



**DesignNews**

# Embedded Studio Primer

## DAY 4 : Microchip SAM Series

Sponsored by



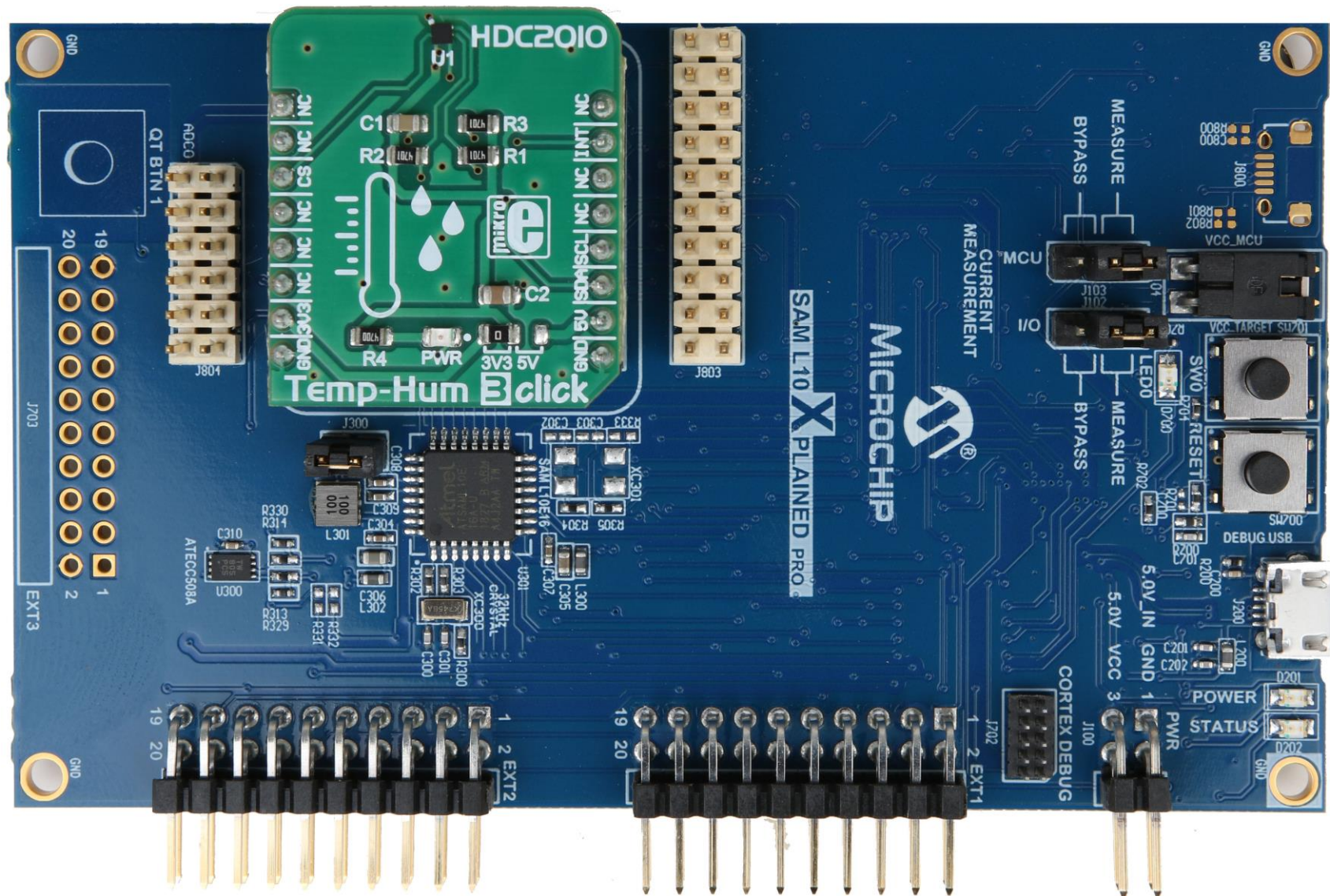
# Webinar Logistics

- Turn on your system sound to hear the streaming presentation.
- If you have technical problems, click “Help” or submit a question asking for assistance.
- Participate in ‘Group Chat’ by maximizing the chat widget in your dock.
- Submit questions for the lecturer using the Q&A widget. They will follow-up after the lecture portion concludes.



