
Migrating an Atmel Software Frameworks (ASF) Project from ASFv3.3 to ASFv4

INTRODUCTION

This application note provides an introductory look into the migration of an Atmel Start ASFv3 project to an ASFv4 project.

Due to ASFv4 optimizations for code density and added flexibility, the two frameworks are mutually exclusive. Which means an ASF v3.3 project cannot be built upon by ASF v4.0. This will require a “ground up” approach to project migration.

ASFv3

The ASF 3 architecture was developed with code-size, performance and low power optimizations as primary goals. Peripheral drivers and middleware have been developed to provide a reduced design cycle when using ASF-enabled projects.

The intention of ASF is to provide a rich set of proven drivers and code modules developed by Atmel experts to reduce customer design time. It simplifies the usage of microcontrollers, providing an abstraction to the hardware and high-value middleware.

ASF consists of source code modules and applications demonstrating the use of these.

- **Drivers** is composed of a `driver.c` and `driver.h` file that provides low level register interface functions to access a peripheral or device specific feature. The services and components will interface the drivers.
- **Services** is a module type which provides more application oriented software such as a USB classes, FAT file system, architecture optimized DSP library, graphical library, etc.
- **Components** is a module type which provides software drivers to access external hardware components such as memory (e.g., Atmel DataFlash®, SDRAM, SRAM, and NAND Flash), displays, sensors, wireless, etc.
- **Boards** contains mapping of all digital and analog peripheral to each I/O pin of Atmel's development kits.

ASFv4

In Atmel Start, the drivers and software stacks are provided as a part of the next iteration of Atmel Software Framework (ASFv4). This version is built from scratch and is a complete redesign and implementation of the whole framework to resolve issues reported by users and contributors of older ASF versions and to better integrate with the Atmel Start Web user interface. Still, it has been a goal to keep ASFv4 feel familiar for experienced ASF users, yet easy to get started with for new users. Some changes in ASFv4 have been necessary to meet the requirements for this version, the most important changes are listed in [ASFv4 vs ASFv3 Benchmark](#) of the *Atmel Start User Guide*.

ASFv4 is tightly integrated into Atmel Start, which means that the ASFv4 code can be much more tailored to the user's specification than before. For instance, instead of using C preprocessor conditional expressions to enable/disable code blocks, disabled code blocks can be removed entirely from the project source which results in cleaner and easier to read code. The integration into Atmel Start means that software configuration is done in a much more user friendly environment and the only configuration information loaded on the device is the raw peripheral register content which makes the firmware image much more compact.

One important issue we have addressed is the memory footprint and performance of ASF-based code. Flash requirements for running ASFv3-based code has been deemed too high by many users. This has been addressed by using code generation and changing the way peripherals are initialized. Performance issues that has been reported is typically high interrupt latency/slow code execution which has been resolved by making the interrupt handlers smaller and less complex.

Migration

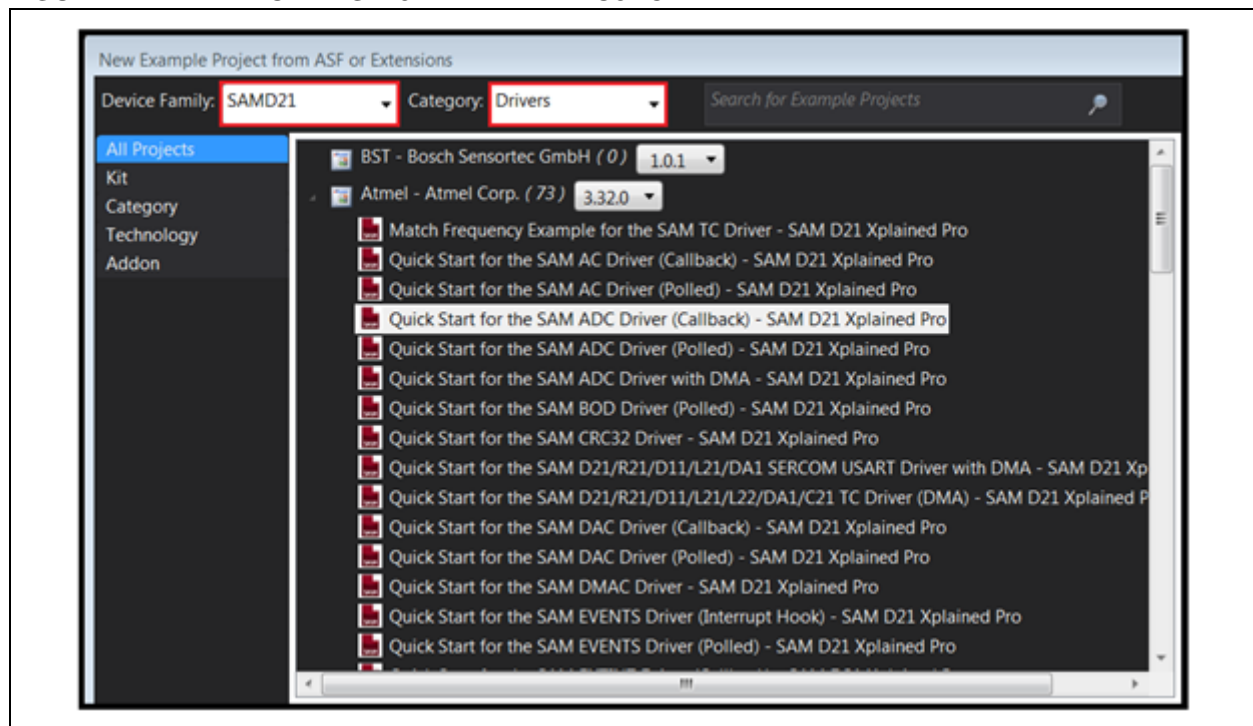
Unfortunately, there is not a way to directly migrate an ASFv3 project to an ASFv4 project. With the new Architecture and APIs, the only method of migration is to rebuild the ASFv3 project from scratch with the online Atmel Start tool (start.atmel.com). While this may present a challenge now, future migrations will be much easier between devices using an ASFv4 project than a migration between devices using an ASFv3 project.

STARTING THE EXAMPLE PROJECT

To step through the migration process, an example project will be used from the ASF v3.3 example project list. The example project used will be selected for use with the SAMD21 Xplained Pro Evaluation Kit board.

1. In Atmel Studio, load the ASFv3 Project by selecting **File > New > Example Project**.
2. In the Device Family menu, select the SAMD21, as shown in Figure 1.
3. Click All Projects to display all Atmel Projects and then select Quick Start for the SAM ADC Driver (Callback) - SAM D21 Xplained Pro.
4. Click **OK** to create the project in the default Studio workspace.
5. Compile and run the example project.

FIGURE 1: IMPORT ASFv3 EXAMPLE PROJECT



6. Application Description

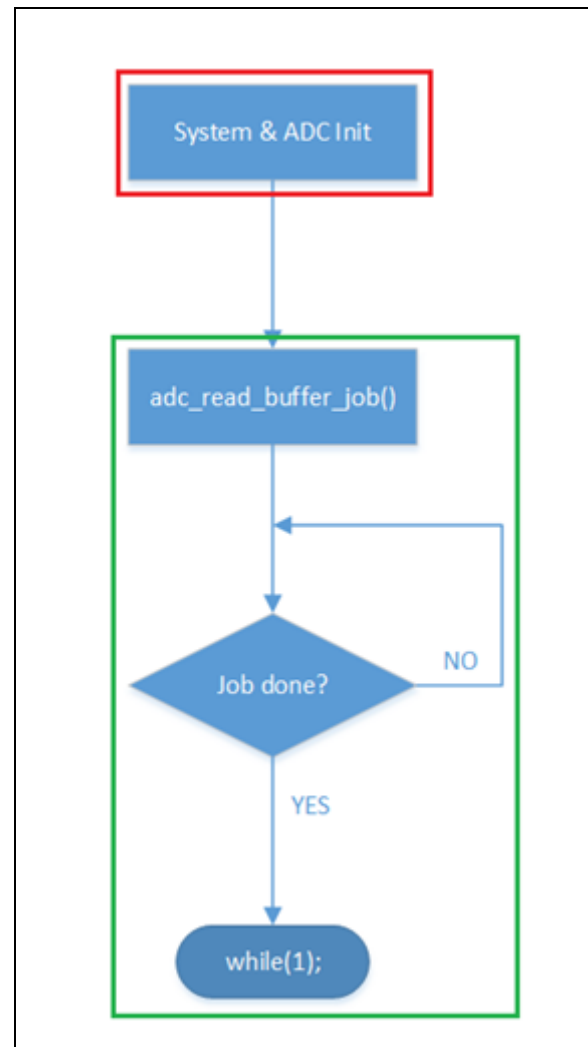
The example ADC program flowchart is shown in [Figure 2](#). The example application will start the ADC to collect a set of samples and then wait forever once the data collection is complete.

The application is very simple, but will need to be broken into two steps for the migration. Step one, outlined in red, will be to extract the needed information from the System and ADC initialization in ASFv3 for later use in Atmel Start. Step 2, outlined in green, will be the functional verification of the API's needed in the application and will be completed after the Atmel Start project is imported into Atmel Studio.

7. Application Components.

The function of the Application will require an understanding of the System (clock, etc.), as well as the ADC peripheral. The red block of [Figure 2](#) includes system drivers required for any project. Some configuration will be required to ensure the system can provide the necessary clocks and configurations to the peripherals. The ADC is also included in this block, and will require evaluation to transfer the needed configuration for the application.

FIGURE 2: EXAMPLE ADC PROGRAM FLOWCHART



EXTRACT ASFv3 CONFIGURATION OF SYSTEM AND ADC

1. Capture the Clock configuration

The system clock settings for ASFv3 can be located in the `conf_clocks.h` file of the example project. From here it can be seen the Main Clock (GCLK_0) is sourced by the internal 8 MHz oscillator.

Figure 3 shows that GCLK_0 is not prescaled and is running the SAMD21 Core at 8 MHz. These settings will be used in Atmel Start to configure the clock tree of the ADC project. For the ADC peripheral in this project, GCLK_0 will be used as the clock source. These values can also be found in the I/O window at run-time under Generic Clock Generator (GCLK) and System Control (SYSCTRL).

2. Capture the Peripheral configuration.

Peripheral modules in ASFv3 use a configuration structure to initialize the peripheral for the application. ASFv3 contains default values that are written to the peripheral for initialization. It is important to note the peripheral configuration structures are populated at run time. ASFv4 has taken a different approach and handles all application configuration with preprocessor macros.

