel Start Importer

New Atmel Start Project

Project Name: ADC_Example

Location: C:\Users\YankiMainDesk\Documents\Atmel Studio\7.0 Browse

Solution: Create New Solution

Solution Name: ADC_Example

[View Project Summary](#)

OK Cancel

SELECTED BOARD: CUSTOM BOARD

You are using a custom board. Predefined boards can be selected from the front page when you create a new project.

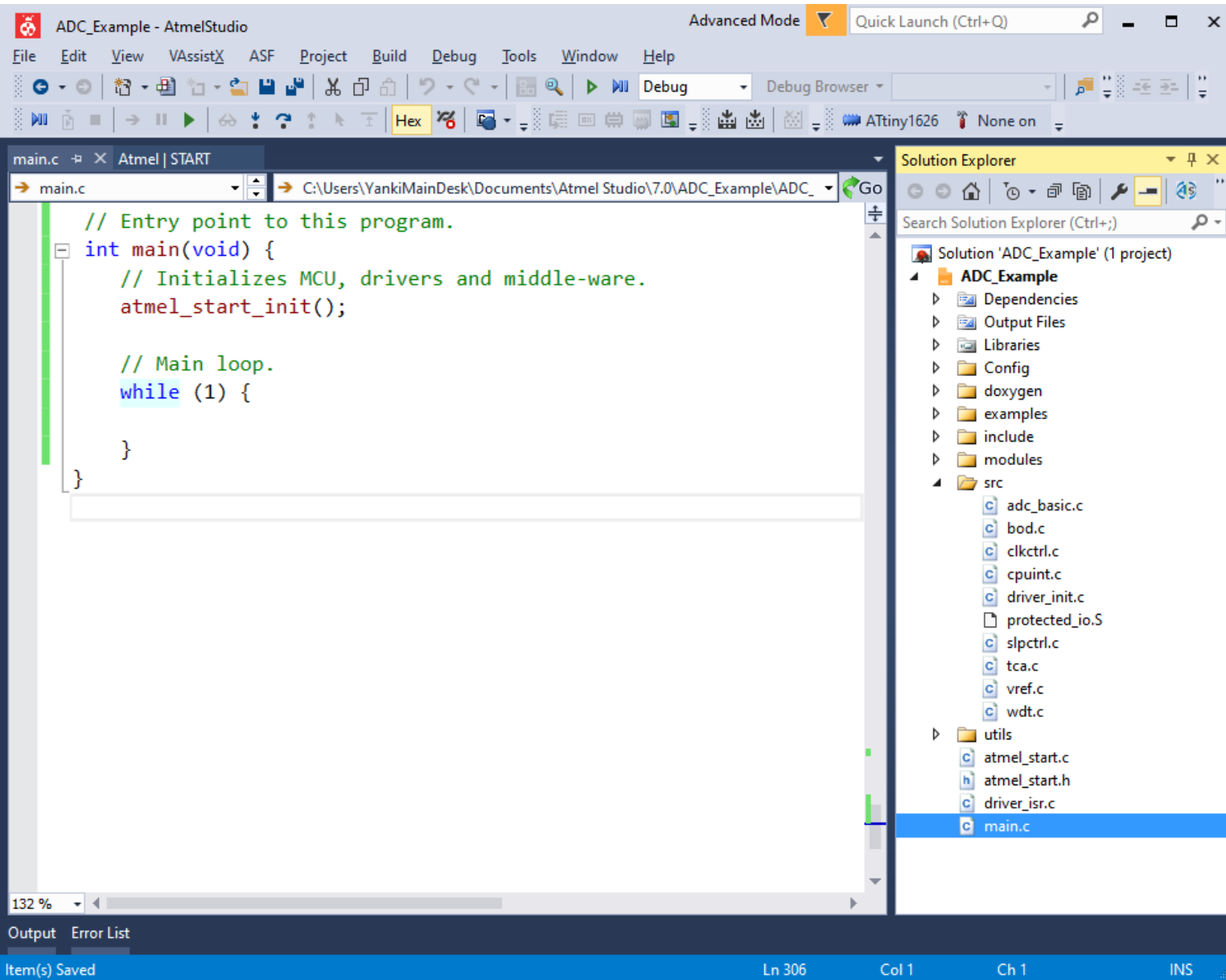
GENERATE PROJECT

Output Error List

Ready

Enter the name of the project and click OK to generate the project.

Use the Solution Explorer to Browse the Project



Use the Solution Explorer to browse the project files. The IDE setup the code skeleton automatically.