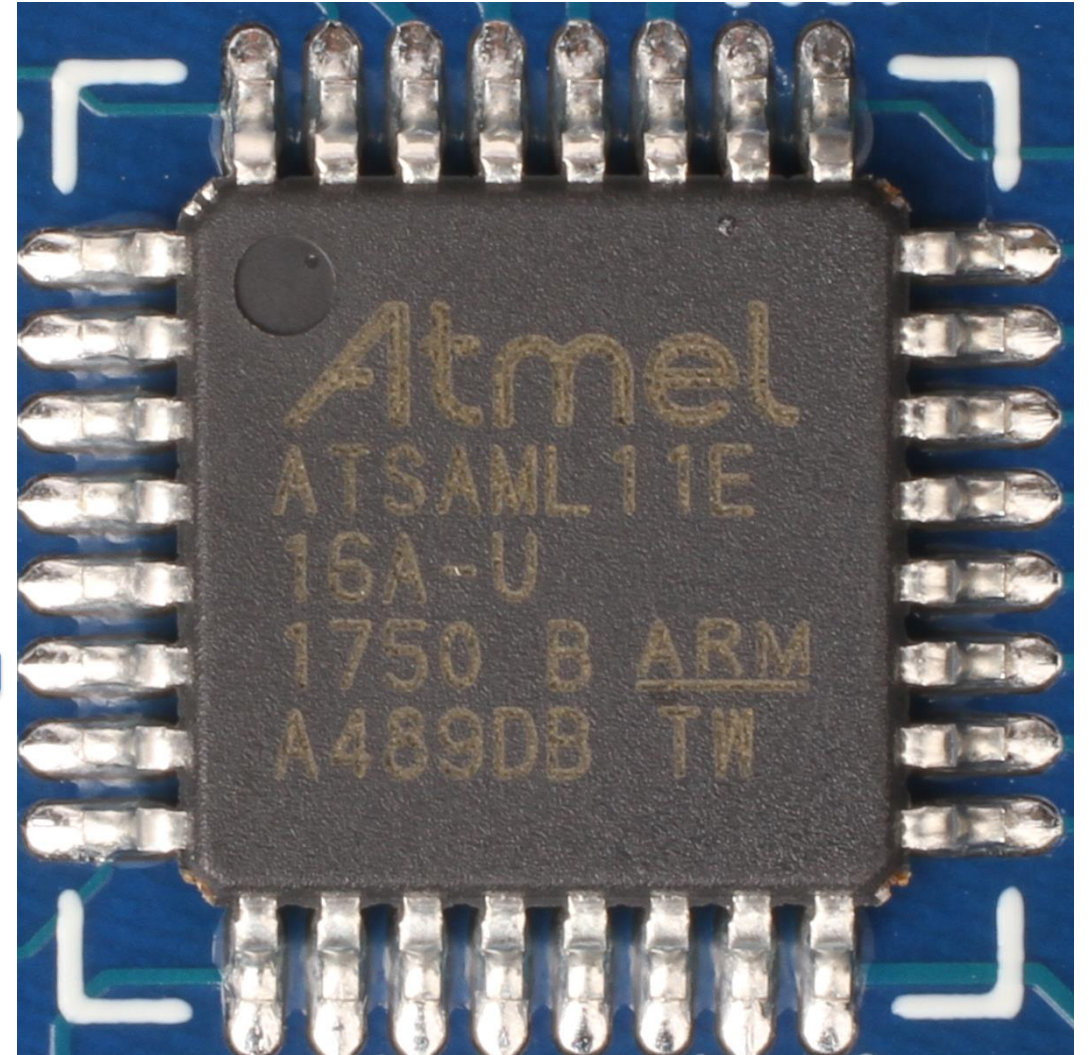