In this example application the `adc_config` structure is used to initialize the ADC in the `configure_adc()` function. Using a Watch window during the execution of the project, the initialization parameters are captured in Figure 4. An effort has been made in ASFv4 development to maintain the configuration value naming convention. With that said, the values in the middle column of Figure 4 will be used to set up the ADC in Atmel Start.

The ADC register Figure 5 displays the register values after initialization with ASFv3 and will be used to verify ADC initialization with Atmel Start.

FIGURE 3: SYSTEM CLOCK CONFIGURATION

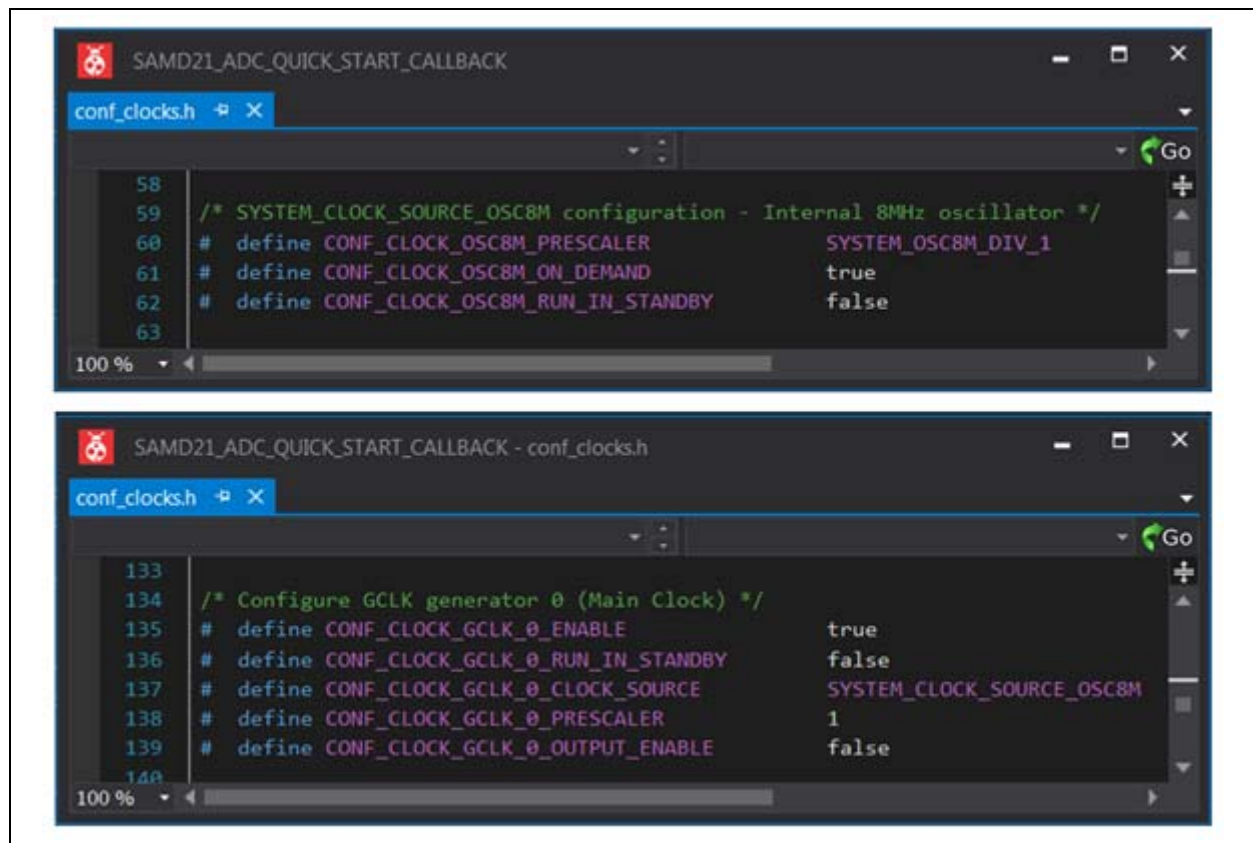
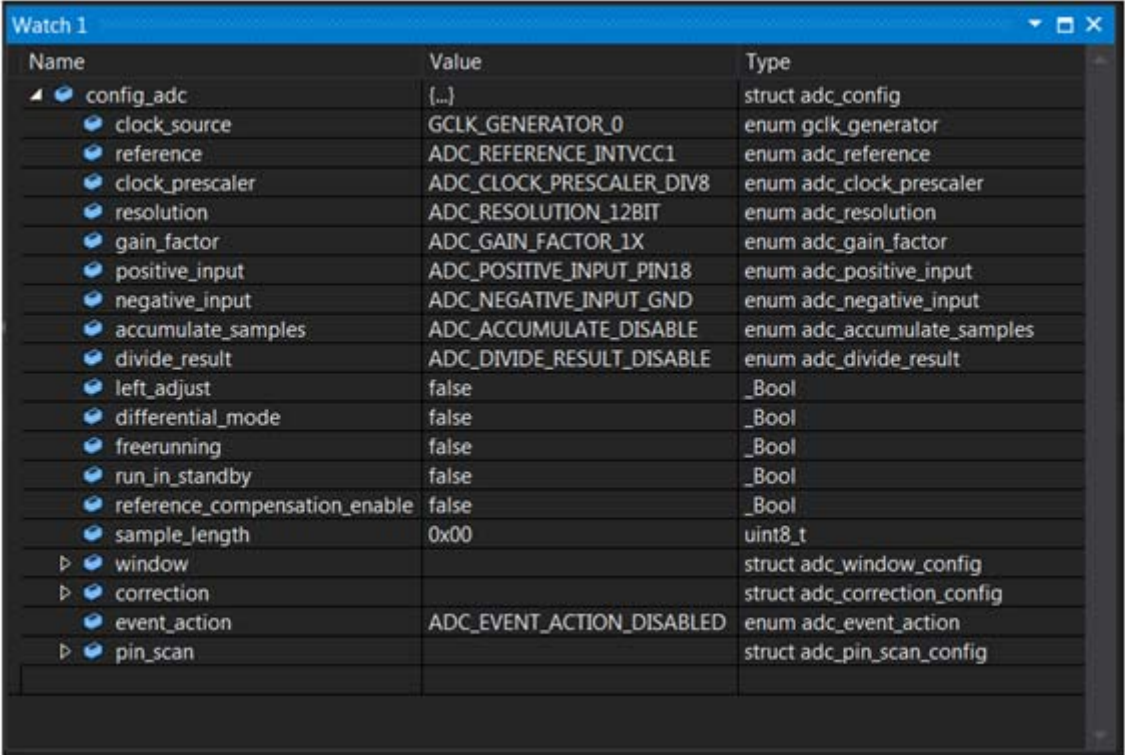


FIGURE 4: ASFv3 ADC CONFIGURATION PARAMETERS



Name	Value	Type
config_adc	[...]	struct adc_config
clock_source	GCLK_GENERATOR_0	enum gclk_generator
reference	ADC_REFERENCE_INTVCC1	enum adc_reference
clock_prescaler	ADC_CLOCK_PRESCALER_DIV8	enum adc_clock_prescaler
resolution	ADC_RESOLUTION_12BIT	enum adc_resolution
gain_factor	ADC_GAIN_FACTOR_1X	enum adc_gain_factor
positive_input	ADC_POSITIVE_INPUT_PIN18	enum adc_positive_input
negative_input	ADC_NEGATIVE_INPUT_GND	enum adc_negative_input
accumulate_samples	ADC_ACCUMULATE_DISABLE	enum adc_accumulate_samples
divide_result	ADC_DIVIDE_RESULT_DISABLE	enum adc_divide_result
left_adjust	false	_Bool
differential_mode	false	_Bool
freerunning	false	_Bool
run_in_standby	false	_Bool
reference_compensation_enable	false	_Bool
sample_length	0x00	uint8_t
window		struct adc_window_config
correction		struct adc_correction_config
event_action	ADC_EVENT_ACTION_DISABLED	enum adc_event_action
pin_scan		struct adc_pin_scan_config

FIGURE 5: ASFv3 ADC REGISTER VALUES



Name	Address	Value	Bits
CTRLA	0x42004000	0x02	
REFCTRL	0x42004001	0x02	
AVGCTRL	0x42004002	0x00	
SAMPCTRL	0x42004003	0x00	
CTRLB	0x42004004	0x0100	
WINCTRL	0x42004008	0x00	
SWTRIG	0x4200400C	0x00	
INPUTCTRL	0x42004010	0x00001806	
EVCTRL	0x42004014	0x00	
INTENCLR	0x42004016	0x00	
INTENSET	0x42004017	0x00	
INTFLAG	0x42004018	0x08	
STATUS	0x42004019	0x00	
RESULT	0x4200401A	0x0000	
WINLT	0x4200401C	0x0000	
WINUT	0x42004020	0x0000	
GAINCORR	0x42004024	0x0000	
OFFSETCO...	0x42004026	0x0000	
CALIB	0x42004028	0x0388	
DBGCTRL	0x4200402A	0x00	

3. Capture the pin configuration.

The example project chosen is only using one ADC channel for the application. In this case, the pin assignments are easy to find and are located in the `config_adc` structure referenced earlier or the `configure_adc()` function. ASFv3 example projects using Atmel hardware, like the Xplained Pro, may have this information where the hardware is initialized in the `system_board_init()` function.

ATMEL START - SYSTEM INITIALIZATION

1. Create a new Atmel Start project by browsing to start.atmel.com and then click the **Create New Project** tab, as shown in Figure 6.
2. In the Create New Projects window, under Results section, search for SAM D21 Xplained Pro and select SAM D21 Xplained Pro, and then Click **Create New Project**, see Figure 7.
3. The New Project Dashboard will be displayed as shown in Figure 8.

