

ARM Development Primer



Learning to Drive the ATSAME54P20A

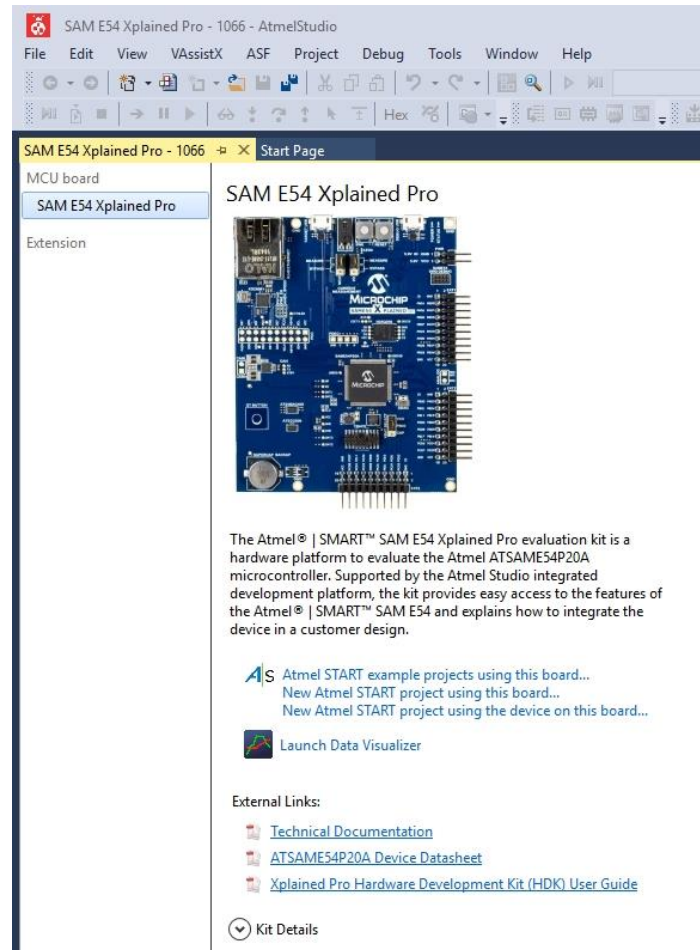
November 16, 2017

FRED EADY

ARM Development Primer

AGENDA

- Crawling Before We Walk
- **USART the ATMEL START Way**
- Day 4's Done



ARM Development Primer

Crawling Before We Walk – First Program



Atmel | START

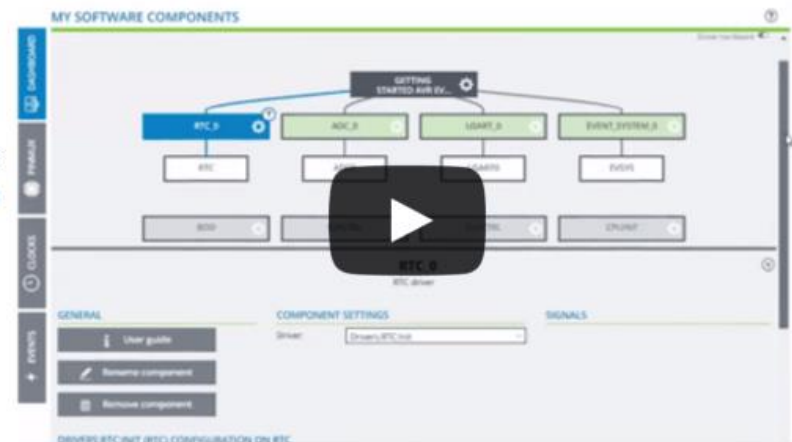
This tool will help you select and configure software components and tailor your embedded application in a usable and optimized manner.

Getting started

To get started you can either create a new project from scratch or open an existing example. In both cases you can configure your software components and device settings such as clocks and pin layout. When you are done, you can export your project and open it using your favourite IDE for further development.

For more information on how to use Atmel START, read the [Getting Started guide](#) or watch our [video tutorials](#).



ARM Development Primer

Crawling Before We Walk – First Program

Atmel | START

← Return To Front Page | 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.

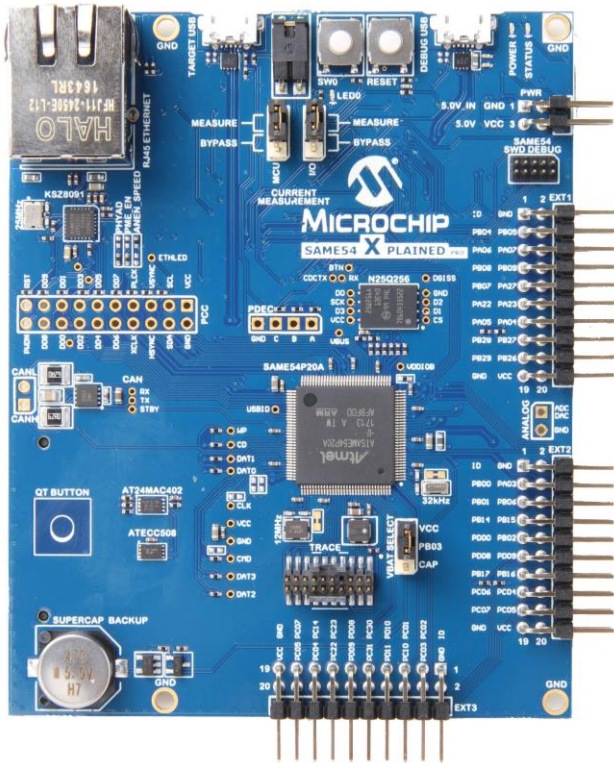
FILTERS

HARDWARE

SEARCH FOR SOFTWARE

Find software...