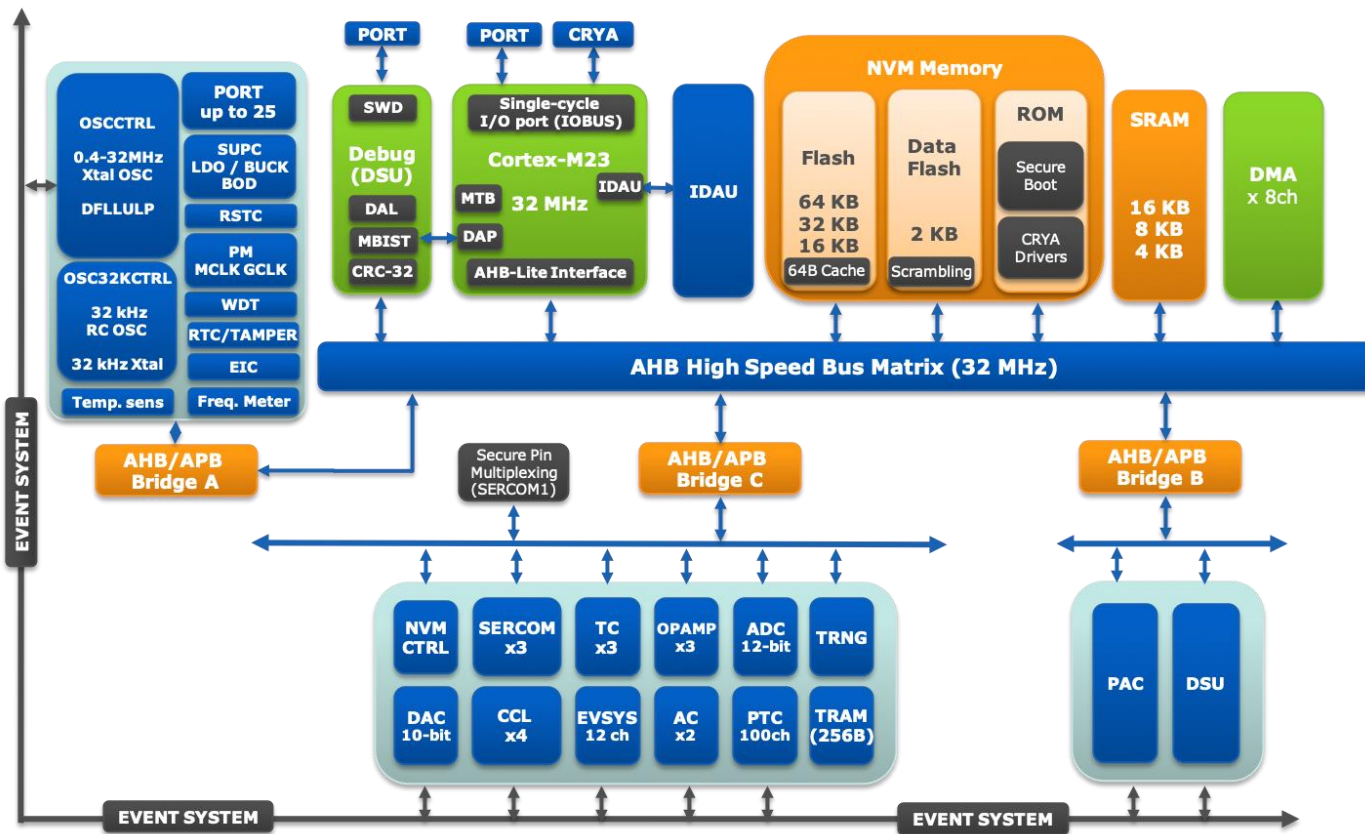
## **Fred Eady**