FIGURE 6: ATMEL START HOME PAGE



FIGURE 7: ATMEL START CREATE NEW PROJECT

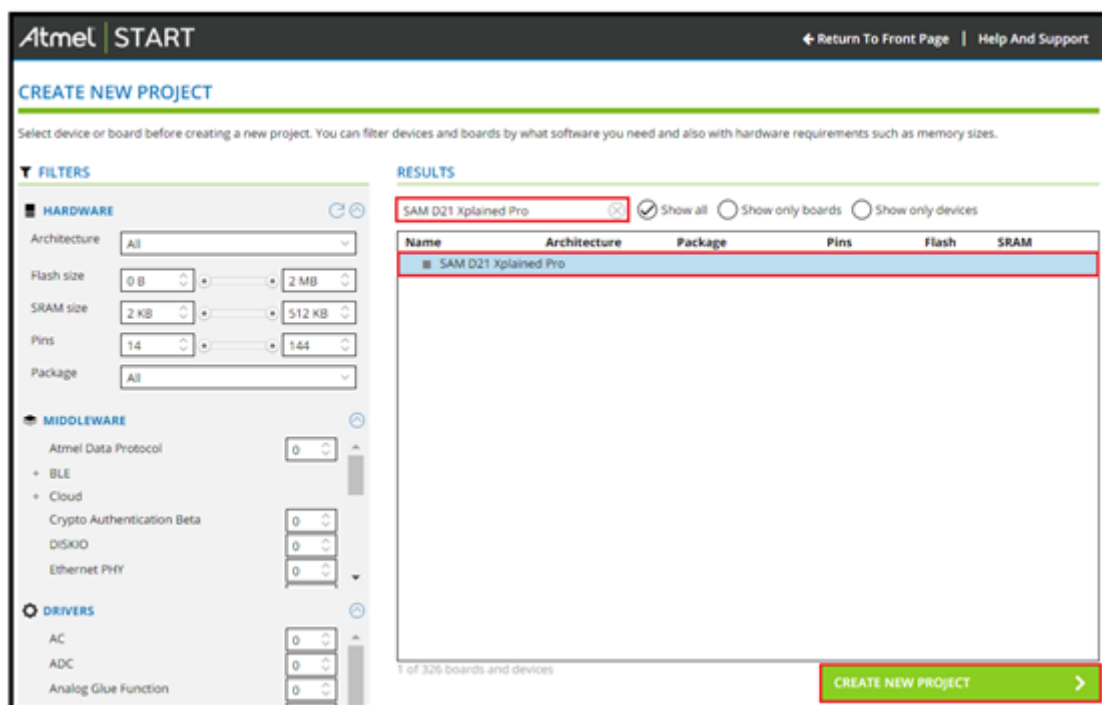
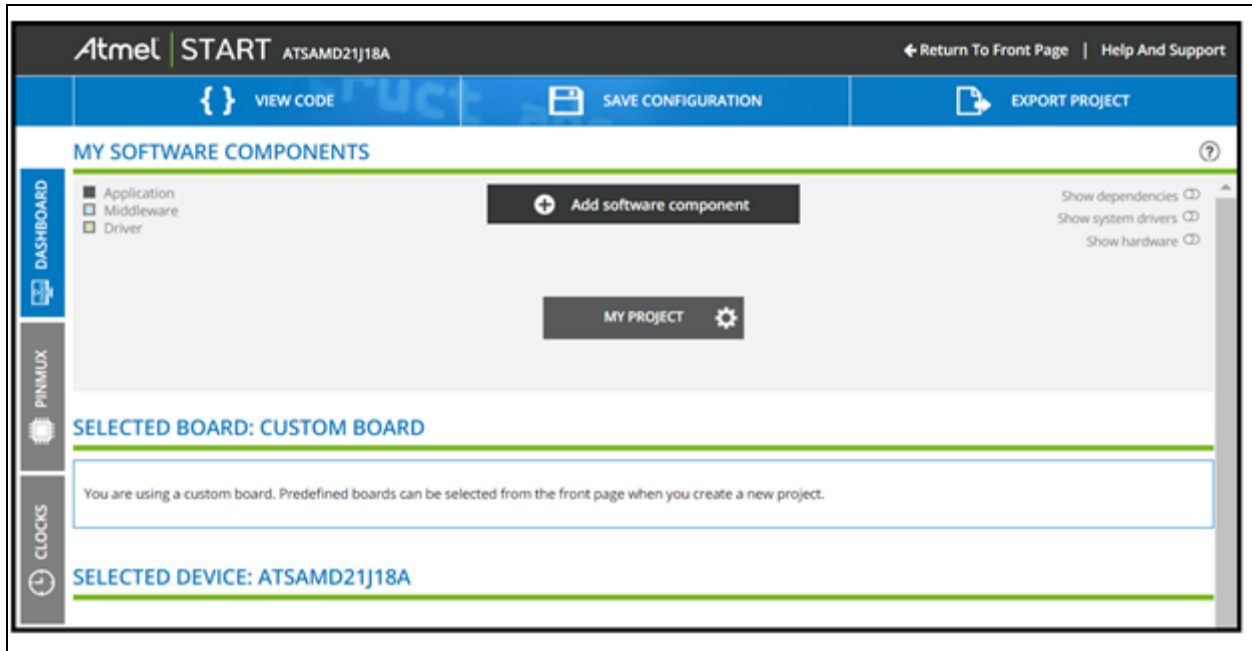
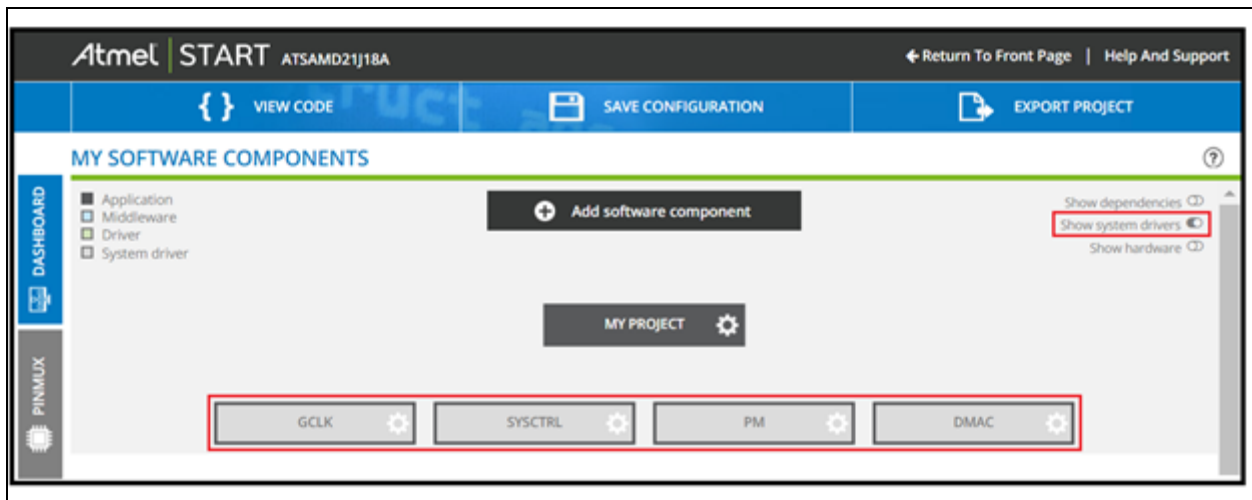


FIGURE 8: DASHBOARD OF NEW PROJECT



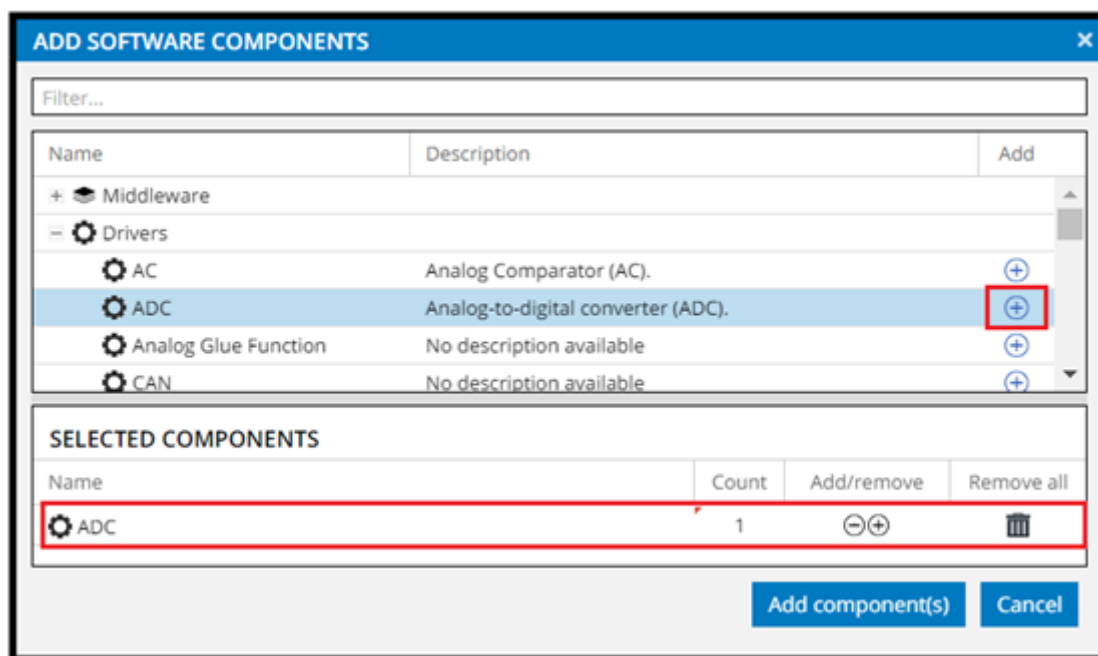
Please note, as in ASFv3, several system drivers are included in every ASFv4 project. These drivers are initially hidden in the Start setup. To see the automatically included system drivers, click the Show system drivers indicator. The four components (SYSCtrl, DMAC, PM, and GCLK) will allow for clock, bus, NVM, and DMA settings to be configured in the Dashboard screen, as shown in Figure 9.