Entry Point to the Program Code

```
#include <atmel_start.h>
```

```
... Missing code here ...
```

```
// Entry point to this program.
```

```
int main(void) {  
    // Initializes MCU, drivers and middle-ware.  
    atmel_start_init();  
    // Initialize GPIOs not used by peripherals.  
    GPIOs_Init();  
    // Initializes the TCA0 to overflow every 1 milliseconds.  
    // This timer is used as a house keeper timer.  
    TCA0_Init(1);  
    // Clear the Overflow Interrupt for TCA0.  
    TCA0.SINGLE.INTFLAGS = TCA_SINGLE_OVF_bm;  
    // Enable global interrupts.  
    sei();  
  
    // Initializes the Watch Dog to 1K cycles (1.0s).  
    WDT_Init(WDT_PERIOD_1KCLK_gc);  
    // Main loop.  
    while (1) {  
        // Measure voltage.  
        ADC0_Measure_Voltage();  
        // Reset Watch Dog.  
        WDT_Reset();  
    }  
}
```

ADC Pins Initialization Code

```
/* Configure pins and initialize registers */
void ADC_0_initialization(void) {

    // Disable digital input buffer
    PA1_set_isc(PORT_ISC_INPUT_DISABLE_gc);
    // Disable pull-up resistor
    PA1_set_pull_mode(PORT_PULL_OFF);

    // Disable digital input buffer
    PA2_set_isc(PORT_ISC_INPUT_DISABLE_gc);
    // Disable pull-up resistor
    PA2_set_pull_mode(PORT_PULL_OFF);

    // Disable digital input buffer
    PA3_set_isc(PORT_ISC_INPUT_DISABLE_gc);
    // Disable pull-up resistor
    PA3_set_pull_mode(PORT_PULL_OFF);

    ADC_0_init();
}
```

ADC Peripheral Initialization Code

```
/* brief Initialize ADC interface */
int8_t ADC_0_init() {

    ADC0.CTRLB = ADC_PRESC_DIV2_gc; /* System clock divided by 2 */

    ADC0.CTRLF = ADC_SAMPNUM_NONE_gc /* No accumulation */
        | 0 << ADC_FREERUN_bp; /* ADC Freerun mode: disabled */

    ADC0.CTRLC = ADC_REFSEL_2500MV_gc /* 2.5V */
        | ADC_TIMEBASE_VALUE; /* timebase value */

    ADC0.CTRLD = ADC_WINCM_NONE_gc /* No Window Comparison */
        | ADC_WINSRC_RESULT_gc; /* Result register */
    ADC0.CTRLE = 0x1b; /* Sample Duration: 0x1b */
    ADC0.DBGCTRL = 0 << ADC_DBGRUN_bp; /* Debug run: disabled */
    ADC0.COMMAND = 0 << ADC_DIFF_bp /* Diff. ADC Conv: disabled */
        | ADC_MODE_SINGLE_8BIT_gc /* Single Conversion 8-bit */
        | ADC_START_STOP_gc; /* Stop an ongoing conversion */
    ADC0.INTCTRL = 0 << ADC_RESRDY_bp /* Result Rdy Int disabled */
        | 0 << ADC_WCMP_bp; /* Window Comp Int Enable: disabled */
    ADC0.MUXPOS = ADC_VIA_ADC_gc /* Via ADC */
        | ADC_MUXPOS_AIN1_gc; /* ADC input pin 1 */
    ADC0.MUXNEG = ADC_VIA_ADC_gc /* Via ADC */
        | ADC_MUXNEG_AIN1_gc; /* ADC input pin 1 */
    ADC0.WINHT = 0x0; /* Window Comparator High Threshold: 0x0 */
    ADC0.WINLT = 0x0; /* Window Comparator Low Threshold: 0x0 */
    ADC0.CTRLA = 1 << ADC_ENABLE_bp /* ADC Enable: enabled */
        | 0 << ADC_RUNSTDBY_bp; /* Run standby mode: disabled */

    return 0;
}
```

Clock Initialization Code

```
/* brief Initialize clkctrl interface */
int8_t CLKCTRL_init() {

    ccp_write_io((void*)&(CLKCTRL.OSC32KCTRLA),0 <<
    CLKCTRL_RUNSTDBY_bp /* Run standby: disabled */);

    ccp_write_io((void*)&(CLKCTRL.XOSC32KCTRLA),
        CLKCTRL_CSUT_1K_gc /* 1k cycles */
    | 0 << CLKCTRL_ENABLE_bp /* Enable: disabled */
    | 0 << CLKCTRL_RUNSTDBY_bp /* Run standby: disabled */
    | 0 << CLKCTRL_SEL_bp /* Select: disabled */);

    ccp_write_io((void*)&(CLKCTRL.OSC20MCTRLA),0 <<
    CLKCTRL_RUNSTDBY_bp /* Run standby: disabled */);

    ccp_write_io((void *)&(CLKCTRL.MCLKCTRLB),
        CLKCTRL_PDIV_4X_gc /* 4 */
    | 1 << CLKCTRL_PEN_bp /* Prescaler enabled */);

    ccp_write_io((void*)&(CLKCTRL.MCLKCTRLA),CLKCTRL_CLKSEL_OSC20M_
    gc /* 20MHz Internal Oscillator (OSC20M) */
    | 0 << CLKCTRL_CLKOUT_bp /* System clock out: disabled */);

    ccp_write_io((void*)&(CLKCTRL.MCLKLOCK),0 << CLKCTRL_LOCKEN_bp
    /* lock enable: disabled */);

    return 0;
}
```