MIDDLEWARE



RESULTS

Filter on device...

Show all Show only boards Show only devices

Name	Architecture	Package	Pins	Flash	SRAM	
■ ATtiny817 Xplained Mini						🔗
■ XMEGA A1U Xplained Pro ¹						🔗
■ SAM B11 Xplained Pro						🔗
■ SAMC21 interface with ATSAMBLDCHV-STK						🔗
■ SAM C21 Xplained Pro						🔗
■ SAM C21N Xplained Pro						🔗
■ SAM D10 Xplained Mini						🔗
■ SAM D11 Xplained Pro						🔗
■ SAM D20 Xplained Pro						🔗
■ SAMD21E16L interface with ATSAMBLDCHV-STK						🔗
■ SAM W25 Xplained Pro						🔗
■ SAMD21J18A interface with ATSAMBLDCHV-STK						🔗
■ SAM D21 Xplained Pro						🔗
■ SAM DA1 Xplained Pro						🔗
■ SAM L21 Xplained Pro						🔗
■ SAM L22 Xplained Pro						🔗
■ SAM R21 Xplained Pro						🔗
■ SAM R30 Xplained Pro						🔗
■ SAM E54 Xplained Pro						🔗
■ SAM G53 Xplained Pro						🔗
■ SAM G55 Xplained Pro						🔗
■ SAM E70 Xplained						🔗
■ SAM V71 Xplained Ultra						🔗

27 of 875 boards and devices

¹ Driver code in Beta

CREATE NEW PROJECT



ARM Development Primer

Crawling Before We Walk – First Program

Atmel | START ATSAME54P20A

← Return To Front Page | Help And Support

{ } VIEW CODE

SAVE CONFIGURATION

EXPORT PROJECT

CLOCK CONFIGURATOR

Zoom in Zoom out Reset

OSCILLATORS

- External Crystal Oscillator 8-48MHz
- External Crystal Oscillator 8-48MHz (XOSC1)
12 MHz
- 32kHz External Crystal Oscillator
- Digital Frequency Locked Loop
- Digital Phase Locked Loop
- Digital Phase Locked Loop
- 32kHz Ultra Low Power Internal Oscillator (OSCULP32K)
32.768 kHz

SOURCES

- Generic clock generator 0
12 MHz
- Generic clock generator 1
- Generic clock generator 2
- Generic clock generator 3
- Generic clock generator 4
- Generic clock generator 5
- Generic clock generator 6
- Generic clock generator 7

Reset clock settings

CPU
12 MHz

COMPONENTS +



ARM Development Primer

Crawling Before We Walk – First Program

5.3.3 LED

There is one yellow LED available on the SAM E54 Xplained Pro board that can be turned ON and OFF. The LED can be activated by driving the connected I/O line to GND.

Table 5-15. LED Connection

SAM E54 pin	Function	Shared functionality
PC18	Yellow LED0	PDEC and EDBG GPIO2



ARM Development Primer

Crawling Before We Walk – First Program

Atmel | START ATSAME54P20A Return To Front Page | Help And Support

VIEW CODE **SAVE CONFIGURATION** **EXPORT PROJECT**

PINMUX CONFIGURATOR

# ↑	Pin label		Board label		Mode	Signal
	Pad	User	Header	Pin		Label
PORT						
72	PC18	LEDYEL			Advanced	P/72
No software components						
1	PA00		Clock	XIN32		
2	PA01		Clock	XOUT32		
3	PC00					
4	PC01		EXT3	GPIO		
6	VDDA...					
7	PC02		EXT3	ADC(+)		
8	PC03		EXT3	ADC(-)		
9	PA02					
10	PA03		EXT2	ADC(-)		
11	PB04		EXT1	ADC(+)		
12	PB05		EXT1	ADC(-)		
13	PD00		EXT2	IRQ/GPIO		
14	GND...					
15	VDDA...					
16	PD01		DGI SPI	DGI_SS		
17	PB06		EXT2	GPIO		
18	PB07		EXT1	IRQ/GPIO		
19	PB08		EXT1	PWM(+)		
20	PB09		EXT1	PWM(-)		

Pin 72 (PC18) is used as P/72 with PORT.
Tip: Use *ctrl* or *shift* to select more than one pin.