FIGURE 9: INCLUDED SYSTEM DRIVERS



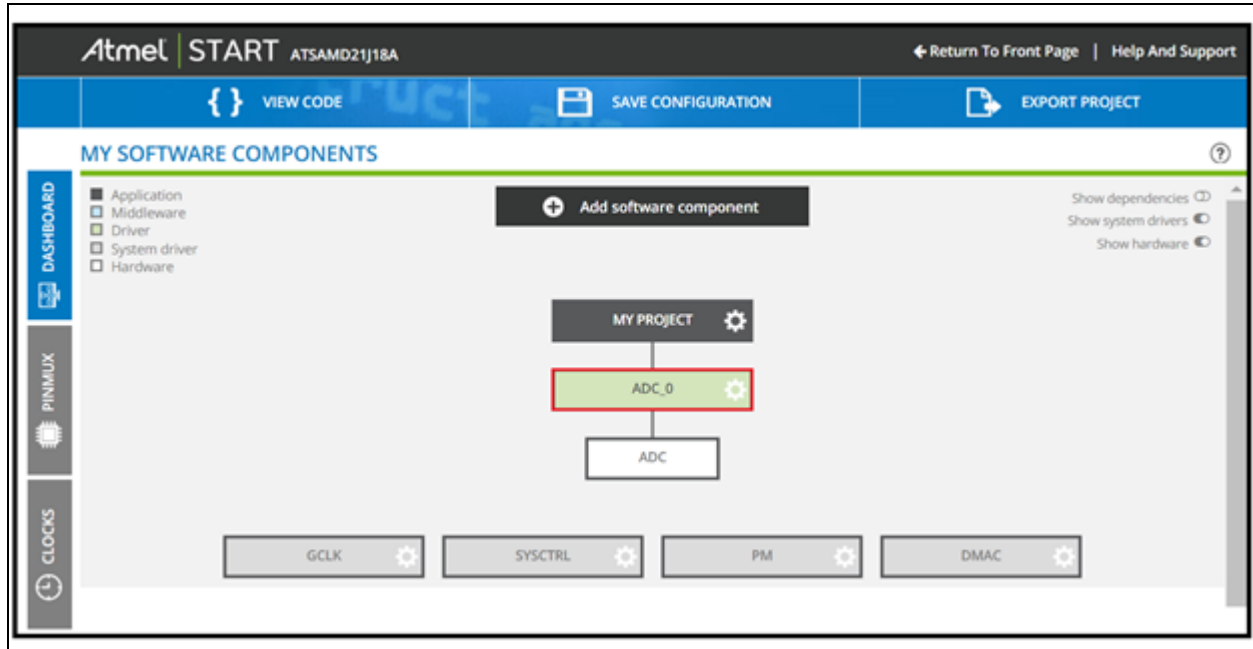
4. From the **Dashboard** tab, Click **Add software component** to add Peripheral modules to the project. The Add Software Components dialog will display available drivers for the SAMD21, as shown in [Figure 10](#).
5. Click “+” to display all the Drivers, and then select ADC by clicking the “+” symbol. The ADC module will be displayed under the Selected Components section.
6. Click **Add Component(s)** to add selected components to the project.

FIGURE 10: ATMEL START ADD SOFTWARE COMPONENTS



- Configure the peripheral modules. Once the ADC module is added, it will be displayed under the **My Project** tab, as shown in Figure 11.

FIGURE 11: CONFIGURE THE PERIPHERAL MODULE



- Click the **ADC_0** tab, and then use the information collected from the `adc_config` structure in the ASFv3 example project to populate the configuration options, and then enable PA06 (AIN/6) for the ADC input, as shown in Figure 12,

FIGURE 12: CONFIGURE ADC PERIPHERAL (1 OF 2)

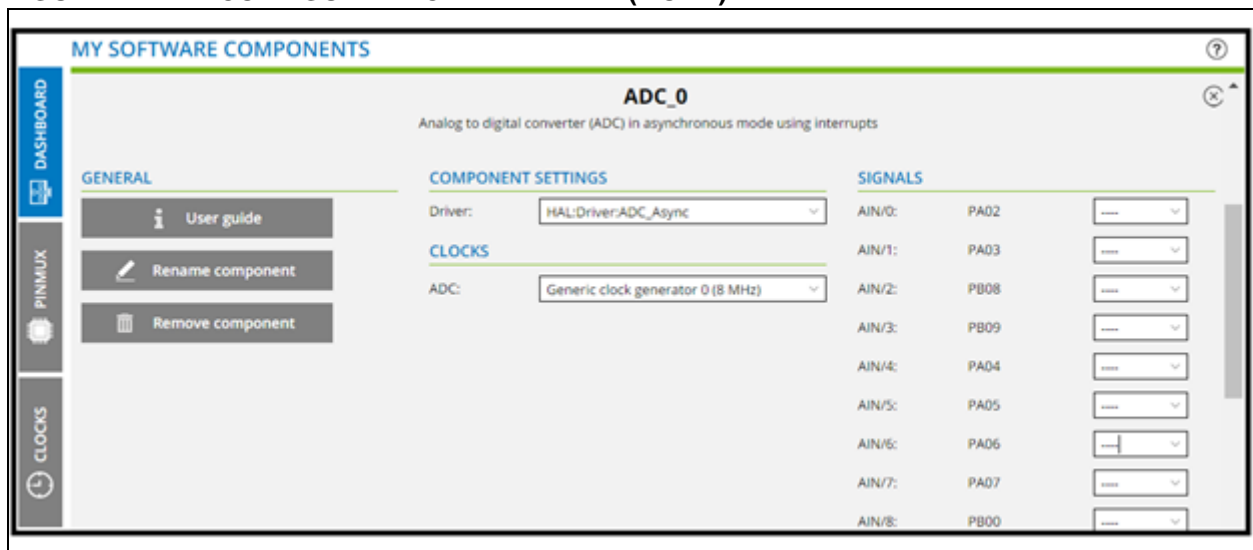
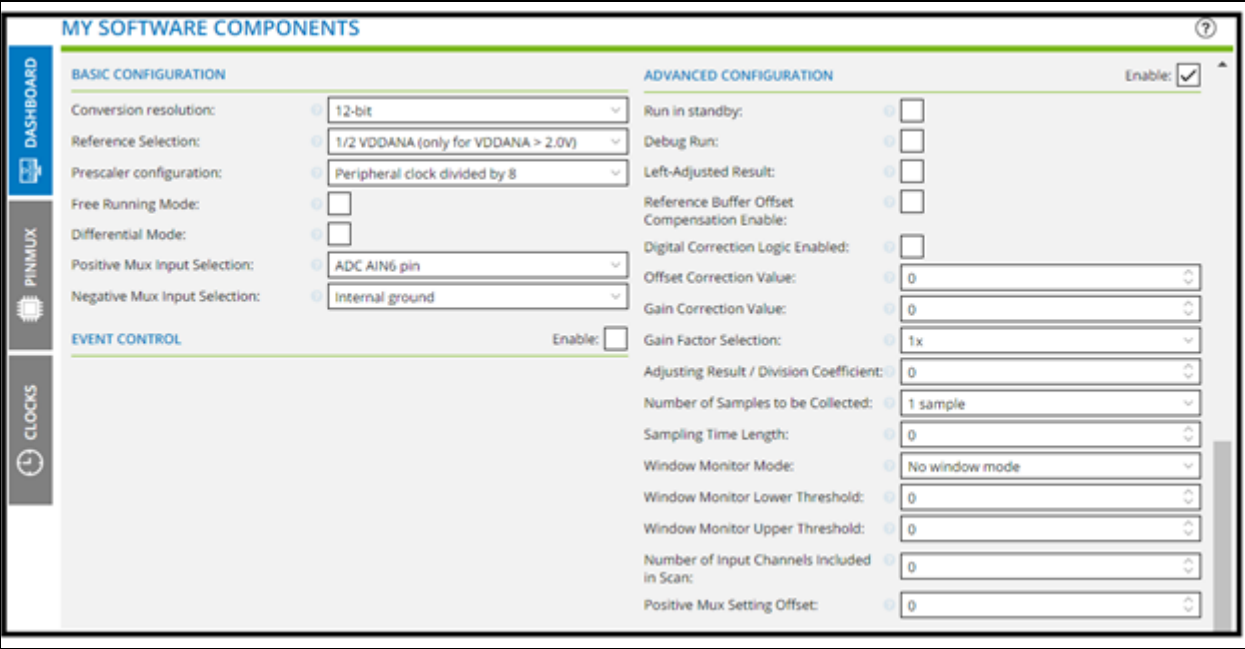


FIGURE 13: CONFIGURE ADC PERIPHERAL (2 OF 2)



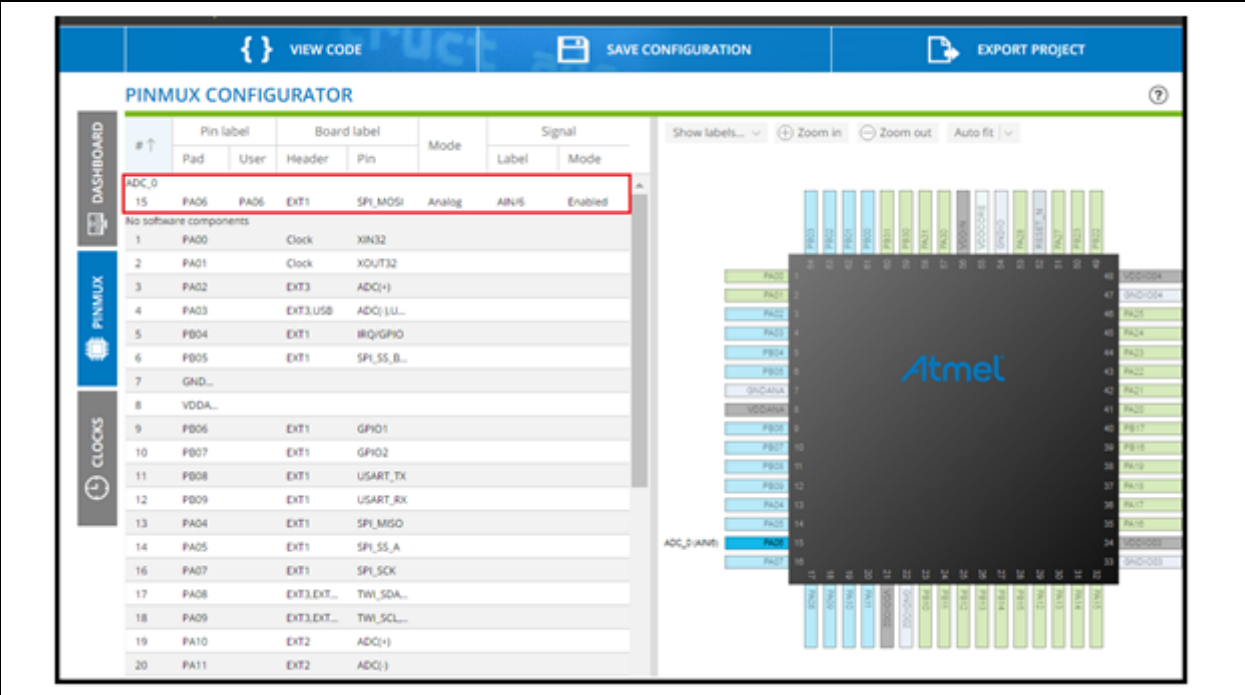
9. Configure the pinout.

When collecting the information from the ASFv3 project, the ADC will only use one pin in this project and that is configured in the ADC configuration on the Dashboard. However, GPIO or the naming of pins is done in the Pinmux Configurator, see Figure 14. By selecting the pin you would like to modify in the Pinmux Configurator, a user window will open near the bottom of the screen to set the parameters.