Function to Measure Voltage with ADC Code

```
/* function is designed to measure voltage using the ADC
   peripheral. */
void ADC0_Measure_Voltage(void) {
    // Variable declaration and initialization in this scope (local
    // variables).
    adc_result_t adc_result = 0;

    // Switch to the appropriate state.
    switch(adc0_state) {
        // State to select the ADC channel to measure.
        case 0:
            if(adc0_channel == 0) {
                ADC0.MUXPOS = ADC_MEAS_CHANNEL_1;
            } else if(adc0_channel == 1) {
                ADC0.MUXPOS = ADC_MEAS_CHANNEL_2;
            } else if(adc0_channel == 2) {
                ADC0.MUXPOS = ADC_MEAS_CHANNEL_3;
            } else {
                // nothing to do here.
                // The code execution should never enter here.
                asm("nop");
            }
            adc0_state = 1;
            adc0_ms_counter = 0;
            break;
    }
}
```

Function to Measure Voltage with ADC Code (cont.)

```
// Wait some time for the ADC Muxer to change channel  
// before to start a new conversion.
```

```
case 1:
```

```
    if(adc0_ms_counter >= 1) {  
        // Start a new conversion.  
        ADC0.COMMAND |= ADC_START_IMMEDIATE_gc;  
        // Change to next state.  
        adc0_state = 2;  
    }  
    break;
```

```
// Check if conversion is done.
```

```
case 2:
```

```
    if(ADC_0_is_conversion_done()) {  
        // Change state.  
        adc0_state = 3;  
    }  
    break;
```


Function to Measure Voltage with ADC Code (cont.)

```
// State to get the conversion result.
case 3:
    // Get conversion from specified ADC channel.
    adc_result = ADC_0_get_conversion_result();

    // Formula to calculate the voltage at the ADC pin is:
    // Vadc_pin = (ADC Result * Vref) / ADC_Resolution
    // Vref = 2.5V, ADC_Resolution = 12 bits (4095)
    // Vadc_pin = (ADC Result * 2.5) / 4095

    // The voltage applied to the ADC pins is reduced by
    // a voltage divider. The voltage is reduced by a
    // factor of 11.
    //  $R1 + R2 / R2 = (100K + 10K) / 10K = 11$ 
    //  $R3 + R4 / R4 = (100K + 10K) / 10K = 11$ 
    //  $R5 + R6 / R6 = (100K + 10K) / 10K = 11$ 
```

Function to Measure Voltage with ADC Code (cont.)

```
if(adc0_channel == 0) {
    // Calculate voltage Vin_1 (Refer to schematic).
    adc0_voltage_1 = ((adc_result * 2.5) / 4095) * 11;
} else if(adc0_channel == 1) {
    // Calculate voltage Vin_2 (Refer to schematic).
    adc0_voltage_2 = ((adc_result * 2.5) / 4095) * 11;
} else if(adc0_channel == 2) {
    // Calculate voltage Vin_2 (Refer to schematic).
    adc0_voltage_2 = ((adc_result * 2.5) / 4095) * 11;
} else {
    // nothing to do here.
    // The code execution should never enter here.
    asm("nop");
}
// Increment to use next ADC channel next time.
if(adc0_channel < 2) {
    adc0_channel++;
} else {
    adc0_channel = 0;
}
// Change state.
adc0_state = 4;
// Clear counter.
adc0_ms_counter = 0;
// Clear the Result Ready Interrupt Flag.
ADC0.INTFLAGS |= ADC_RESRDY_bm;
break;
```

Function to Measure Voltage with ADC Code (cont.)

```
// Wait a predefined time to start next measurement.
case 4:
    if(adc0_ms_counter > 1) { // 1 milli seconds
        // Change state.
        adc0_state = 0;
    }
    break;

// In case of an error.
default:
    // Change state.
    adc0_state = 0;
    // Clear the Result Ready Interrupt Flag.
    ADC0.INTFLAGS |= ADC_RESRDY_bm;
    break;
}
}
```

Conclusion

The Atmel Studio 7 IDP is a great tool to quickly develop projects for AVR MCUs.

The ATtiny1626 is an excellent MCU to use in projects. It uses the latest technologies from Microchip with a flexible and low-power architecture, including an Event System, advanced digital peripherals, and accurate analog features such as a 12-bit differential ADC with Programmable Gain Amplifier (PGA).

References

1. Complete Code of Project

https://github.com/god233012yamil/attiny1626_adc_example

2. Atmel® Studio 7

<https://microchipdeveloper.com/atstudio:studio7intro>

3. ATTINY1626 with 12-bit diff ADC with PGA

<https://www.microchip.com/en-us/product/ATtiny1626>

4. ATTINY1627 CURIOSITY NANO EVALUATION KIT

<https://www.microchip.com/en-us/development-tool/DM080104>

5. Atmel-ICE is a powerful development tool for debugging and programming

<https://www.microchip.com/en-us/development-tool/ATATMEL-ICE>