User label:
Pin mode:

```
PREVIEW - ATMEL_START_PINS.H
config
examples
hal
hpl
hri
atmel_start.c
atmel_start.h
atmel_start_pins.h
driver_init.c
driver_init.h
main.c
1 /*
2 * Code generated from Atmel Start.
3 *
4 * This file will be overwritten when reconfiguring your Atmel Start project.
5 * Please copy examples or other code you want to keep to a separate file
6 * to avoid losing it when reconfiguring.
7 */
8 #ifndef ATMEL_START_PINS_H_INCLUDED
9 #define ATMEL_START_PINS_H_INCLUDED
10
11 #include <hal_gpio.h>
12
13 // SAME54 has 14 pin functions
14
15 #define GPIO_PIN_FUNCTION_A 0
16 #define GPIO_PIN_FUNCTION_B 1
17 #define GPIO_PIN_FUNCTION_C 2
18 #define GPIO_PIN_FUNCTION_D 3
19 #define GPIO_PIN_FUNCTION_E 4
20 #define GPIO_PIN_FUNCTION_F 5
21 #define GPIO_PIN_FUNCTION_G 6
22 #define GPIO_PIN_FUNCTION_H 7
23 #define GPIO_PIN_FUNCTION_I 8
24 #define GPIO_PIN_FUNCTION_J 9
25 #define GPIO_PIN_FUNCTION_K 10
26 #define GPIO_PIN_FUNCTION_L 11
27 #define GPIO_PIN_FUNCTION_M 12
28 #define GPIO_PIN_FUNCTION_N 13
29
30 #define LEDYEL GPIO(GPIO_PORTC, 18)
31
32 #endif // ATMEL_START_PINS_H_INCLUDED
```



ARM Development Primer

Crawling Before We Walk – First Program

Atmel | START ATSAME54P20A

VIEW CODE

SAVE CONFIGURATION

PINMUX CONFIGURATOR

#	Pin label		Board label		Mode	Signal
	Pad	User	Header	Pin		Label
PORT						
72	PC18	LEDYEL			Advanced	P/72
No software components						
1	PA00		Clock	XIN32		
2	PA01		Clock	XOUT32		
3	PC00					
4	PC01		EXT3	GPIO		
6	VDDA...					
7	PC02		EXT3	ADC(+)		
8	PC03		EXT3	ADC(-)		
9	PA02					
10	PA03		EXT2	ADC(-)		
11	PB04		EXT1	ADC(+)		
12	PB05		EXT1	ADC(-)		
13	PD00		EXT2	IRQ/GPIO		
14	GND...					
15	VDDA...					
16	PD01	DGI SPI		DGI_SS		
17	PB06		EXT2	GPIO		
18	PB07		EXT1	IRQ/GPIO		
19	PB08		EXT1	PWM(+)		
20	PB09		EXT1	PWM(-)		

PREVIEW - DRIVER_INIT.C

- config
- examples
- hal
- hpl
- hri
- atmel_start.c
- atmel_start.h
- atmel_start_pins.h
- driver_init.c
- driver_init.h
- main.c

```
17 void main(void)
18 {
19     // GPIO on PC18
20     gpio_set_pin_direction(LEDYEL,
21                             // <y> Pin direction
22                             // <id> pad_direction
23                             // <GPIO_DIRECTION_OFF"> Off
24                             // <GPIO_DIRECTION_IN"> In
25                             // <GPIO_DIRECTION_OUT"> Out
26                             GPIO_DIRECTION_OUT);
27
28     gpio_set_pin_level(LEDYEL,
29                       // <y> Initial level
30                       // <id> pad_initial_level
31                       // <false"> Low
32                       // <true"> High
33                       false);
34
35     gpio_set_pin_pull_mode(LEDYEL,
36                           // <y> Pull configuration
37                           // <id> pad_pull_config
38                           // <GPIO_PULL_OFF"> Off
39                           // <GPIO_PULL_UP"> Pull-up
40                           // <GPIO_PULL_DOWN"> Pull-down
41                           GPIO_PULL_OFF);
42
43     gpio_set_pin_function(LEDYEL,
44                          // <y> Pin function
45                          // <id> pad_function
46                          // <i> Auto : use driver pinmux if signal is imported
47                          // <GPIO_PIN_FUNCTION_OFF"> Auto
48                          // <GPIO_PIN_FUNCTION_OFF"> Off
49                          // <GPIO_PIN_FUNCTION_A"> A
50                          // <GPIO_PIN_FUNCTION_B"> B
51                          // <GPIO_PIN_FUNCTION_C"> C
52                          // <GPIO_PIN_FUNCTION_D"> D
53                          // <GPIO_PIN_FUNCTION_E"> E
54                          // <GPIO_PIN_FUNCTION_F"> F
55                          // <GPIO_PIN_FUNCTION_G"> G
56                          // <GPIO_PIN_FUNCTION_H"> H
57                          // <GPIO_PIN_FUNCTION_I"> I
58                          // <GPIO_PIN_FUNCTION_J"> J
59                          // <GPIO_PIN_FUNCTION_K"> K
60                          // <GPIO_PIN_FUNCTION_L"> L
61                          // <GPIO_PIN_FUNCTION_M"> M
62                          // <GPIO_PIN_FUNCTION_N"> N
63                          GPIO_PIN_FUNCTION_OFF);
64 }
```

Pin 72 (PC18) is used as P/72 with PORT.

Tip: Use *ctrl* or *shift* to select more than one pin.