10. Configure the clock tree.

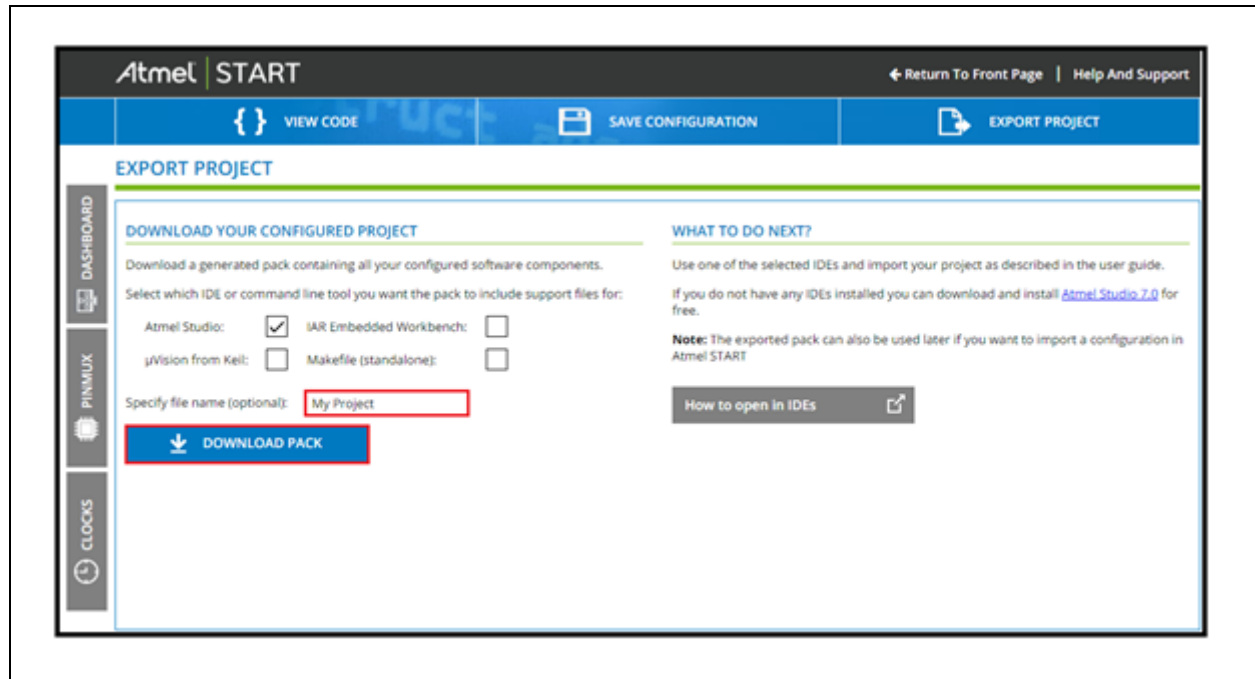
The Clock Tree can be configured in two places in the Start tool. The Dashboard, through the System Driver modules, and the Clock Configurator. The Clock Configurator provides a graphical representation of the clock tree and provides the same configuration capabilities as the Dashboard. The Clock Configurator will be used for this example.

FIGURE 14: ATMEL START - PIN MUX



11. Export the project by clicking the Export Project tab at the top right portion of the Atmel Start screen to export the project, as shown in Figure 15. Only Atmel Studio should be selected, as this is the only IDE intended on using this exported project. The output from Atmel Start will be a atzip file.

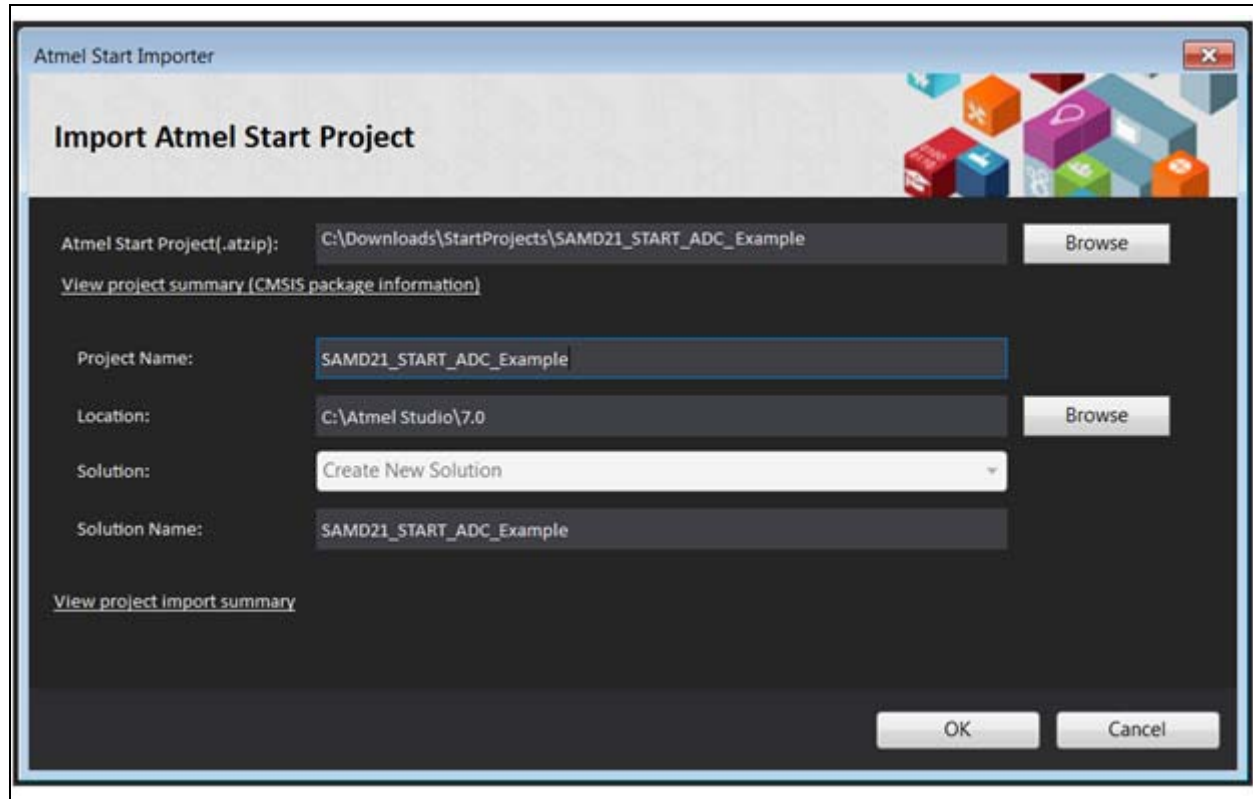
FIGURE 15: ATMEL START - EXPORT PROJECT



VERIFICATION OF SYSTEM CONFIGURATION

1. From Atmel Studio, import the atzip file exported from Atmel Start by selecting *File > Import > Atmel Start Project*.

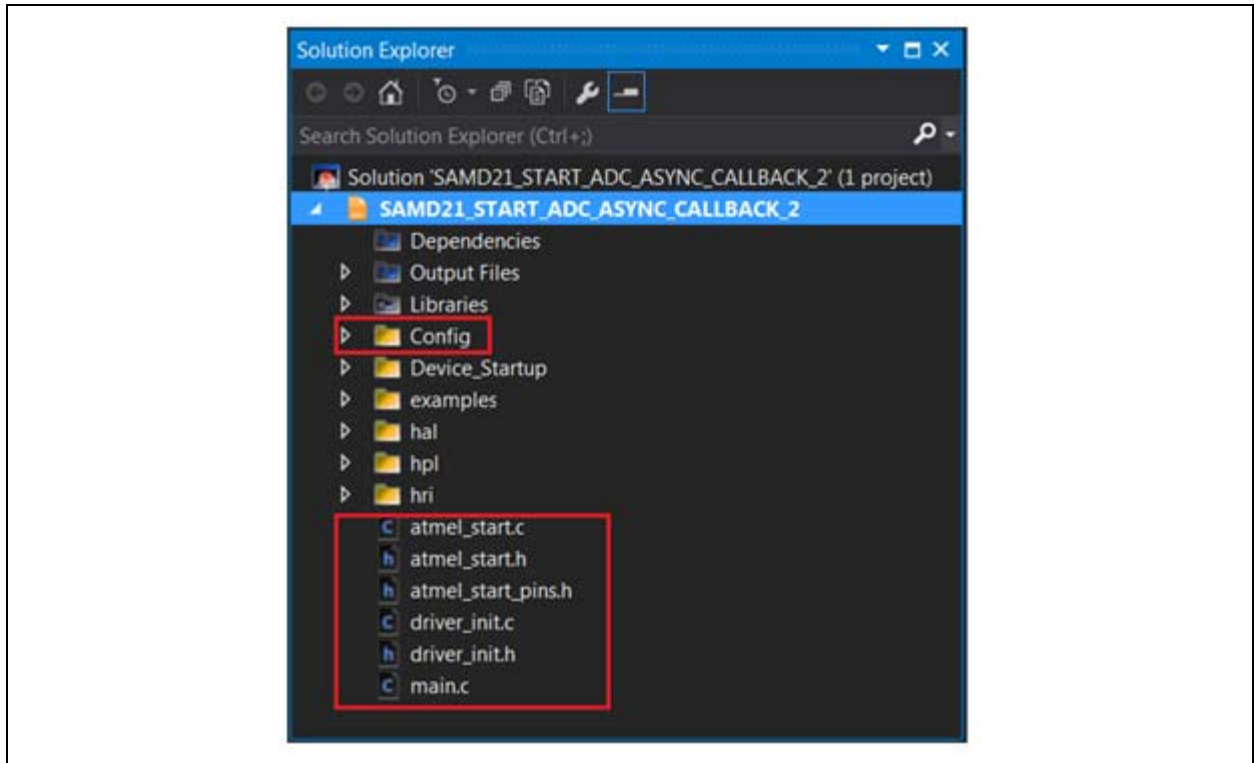
FIGURE 16: ATMEL START - IMPORT PROJECT



The parameters set in Start are conveniently located in the `Config` folder for the system initialization, `atmel_start_pins.h` for the pin assignments, and `driver_init.c/driver_init.h` for the peripheral initializations, see [Figure 17](#).