User label:	<input type="text" value="LEDYEL"/>	Pin direction:	<input type="text" value="Out"/>
Pin mode:	<input type="text" value="Advanced"/>	Initial level:	<input type="text" value="Low"/>
		Pull configuration:	<input type="text" value="Off"/>
		Pin function:	<input type="text" value="Auto"/>



ARM Development Primer

Crawling Before We Walk – First Program

Atmel | START ATSAME54P20A

[← Return To Front Page](#) | [Help And Support](#)

 VIEW CODE

 SAVE CONFIGURATION

 EXPORT PROJECT

EXPORT PROJECT

DOWNLOAD YOUR CONFIGURED PROJECT

Download a generated pack containing all your configured software components.

Select which IDE or command line tool you want the pack to include support files for:

Atmel Studio: IAR Embedded Workbench:

µVision from Keil: Makefile (standalone):

Specify file name (optional):

 DOWNLOAD PACK

WHAT TO DO NEXT?

Use one of the selected IDEs and import your project as described in the user guide.

If you do not have any IDEs installed you can download and install [Atmel Studio 7.0](#) for free.

Note: The exported pack can also be used later if you want to import a configuration in Atmel START

How to open in IDEs



DASHBOARD

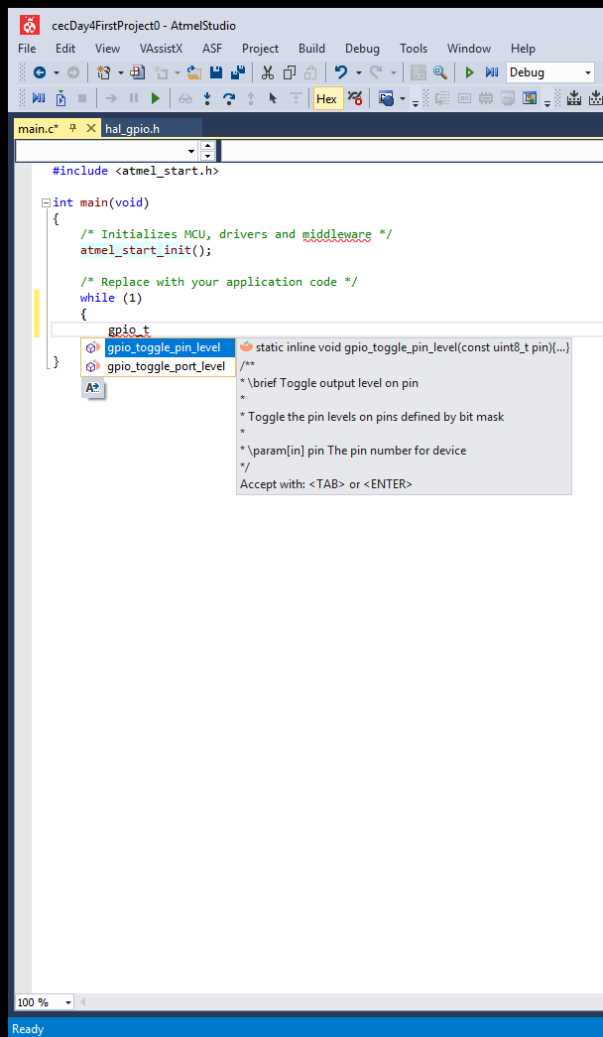
PINMUX

CLOCKS



ARM Development Primer

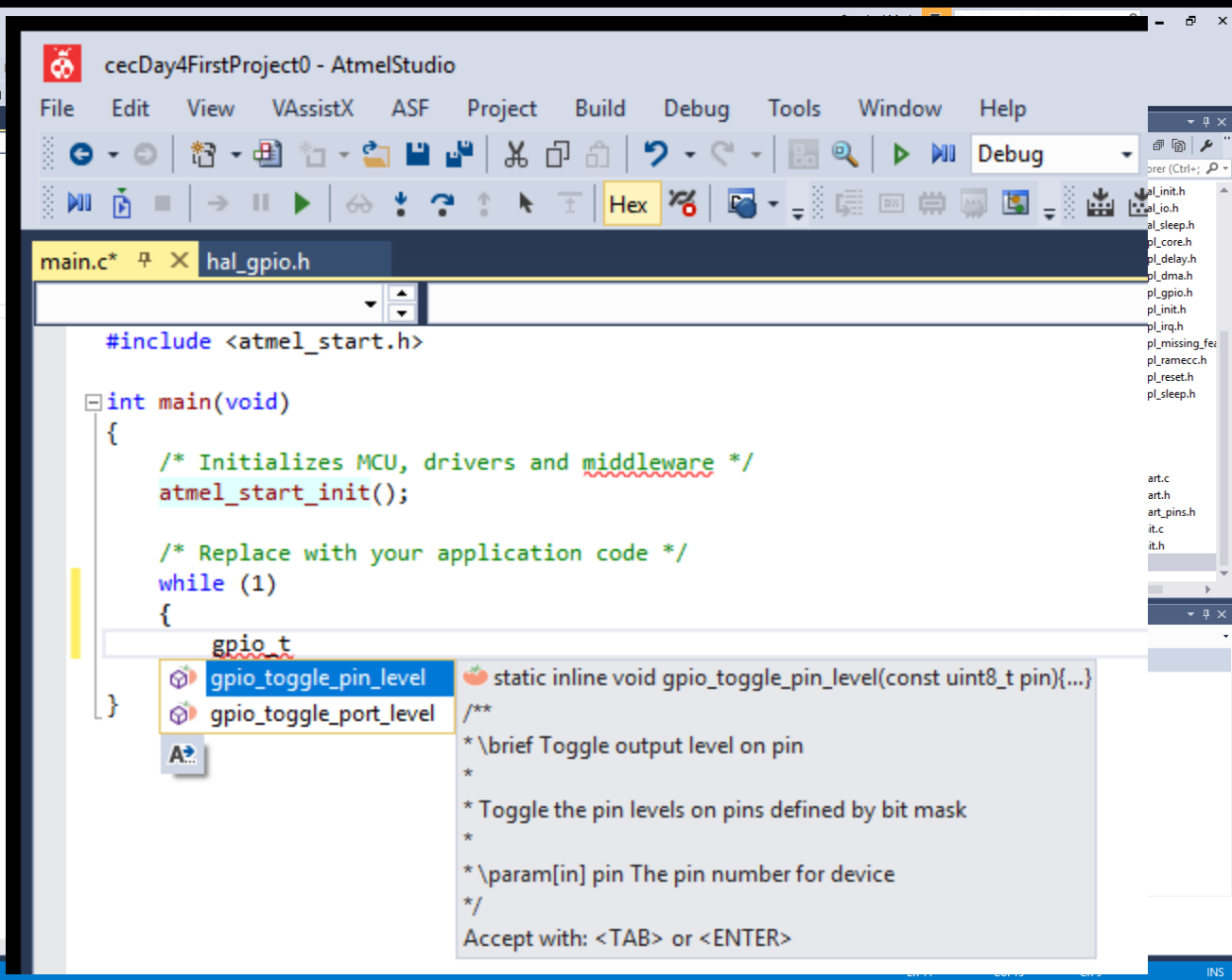
Crawling Before We Walk – First Program



```
#include <atmel_start.h>

int main(void)
{
    /* Initializes MCU, drivers and middleware */
    atmel_start_init();

    /* Replace with your application code */
    while (1)
    {
        gpio_t
        gpio_toggle_pin_level
        gpio_toggle_port_level /**
        * \brief Toggle output level on pin
        *
        * Toggle the pin levels on pins defined by bit mask
        *
        * \param[in] pin The pin number for device
        */
        Accept with: <TAB> or <ENTER>
    }
}
```



```
#include <atmel_start.h>

int main(void)
{
    /* Initializes MCU, drivers and middleware */
    atmel_start_init();

    /* Replace with your application code */
    while (1)
    {
        gpio_t
        gpio_toggle_pin_level
        gpio_toggle_port_level /**
        * \brief Toggle output level on pin
        *
        * Toggle the pin levels on pins defined by bit mask
        *
        * \param[in] pin The pin number for device
        */
        Accept with: <TAB> or <ENTER>
    }
}
```



ARM Development Primer

Crawling Before We Walk – First Program

The screenshot displays the Atmel Studio IDE interface for a project named 'cecDay4FirstProject0'. The main editor window shows the following C code:

```
#include <atmel_start.h>

int main(void)
{
    /* Initializes MCU, drivers and middleware */
    atmel_start_init();

    /* Replace with your application code */
    while (1)
    {
        gpio_toggle_pin_level(LEDYEL);
        delay
    }
}
```

A tooltip for the `delay` function is shown, containing the following information:

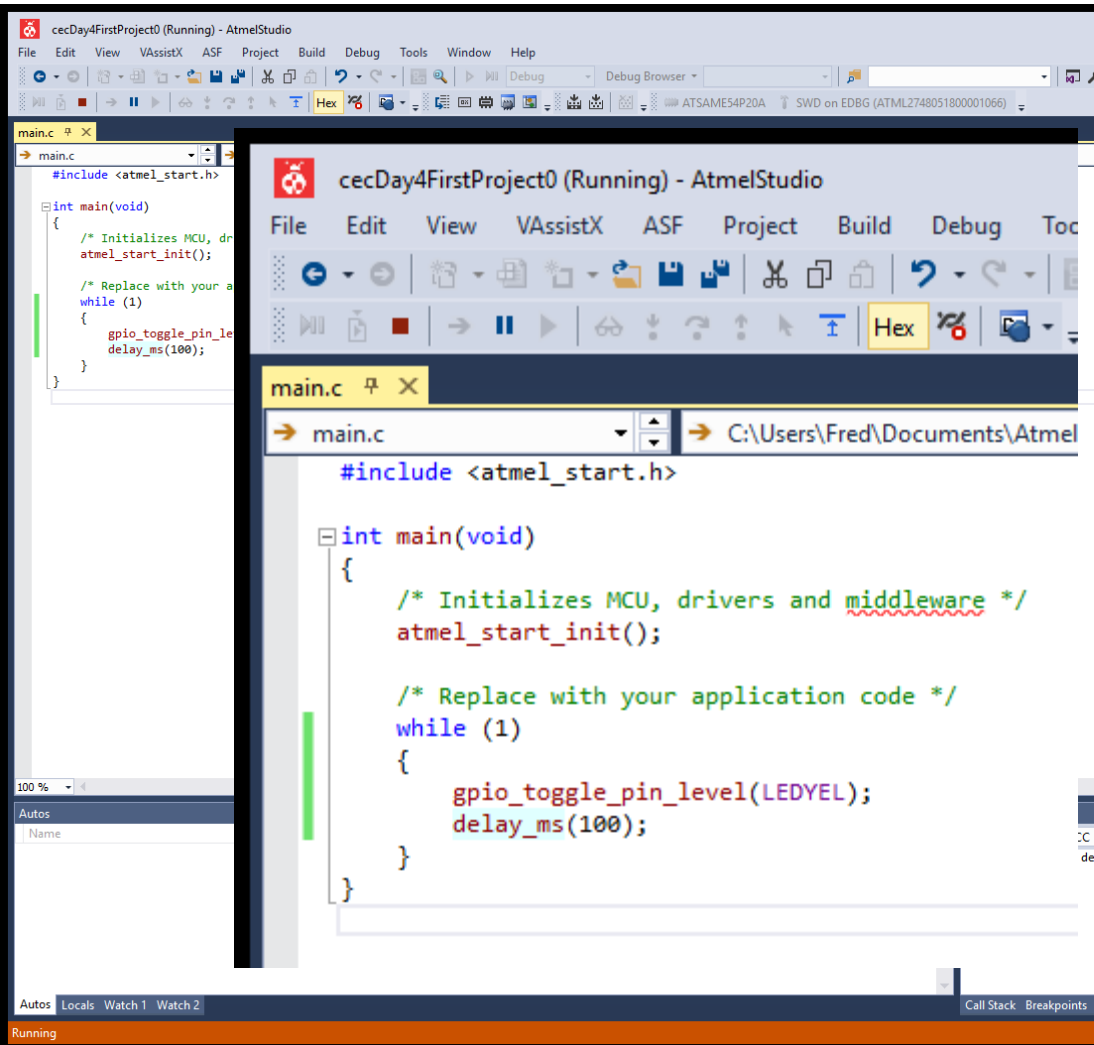
- Function: `uint32_t delay_get_version(void)`
- Comments: `/** \brief Retrieve the current driver version */`
- Return Value: `* \return Current driver version.`
- Usage: `Accept with: <TAB> or <ENTER>`

The Solution Explorer on the right shows the project structure, including folders for `src`, `utils`, `hpl`, and `hri`, and files such as `atmel_start.c`, `atmel_start.h`, `atmel_start_pins.h`, `driver_init.c`, `driver_init.h`, and `main.c`.