- `Config` - contains register settings from Start used for preprocessor device configuration
- `atmel_start.c` - Initializes MCU, drivers and middleware
- `atmel_start_pins.h` - holds the MCU pin assignments and naming from Start configuration
- `driver_init.c` - contains initialization functions for peripherals
- `main.c` - calls `atmel_start_init()` to initialize the system

FIGURE 17: ASFV4 FILE STRUCTURE



2. Build and run the project.

As the new project will not contain any application code, Start does output initialization code for the defined peripherals in `system_init()`. Run the project and stop after the initialization sequence to view the register settings in the I/O Window.

The I/O Window is available to open in Atmel Studio by clicking the I/O icon.



- Compare the register settings between ASFv3 and ASFv4 by opening the I/O window and selecting the Analog-to-Digital Converter and comparing the results to those collected from ASFv3 (see [Figure 18](#)).

FIGURE 18: ASFv3 AND ASFv4 ADC REGISTER COMPARISON

Register Name	Address	ASFv3 Value	ASFv4 Value
CTRLA	0x42004000	0x02	0x02
REFCTRL	0x42004001	0x02	0x02
AVGCTRL	0x42004002	0x00	0x00
SAMPCTRL	0x42004003	0x00	0x00
CTRLB	0x42004004	0x0100	0x0100
WINCTRL	0x42004008	0x00	0x00
SWTRIG	0x4200400C	0x00	0x00
INPUTCTRL	0x42004010	0x0F001806	0x0F001806
EVCTRL	0x42004014	0x00	0x00
INTENCLR	0x42004016	0x00	0x01
INTENSET	0x42004017	0x00	0x01
INTFLAG	0x42004018	0x08	0x08
STATUS	0x42004019	0x00	0x00
RESULT	0x4200401A	0x0000	0x0000
WINLT	0x4200401C	0x0000	0x0000
WINUT	0x42004020	0x0000	0x0000
GAINCORR	0x42004024	0x0000	0x0000
OFFSETC...	0x42004026	0x0000	0x0000
CALIB	0x42004028	0x0388	0x0000
DBGCTRL	0x4200402A	0x00	0x00

Two differences immediately become clear when the above comparison is made. The location at which the ADC interrupts are enabled and the calibration setting of the ADC.

In ASFv3, the interrupts are enabled within the ADC job function while ASFv4 enables interrupts at the time the callback function is assigned.

Additionally, ASFv4 ADC driver does not import the factory calibration settings for the ADC. This will have to be done by the user. The calibration provides bias and linearity settings for more precise ADC readings. For additional information refer to the **Section 9.3.2 “NVM Software Calibration Area Mapping”** of the SAM D21 Family Data Sheet ([DS40001882](#)). To import the calibration into the ASFv4 project refer to [Appendix A: “Application Code Examples”](#).

APPLICATION CONVERSION

1. Application description.

The ADC will be started after initialization and collect 128 12-bit samples and store the data in a buffer. When the samples have been collected, the ADC callback will be entered and a flag set indicating completion of sampling.

2. Compare API.

The use of the ADC module does not require any additional configuration other than that provided from the Start setup. ASFv3 is very similar in that a set of defaults are provided, but requires the developer to create the initialization function and modify the defaults directly.

The example program uses `adc_read_buffer_job` to fill a buffer with a defined number of samples. This functionality does not have a direct match in ASFv4. The closest to `adc_read_buffer_job` in ASFv4 would be `adc_async_start_conversion`. However, these two functions will produce very different results. In ASFv3, the `ADC_adc_interrupt_handler` is configured to complete a job and populate the assigned buffer with the number of requested samples. The ASFv4 ADC Handler is designed to take a more general approach and requires the developer to handle the additional sampling in the callback. To match the functionality of ASFv3, the ADC callback in ASFv4 will place ADC results into an array and start the next ADC transaction.

Refer to [Appendix A: “Application Code Examples”](#) for complete setup code.

3. Callback setup.

For a general application, the `example` folder in the main project tree contains code that will set up the peripherals enabled for the project. In this case, the `example` folder contains everything needed to enable the callback for the application.

The ADC callback is set up similarly in ASFv3 and ASFv4. The difference is in the way the ADC interrupt handler functions when an interrupt is signaled. Search `_adc_interrupt_handler` to view the differences between ASFv3 and ASFv4. ASFv4 provides less overhead and allows the developer to handle special functionality in the callback.

Refer to [Appendix A: “Application Code Examples”](#) for complete setup code.

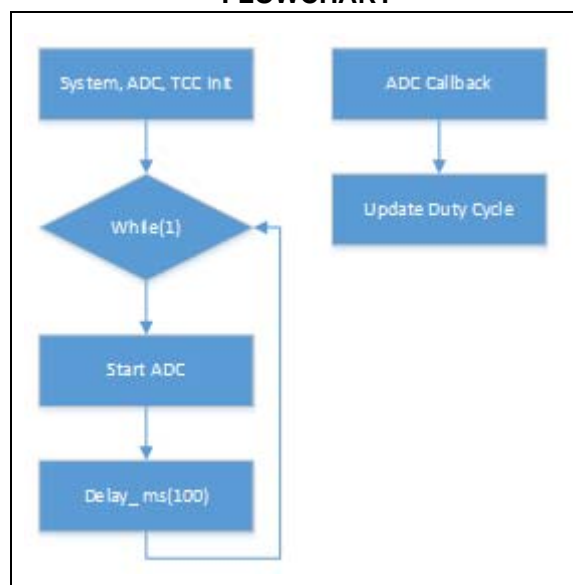
Project Expansion

Add the Timer Control module to the ASFv3 Project, as follows.

1. Application description.

The updated application will use the callback of the ADC to update the duty cycle of a PWM channel connected to a LED. The Duty Cycle will be calculated using the value read from the ADC. Additionally, rather than have a single job collecting a block of ADC values, the ADC will be started in a continuous loop to update the duty cycle of the PWM channel every 100 ms.

FIGURE 19: EXPANDED PROJECT FLOWCHART



2. Application components.

To achieve the new functionality of the project, the Delay services and the TCC0 Driver modules will have to be added from the ASF Wizard.

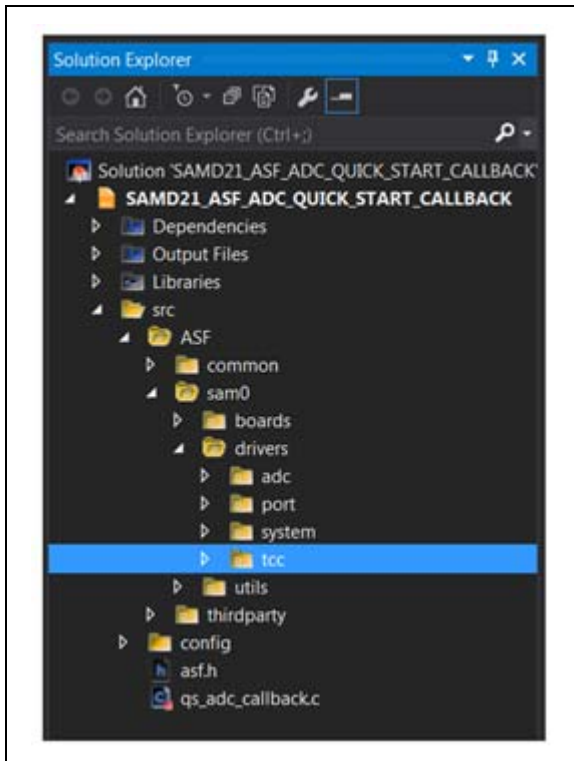
3. Open the ASFv3 project and add the Timer Control module.

In Atmel Studio, under Project, select the ASF Wizard. Once the window is open, make sure the project drop down menu is populated with the Example project to be modified.

In the Available Modules menu, select the TC - Timer Counter (driver), with the callback option selected, and the Delay Services module. Apply the added modules to the project by clicking **Apply** located at the bottom of the window. Both the TC driver and Delay Services will be displayed in the Select Modules menu.

Once the TCC has been added to the project by the ASF Wizard the `tcc` driver folder can be found in the project, as shown in [Figure 20](#) and the appropriate header files are added to `asf.h` for use in the application.

FIGURE 20: ADDED TCC0 DRIVER



SET UP THE ASFv3 TIMER CONFIGURATION

1. Set up the Timer configuration.

Since the project is built on the Xplained Pro, some definitions have been assigned to the on-board LED for use as a PWM output. The definitions are contained in the `samd21_xplained_pro.h` file of the project, see Figure 21.

Create a `configure_tcc` function, as shown in Figure 22, using the macros defined in `samd21_xplained_pro.h`. Create an instance of the `tcc_module` structure called `tcc_instance`. Notice one of the first lines in

the `configure_tcc` function is a function call to get a set of values for configuration of a default TCC module. ASFv3 contains a set of default settings for each peripheral. These defaults will be used as a basis for this project and modified only where needed.

2. Capture the TCC0 register settings.

Once the TCC module has been configured compile and run the code in Atmel Studio. Capture the register settings as was done previously for the ADC using the I/O window. These values will be used to compare to the output from Atmel Start.

FIGURE 21: XPLAINED PRO DEFINITIONS

```

116 #define LED0_GPIO          LED0_PIN
117 #define LED0               LED0_PIN
118
119 #define LED_0_PWM4CTRL_MODULE  TCC0
120 #define LED_0_PWM4CTRL_CHANNEL 0
121 #define LED_0_PWM4CTRL_OUTPUT  0
122 #define LED_0_PWM4CTRL_PIN     PIN_PB30E_TCC0_W00
123 #define LED_0_PWM4CTRL_MUX     MUX_PB30E_TCC0_W00
124 #define LED_0_PWM4CTRL_PINMUX  PINMUX_PB30E_TCC0_W00

```

FIGURE 22: XPLAINED PRO DEFINITIONS

```

struct tcc_module tcc_instance;

static void configure_tcc(void)
{
    struct tcc_config config_tcc;

    tcc_get_config_defaults(&config_tcc, LED_0_PWM4CTRL_MODULE);

    config_tcc.counter.clock_prescaler = TCC_CLOCK_PRESCALER_DIV8;
    config_tcc.counter.period = 0x7C;
    config_tcc.compare.wave_generation = TCC_WAVE_GENERATION_SINGLE_SLOPE_PWM;
    config_tcc.compare.match[LED_0_PWM4CTRL_CHANNEL] = 0x3E;

    config_tcc.pins.enable_wave_out_pin[LED_0_PWM4CTRL_OUTPUT] = true;
    config_tcc.pins.wave_out_pin[LED_0_PWM4CTRL_OUTPUT] = LED_0_PWM4CTRL_PIN;
    config_tcc.pins.wave_out_pin_mux[LED_0_PWM4CTRL_OUTPUT] = LED_0_PWM4CTRL_MUX;
    //config_tcc.double_buffering_enabled = false;

    tcc_init(&tcc_instance, LED_0_PWM4CTRL_MODULE, &config_tcc);

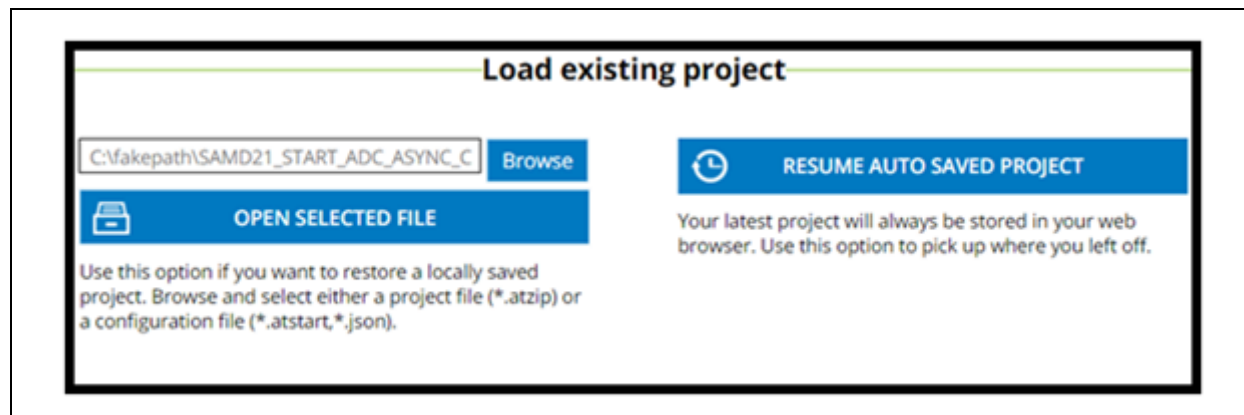
    tcc_enable(&tcc_instance);
}

```

ATMEL START - ADD TIMER

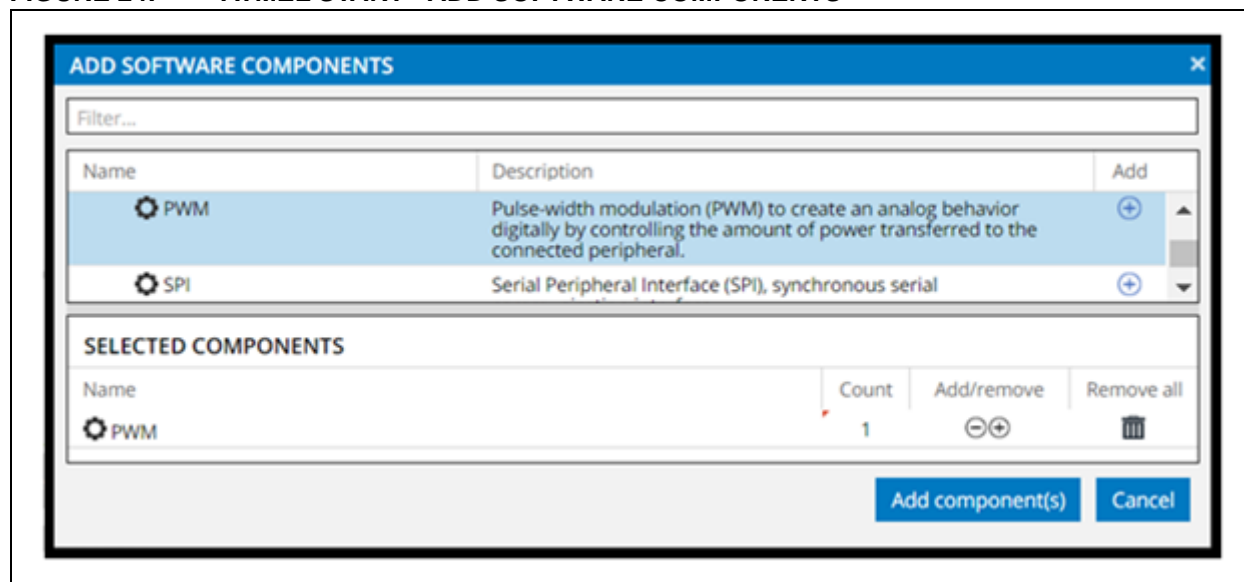
1. Import the Atmel Start project by browsing to the Atmel Start home page and select the option to load an existing project. Then, browse to the downloaded atzip file and click **Open Selected File**.

FIGURE 23: ATMEL START - LOAD PROJECT



2. Add the peripheral module to the project by clicking **Add Software Components**, and then selecting the PWM component in the window. Note that the PWM component is different from the TCC component, even though both would use the TCC module.

FIGURE 24: ATMEL START - ADD SOFTWARE COMPONENTS



- Configure the peripheral module by selecting the newly added PWM_0 module and using the values collected from the ASFv3 project to configure, as shown in Figure 25 and Figure 26.

FIGURE 25: ATMEL START - PWM CONFIGURATION (1 OF 2)

MY SOFTWARE COMPONENTS

PWM_0
PWM functionality using a TCC peripheral

GENERAL

- User guide
- Rename component
- Remove component

COMPONENT SETTINGS

Driver: HAL:Driver:PWM

Instance: TCC0

CLOCKS

TCC: Generic clock generator 0 (8 MHz)

SIGNALS

WO/0:	WO/1:	WO/2:	WO/3:	WO/4:	WO/5:	WO/6:	WO/7:
PB30	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
PWM out	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx

FIGURE 26: ATMEL START - PWM CONFIGURATION (2 OF 2)

MY SOFTWARE COMPONENTS

BASIC SETTINGS

TCC0 Prescaler: Divide by 8

TCC0 Period Value: 0xff

TCC0 Waveform Period Value (uS): 0xff

TCC0 Waveform Duty Value (0.1%): 0x7f

TCC0 Waveform Channel Select: 0x0

ADVANCED SETTINGS

Run in standby: ☐

TCC0 Prescaler and Counter Synchronization Selection: Reload or reset counter on next GCLK

TCC0 Waveform Generation Selection: Single-slope PWM

TCC0 Auto Lock: ☐

TCC0 Capture Channel 0 Enable: ☐

TCC0 Capture Channel 1 Enable: ☐

TCC0 Capture Channel 2 Enable: ☐

TCC0 Capture Channel 3 Enable: ☐

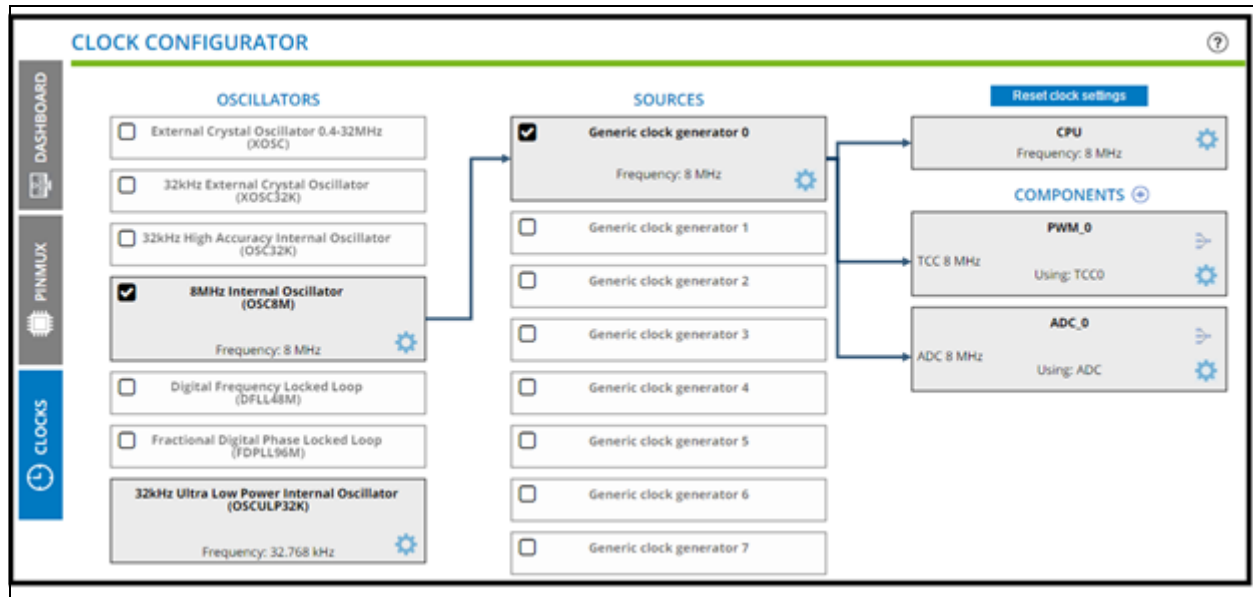
TCC0 Lock update: ☐

TCC0 Debug Running Mode: ☐

4. Configure the clock tree.

The initial version of this project had 8 MHz going to the Generic Clock Generator 0 (GCLK_0). Make sure both ADC_0 and PWM_0 are supplied a clock signal by GCLK_0. If the PWM_0 component is not sourced by GCLK_0, click the settings icon on the component block and set the TCC clock source to GCLK_0.

FIGURE 27: ATMEL START - TCC0 CLOCK CONFIGURATOR



VERIFICATION OF SYSTEM CONFIGURATION

1. Compare the register settings between ASFv3 and ASFv4 by importing and building the project as previously described.
2. Run the project until the TCC0 peripheral has been initialized and enabled. [Figure 28](#) displays the initialization differences between ASFv3 and ASFv4. From the comparison in [Figure 28](#), the initialization of the TCC0 module is configured in ASFv4.