ARM Development Primer

Crawling Before We Walk – First Program

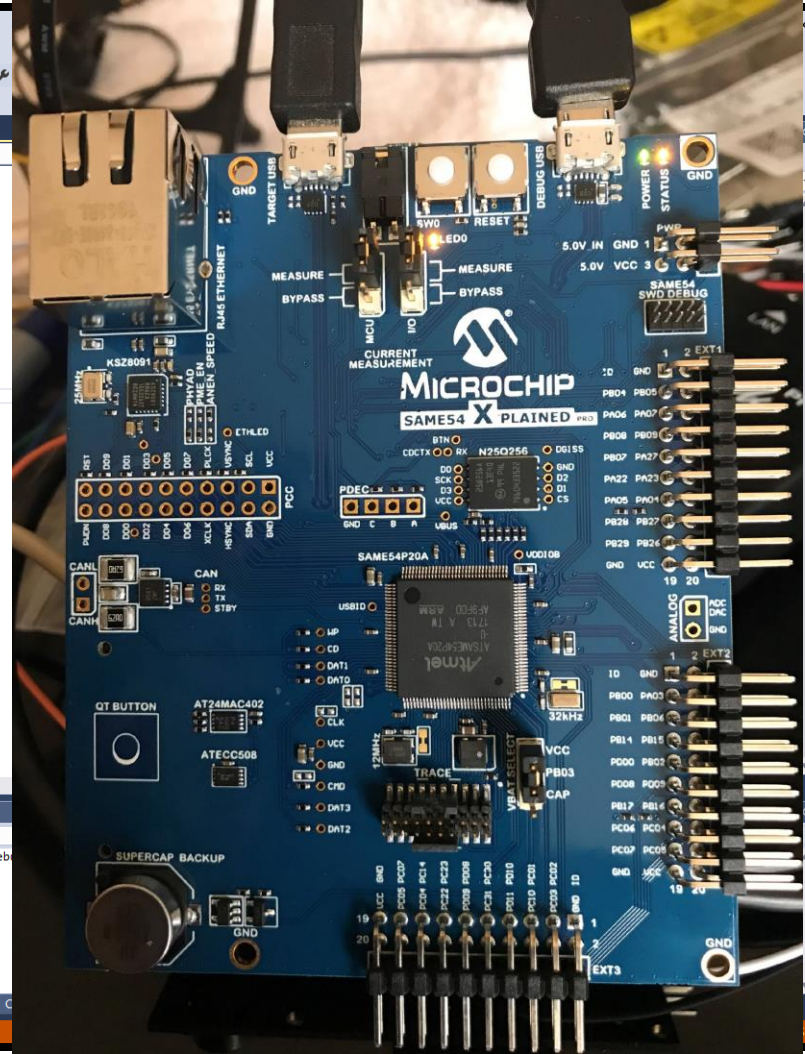


The screenshot shows the Atmel Studio IDE with a C program in the main.c file. The code is as follows:

```
#include <atmel_start.h>

int main(void)
{
    /* Initializes MCU, drivers and middleware */
    atmel_start_init();

    /* Replace with your application code */
    while (1)
    {
        gpio_toggle_pin_level(LED_YELLOW);
        delay_ms(100);
    }
}
```



ARM Development Primer

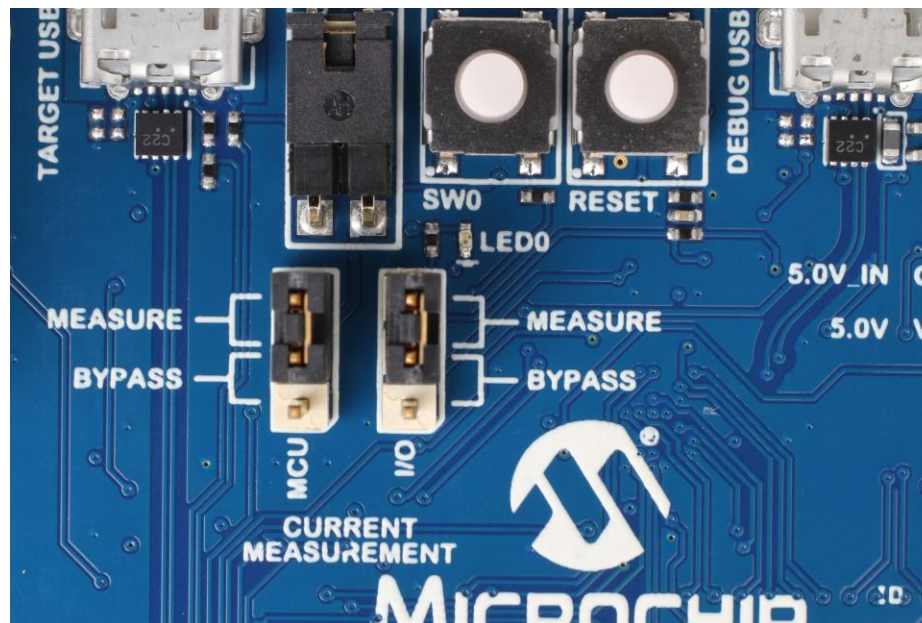
USART the ATMEL START Way

5.4.2 Virtual COM Port

The Embedded Debugger acts as a Virtual Com Port gateway by using one of the ATSAME54P20A UARTs. For further information on how to use the Virtual COM port, see [Embedded Debugger](#).

Table 5-25. Virtual COM Port Connections

SAM E54 pin	Function	Shared functionality
PB24	SERCOM2 PAD[1] UART RXD (SAM E54 RX line)	-
PB25	SERCOM2 PAD[0] UART TXD (SAM E54 TX line)	-



ARM Development Primer

USART the ATMEL START Way

Atmel | START ATSAM54P20A

← Return To Front Page | Help And Support

{ } VIEW CODE

SAVE CONFIGURATION

EXPORT PROJECT

MY SOFTWARE COMPONENTS

- Application
- Middleware
- Driver
- System driver
- Hardware

+ Add software component

Show system drivers

Show hardware

Click "Add software components" to add drivers and middleware to your project.

CECDAY4USART

USART_0VCP

SERCOM2

MCLK

GCLK

OSC32KCTRL

OSCTRL

DMAC

RAMECC



ARM Development Primer

USART the ATMEL START Way

USART_0VCP
Universal Asynchronous receiver/transmitter (UART) communication in synchronous/blocking mode

GENERAL

- User guide
- Rename component
- Remove component

COMPONENT SETTINGS

Driver: HAL:Driver:USART_Sync
Mode: UART
Instance: SERCOM2

CLOCKS

Core: Generic clock generator 0 (12 MHz)
Slow: Generic clock generator 3 (32.77 kHz)

SIGNALS

RX: PB24
TX: PB25

HAL:DRIVER:USART SYNC (UART) CONFIGURATION ON SERCOM2

BASIC CONFIGURATION

Receive buffer enable:
Transmitt buffer enable:
Frame parity: No parity
Character Size: 8 bits
Stop Bit: One stop bit
Baud rate: 9600

ADVANCED CONFIGURATION Enable:



ARM Development Primer

USART the ATMEL START Way

{ } VIEW CODE

SAVE CONFIGURATION

EXPORT PROJECT

CLOCK CONFIGURATOR

Zoom in Zoom out Reset

OSCILLATORS

- External Crystal Oscillator 8-48MHz
- External Crystal Oscillator 8-48MHz (XOSC1)
12 MHz
- 32kHz External Crystal Oscillator
- Digital Frequency Locked Loop
- Digital Phase Locked Loop
- Digital Phase Locked Loop
- 32kHz Ultra Low Power Internal Oscillator (OSCULP32K)
32.768 kHz