FIGURE 28: ATMEL START - TCC0 REGISTER COMPARISON

Name	Address	Value	Bits
CTRLA	0x42002000	0x00000302	
CTRLBCLR	0x42002004	0x00	
CTRLBSET	0x42002005	0x00	
SYNCBUSY	0x42002008	0x00000000	
FCTRLA	0x4200200C	0x00000000	
FCTRLB	0x42002010	0x00000000	
WEXCTRL	0x42002014	0x00000000	
DRVCTRL	0x42002018	0x00000000	
DBGCTRL	0x4200201E	0x00	
EVCTRL	0x42002020	0x00000000	
INTENCLR	0x42002024	0x00000000	
INTENSET	0x42002028	0x00000000	
INTFLAG	0x4200202C	0x00000014	
STATUS	0x42002030	0x01000000	
COUNT	0x42002034	0x00000000	
PATT	0x42002038	0x0000	
WAVE	0x4200203C	0x00000002	
PER	0x42002040	0x000000FE	
CC0	0x42002044	0x00000020	
CC1	0x42002048	0x00000000	
CC2	0x4200204C	0x00000000	
CC3	0x42002050	0x00000000	
PATTB	0x42002064	0x0000	
WAVEB	0x42002068	0x00000000	
PERB	0x4200206C	0x00FFFFFF	
CCB0	0x42002070	0x00000000	
CCB1	0x42002074	0x00000000	
CCB2	0x42002078	0x00000000	
CCB3	0x4200207C	0x00000000	

APPLICATION UPDATE

1. Application description.

To provide the input for the PWM Duty Cycle calculation, only the algorithm in the ADC callback needs to be implemented. The ADC is set up to collect a single 12-bit sample, and then trigger the callback. In the ADC callback, the ADC RESULT register will be read directly and shifted to provide an 8-bit value. This 8-bit value will be written to the Duty Cycle of the PWM. The PWM was configured to have an 8-bit period so the Duty Cycle will scale appropriately.

2. Compare API.

The use of the TCC0 module does not require any additional configuration other than that provided from the Start setup. ASFv3 is very similar in that a set of defaults are provided, but requires the developer to create the initialization function and modify the defaults directly.

ASFv4 has a more use case driven model throughout the entire framework. This can be seen from when the TCC component was added in ASFv3 and a PWM component was added in Start. The API for the PWM component in ASFv4 displays this as well in `pwm_set_parameters`. ASFv3 uses `tcc_set_compare_value` to achieve the same function. The parameters passed in these functions do not differ dramatically, but drilling down into the functions a much thinner, use case-driven approach is achieved in ASFv4.

3. Callback Setup.

The TCC0 module in this project is not using a callback, but the Duty Cycle is changed in the ADC callback function.

CONCLUSION

ASFv4 is a new framework containing improvements in code size and driver efficiency with an application driven use case approach. While the architecture of ASFv4 is different to that of ASFv3, naming conventions and fundamental aspects pertaining to the devices have remained.

There is not a canned solution to approach a conversion from ASFv3 to ASFv4. The migration will have to occur from the ground up. While this document has only covered a couple of the functional differences, a basic principal is provided to achieve a functional migration.

APPENDIX A: APPLICATION CODE EXAMPLES

EXAMPLE 1: ASFv3 PROJECT EXPANSION - ADC AND PWM

```
#define ADC_SAMPLES 1
uint16_t adc_result_buffer[ADC_SAMPLES];

struct adc_module adc_instance;
struct tcc_module tcc_instance;

volatile bool adc_read_done = false;

void adc_complete_callback(struct adc_module *const module)
{
    uint32_t duty;

    duty = (adc_result_buffer[0] >> 4) & 0xFFFF;

    tcc_set_compare_value(&tcc_instance, (enum tcc_match_capture_channel)(CONF_PWM_CHANNEL),
duty);
}

void configure_adc(void)
{
    struct adc_config config_adc;

    adc_get_config_defaults(&config_adc);

    config_adc.clock_prescaler = ADC_CLOCK_PRESCALER_DIV8;
    config_adc.reference       = ADC_REFERENCE_INTVCC1;
    config_adc.positive_input  = ADC_POSITIVE_INPUT_PIN18;
    config_adc.resolution      = ADC_RESOLUTION_12BIT;

    adc_init(&adc_instance, ADC, &config_adc);
    adc_enable(&adc_instance);
}

static void configure_tcc(void)
{
    struct tcc_config config_tcc;

    tcc_get_config_defaults(&config_tcc, LED_0_PWM4CTRL_MODULE);

    config_tcc.counter.clock_prescaler = TCC_CLOCK_PRESCALER_DIV8;
    config_tcc.counter.period = 0xFE;
    config_tcc.compare.wave_generation = TCC_WAVE_GENERATION_SINGLE_SLOPE_PWM;
    config_tcc.compare.match[LED_0_PWM4CTRL_CHANNEL] = 0x7F;

    config_tcc.pins.enable_wave_out_pin[LED_0_PWM4CTRL_OUTPUT] = true;
    config_tcc.pins.wave_out_pin[LED_0_PWM4CTRL_OUTPUT]       = LED_0_PWM4CTRL_PIN;
    config_tcc.pins.wave_out_pin_mux[LED_0_PWM4CTRL_OUTPUT]    = LED_0_PWM4CTRL_MUX;

    tcc_init(&tcc_instance, LED_0_PWM4CTRL_MODULE, &config_tcc);

    tcc_enable(&tcc_instance);
}

void configure_adc_callbacks(void)
{
    adc_register_callback(&adc_instance, adc_complete_callback, ADC_CALLBACK_READ_BUFFER);
    adc_enable_callback(&adc_instance, ADC_CALLBACK_READ_BUFFER);
}

int main(void)
{
    system_init();
    delay_init();

    configure_adc();
    configure_adc_callbacks();
    configure_tcc();

    system_interrupt_enable_global();

    while (1) {
        adc_read_buffer_job(&adc_instance, adc_result_buffer, ADC_SAMPLES);
        delay_cycles_ms(100);
    }
}
```


EXAMPLE 2: ASFv3 MIGRATION - ADC ONLY

```
void adc_complete_callback(const struct adc_async_descriptor *const descr, const uint8_t
channel)
{
    static uint8_t i = 0;

    if (i < ADC_SAMPLES)
    {
        adc_result_buffer[i++] = ADC->RESULT.reg;
        adc_async_start_conversion(&ADC_0);
    }
    else
    {
        adc_read_done = true;
    }
}

int main(void)
{
    system_init();

    adc_async_register_callback(&ADC_0, 0, ADC_ASYNC_CONVERT_CB, adc_complete_callback);
    adc_async_enable_channel(&ADC_0, 0);

    adc_async_start_conversion(&ADC_0);

    while (adc_read_done == false)
    {
        /* Wait for asynchronous ADC reads to complete */
    }

    while(1)
    {
        asm("nop");
    }
}
```

EXAMPLE 3: PROJECT EXPANSION - ADC AND PWM

```
#define PWM_PERIOD 254
static uint32_t pwm_duty;
static uint16_t adc_value;

static void adc_cb(const struct adc_async_descriptor *const descr, const uint8_t channel)
{
    adc_value = ADC->RESULT.reg;

    pwm_duty = (adc_value >> 4) & 0xFF;

    pwm_set_parameters(&PWM_0, PWM_PERIOD, pwm_duty);
}

int main(void)
{
    atmel_start_init();

    adc_async_register_callback(&ADC_0, 0, ADC_ASYNC_CONVERT_CB, adc_cb);

    ADC->CALIB.reg = ADC_CALIB_BIAS_CAL((*(uint32_t *)ADC_FUSES_BIASCAL_ADDR >>
ADC_FUSES_BIASCAL_Pos)) |
    ADC_CALIB_LINEARITY_CAL((*(uint64_t *)ADC_FUSES_LINEARITY_0_ADDR >>
ADC_FUSES_LINEARITY_0_Pos));

    adc_async_enable_channel(&ADC_0, 0);
    pwm_enable(&PWM_0);

    while(1)
    {
        adc_async_start_conversion(&ADC_0);
        delay_ms(100);
    }
}
```

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
== ISO/TS 16949 ==

Trademarks

The Microchip name and logo, the Microchip logo, AnyRate, AVR, AVR logo, AVR Freaks, BeaconThings, BitCloud, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, Helder, JukeBlox, KEELOQ, KEELOQ logo, Klear, LANCheck, LINK MD, maXStylus, maXTouch, MediaLB, megaAVR, MOST, MOST logo, MPLAB, OptoLyzer, PIC, picoPower, PICSTART, PIC32 logo, Prochip Designer, QTouch, RightTouch, SAM-BA, SpyNIC, SST, SST Logo, SuperFlash, tinyAVR, UNI/O, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

ClockWorks, The Embedded Control Solutions Company, EtherSynch, Hyper Speed Control, HyperLight Load, IntellIMOS, mTouch, Precision Edge, and Quiet-Wire are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, BodyCom, chipKIT, chipKIT logo, CodeGuard, CryptoAuthentication, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, EtherGREEN, In-Circuit Serial Programming, ICSP, Inter-Chip Connectivity, JitterBlocker, KlearNet, KlearNet logo, Mindi, MiWi, motorBench, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICKit, PICtail, PureSilicon, QMatrix, RightTouch logo, REAL ICE, Ripple Blocker, SAM-ICE, Serial Quad I/O, SMART-I.S., SQI, SuperSwitcher, SuperSwitcher II, Total Endurance, TSHARC, USBCheck, VariSense, ViewSpan, WiperLock, Wireless DNA, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

Silicon Storage Technology is a registered trademark of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

© 2017, Microchip Technology Incorporated, All Rights Reserved.
ISBN: 978-1-5224-1864-1

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
<http://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

ASIA/PACIFIC

Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon

Hong Kong
Tel: 852-2943-5100
Fax: 852-2401-3431

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

China - Chongqing
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

China - Dongguan
Tel: 86-769-8702-9880

China - Guangzhou
Tel: 86-20-8755-8029

China - Hangzhou
Tel: 86-571-8792-8115
Fax: 86-571-8792-8116

China - Hong Kong SAR
Tel: 852-2943-5100
Fax: 852-2401-3431

China - Nanjing
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

China - Qingdao
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

China - Shanghai
Tel: 86-21-3326-8000
Fax: 86-21-3326-8021

China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen
Tel: 86-755-8864-2200
Fax: 86-755-8203-1760

China - Wuhan
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

China - Xian
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

ASIA/PACIFIC

China - Xiamen
Tel: 86-592-2388138
Fax: 86-592-2388130

China - Zhuhai
Tel: 86-756-3210040
Fax: 86-756-3210049

India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

India - Pune
Tel: 91-20-3019-1500

Japan - Osaka
Tel: 81-6-6152-7160
Fax: 81-6-6152-9310

Japan - Tokyo
Tel: 81-3-6880-3770
Fax: 81-3-6880-3771

Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Hsin Chu
Tel: 886-3-5778-366
Fax: 886-3-5770-955

Taiwan - Kaohsiung
Tel: 886-7-213-7830

Taiwan - Taipei
Tel: 886-2-2508-8600
Fax: 886-2-2508-0102

Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
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

France - Saint Cloud
Tel: 33-1-30-60-70-00

Germany - Garching
Tel: 49-8931-9700

Germany - Haan
Tel: 49-2129-3766400

Germany - Heilbronn
Tel: 49-7131-67-3636

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: 47-7289-7561

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