SOURCES

- Generic clock generator 0
12 MHz
- Generic clock generator 1
- Generic clock generator 2
- Generic clock generator 3
- Generic clock generator 4
- Generic clock generator 5
- Generic clock generator 6
- Generic clock generator 7

Reset clock settings

CPU
12 MHz

COMPONENTS

USART_OVCP
Core 12 MHz
Slow 32.768 kHz



ARM Development Primer

USART the ATMEL START Way

Atmel | START ATSAM54P20A

← Return To Front Page | Help And Support

{ } VIEW CODE

SAVE CONFIGURATION

EXPORT PROJECT

PINMUX CONFIGURATOR

# ↑	Pin label		Board label		Mode	Signal Label
	Pad	User	Header	Pin		
USART_0						
100	PB24	CDC_UA...	RXD,RXD	Peripheral IO	Peripheral IO	RX
101	PB25	CDC_UA...	TXD,TXD	Peripheral IO	Peripheral IO	TX
No software components						
1	PA00	Clock	XIN32			
2	PA01	Clock	XOUT32			
3	PC00					
4	PC01	EXT3	GPIO			
6	VDDA...					
7	PC02	EXT3	ADC(+)			
8	PC03	EXT3	ADC(-)			
9	PA02					
10	PA03	EXT2	ADC(-)			
11	PB04	EXT1	ADC(+)			
12	PB05	EXT1	ADC(-)			

PREVIEW - ATMEL_START_PINS.H

```

1  /*
2  * Code generated from Atmel Start.
3  *
4  * This file will be overwritten when reconfiguring your Atmel Start project.
5  * Please copy examples or other code you want to keep to a separate file
6  * to avoid losing it when reconfiguring.
7  */
8  #ifndef ATMEL_START_PINS_H_INCLUDED
9  #define ATMEL_START_PINS_H_INCLUDED
10
11 #include <hal_gpio.h>
12
13 // SAME54 has 14 pin functions
14
15 #define GPIO_PIN_FUNCTION_A 0
16 #define GPIO_PIN_FUNCTION_B 1
17 #define GPIO_PIN_FUNCTION_C 2
18 #define GPIO_PIN_FUNCTION_D 3
19 #define GPIO_PIN_FUNCTION_E 4
20 #define GPIO_PIN_FUNCTION_F 5
21 #define GPIO_PIN_FUNCTION_G 6
22 #define GPIO_PIN_FUNCTION_H 7
23 #define GPIO_PIN_FUNCTION_I 8
24 #define GPIO_PIN_FUNCTION_J 9
25 #define GPIO_PIN_FUNCTION_K 10
26 #define GPIO_PIN_FUNCTION_L 11
27 #define GPIO_PIN_FUNCTION_M 12
28 #define GPIO_PIN_FUNCTION_N 13
29
30 #define PB24 GPIO(GPIO_PORTB, 24)
31 #define PB25 GPIO(GPIO_PORTB, 25)
32
33 #endif // ATMEL_START_PINS_H_INCLUDED

```



ARM Development Primer

USART the ATMEL START Way

The image shows a screenshot of the Atmel Studio IDE running a project named 'cecDay4USART0'. The main window displays the source code for 'main.c' in the file 'driver_examples.c'. The code is as follows:

```
#include <atmel_start.h>

static uint8_t msg[27] = "USART the ATMEL START Way\r\n";

int main(void)
{
    /* Initializes MCU, drivers and middleware */
    atmel_start_init();
    usart_sync_enable(&USART_0VCP);
    /* Replace with your application code */
    while (1)
    {
        io_write(&USART_0VCP.io,msg,27);
        delay_ms(100);
    }
}
```

The Solution Explorer on the right shows the project structure:

- Solution 'cecDay4USART0' (1 project)
 - Dependencies
 - Output Files
 - Libraries
 - Config
 - Device_Startup
 - examples
 - driver_examples.c
 - driver_examples.h
 - hal
 - hpl
 - hri
 - atmel_start.c
 - atmel_start.h
 - atmel_start_pins.h
 - driver_init.c
 - driver_init.h
 - main.c



ARM Development Primer

Day 4's Done

The screenshot shows a 'Serial Input/Output Monitor' application window. The main display area contains a repeating sequence of text: 'USART the ATMEL START Way'. Below this, a hex dump shows the ASCII values of the text: '55 53 41 52 54 20 74 68 65 20 41 54 4D 45 4C 20'. The application interface includes a menu bar (File, Edit, View, Configuration), a toolbar with buttons for ASCII, HEX, ASCII Send, and HEX Send, and a 'Send' button at the bottom right. A 'Disconnect' button is also visible. The background of the screenshot is a photograph of a blue Microchip SAME54 X PLAINED development board. The board features an ATmel ATmega320P microcontroller, a USB-to-UART bridge, a CAN interface, and various other components like a supercapacitor and a debugger connector.