Visit 'Lecturer Profile' in your console for more details.



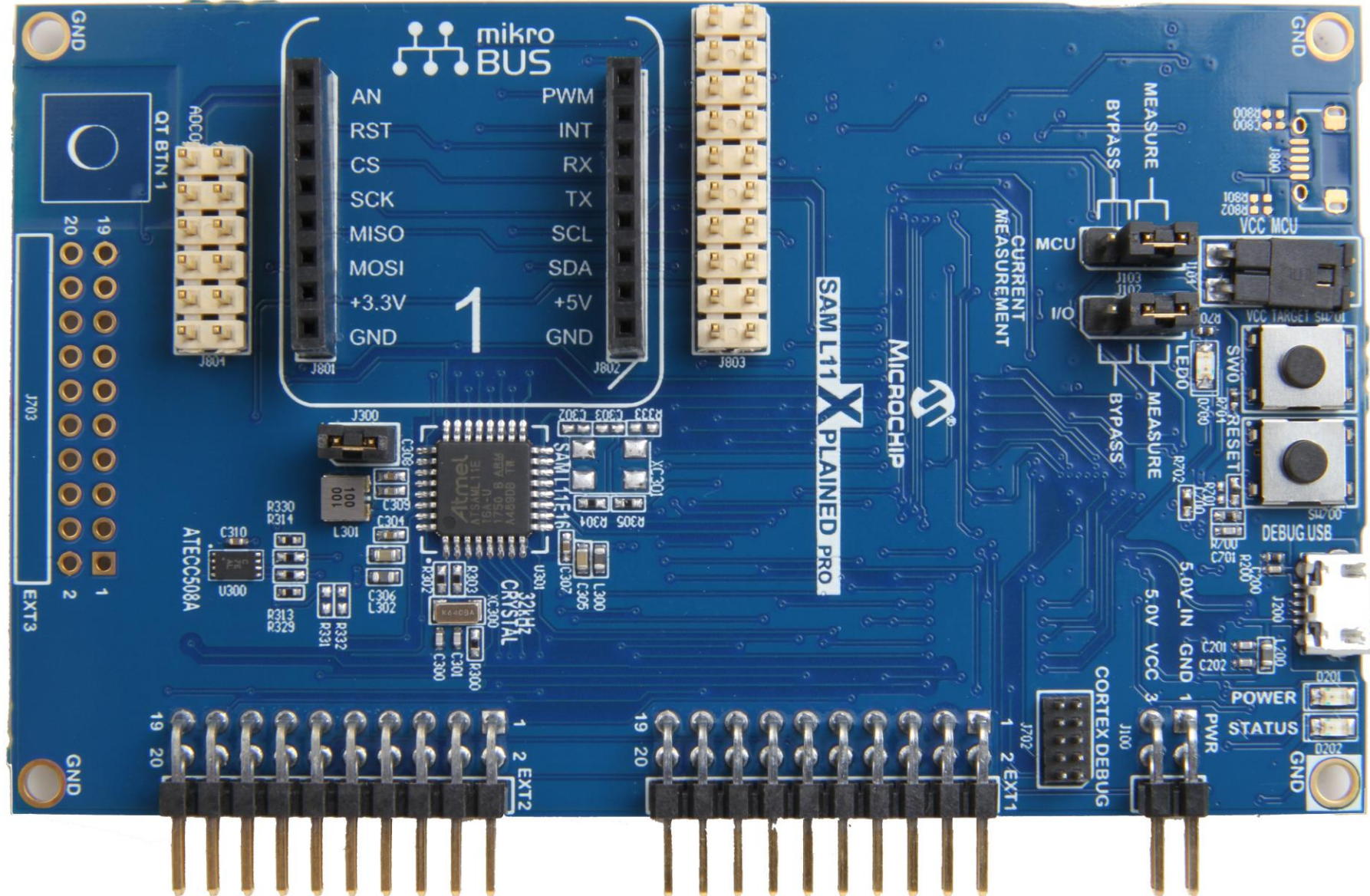
## AGENDA

- Programming the SAML11
- Programming the SAML10



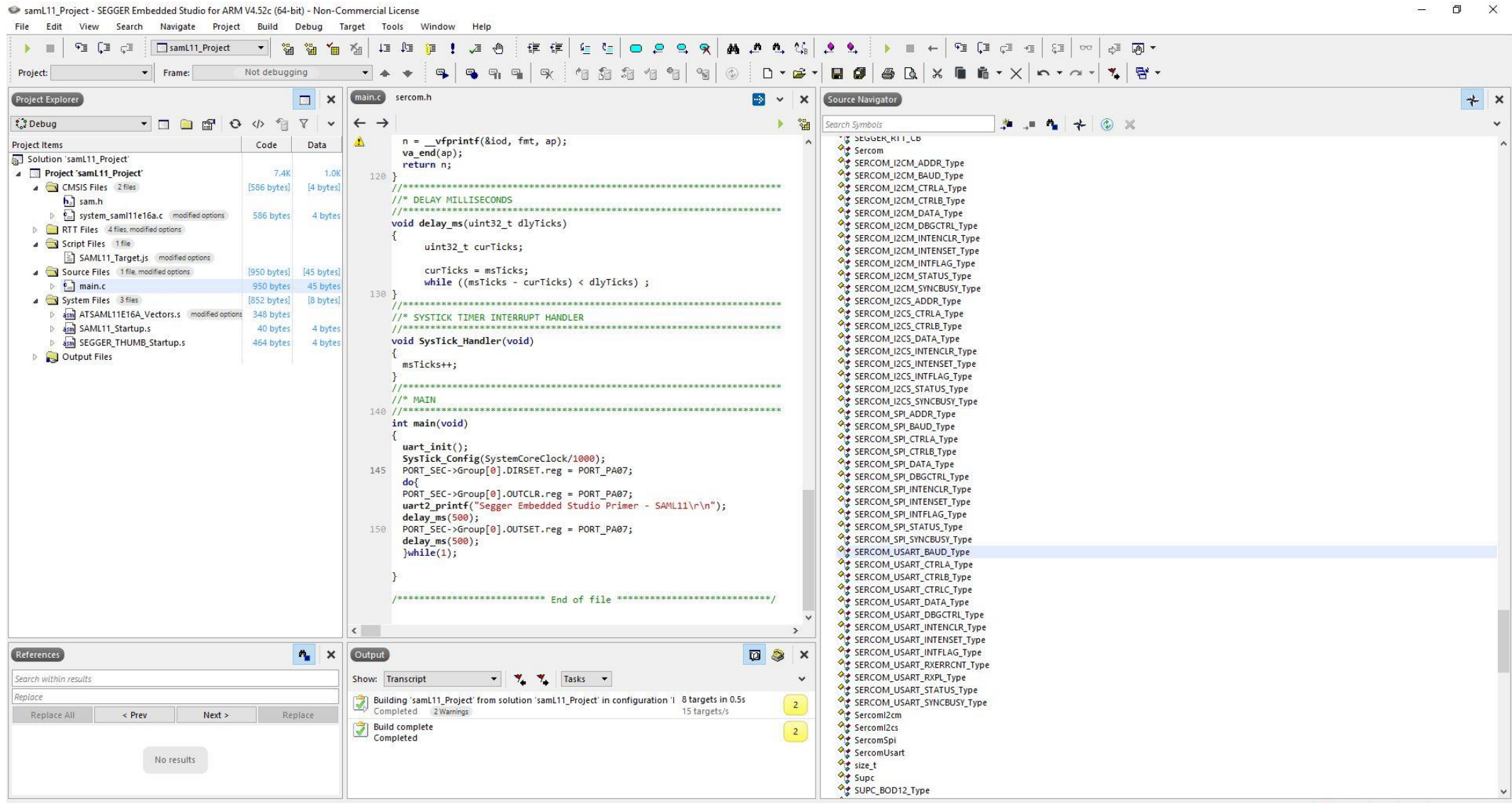
# Embedded Studio Primer

## Programming the SAML11



# Embedded Studio Primer

## Programming the SAML11



The screenshot displays the SEGGER Embedded Studio IDE for ARM V4.52c. The main window shows the source code for `main.c` in the `sercom.h` project. The code includes a `delay_ms` function and a `SysTick_Handler` function. The `main` function initializes UART and SysTick, prints a message, and enters a loop with delays.

```

n = _vfprintf(&ioid, fmt, ap);
va_end(ap);
return n;

120 }
//*****
//** DELAY MILLISECONDS
//*****
void delay_ms(uint32_t dlyTicks)
{
    uint32_t curTicks;

    curTicks = mstTicks;
    while ((mstTicks - curTicks) < dlyTicks) ;
}
//*****
//** SYSTICK TIMER INTERRUPT HANDLER
//*****
void SysTick_Handler(void)
{
    mstTicks++;
}
//*****
//** MAIN
//*****
140
int main(void)
{
    uart_init();
    SysTick_Config(SystemCoreClock/1000);
    PORT_SEC->Group[0].DIRSET.reg = PORT_PA07;
    do{
    PORT_SEC->Group[0].OUTCLR.reg = PORT_PA07;
    145    uart2_printf("Segger Embedded Studio Primer - SAML11\r\n");
    delay_ms(500);
    PORT_SEC->Group[0].OUTSET.reg = PORT_PA07;
    150    delay_ms(500);
    }while(1);
}

//***** End of file *****/

```

The Project Explorer on the left shows the project structure for 'saml11\_Project', including source files like `main.c` and `sercom.h`. The Source Navigator on the right lists various hardware-related symbols. The Output window at the bottom shows build messages, including 'Building 'saml11\_Project' from solution 'saml11\_Project' in configuration '1' 8 targets in 0.5s' and 'Build complete Completed'.

## Programming the SAML11 - UART

```

//*****
/* SAML11 UART DEFINITIONS
//*****

#define UART_TX_PORT          0 // PORTA-GROUP 0
#define UART_TX_PIN          PIN_PA08D_SERCOM2_PAD0
#define UART_TX_MUX          MUX_PA08D_SERCOM2_PAD0

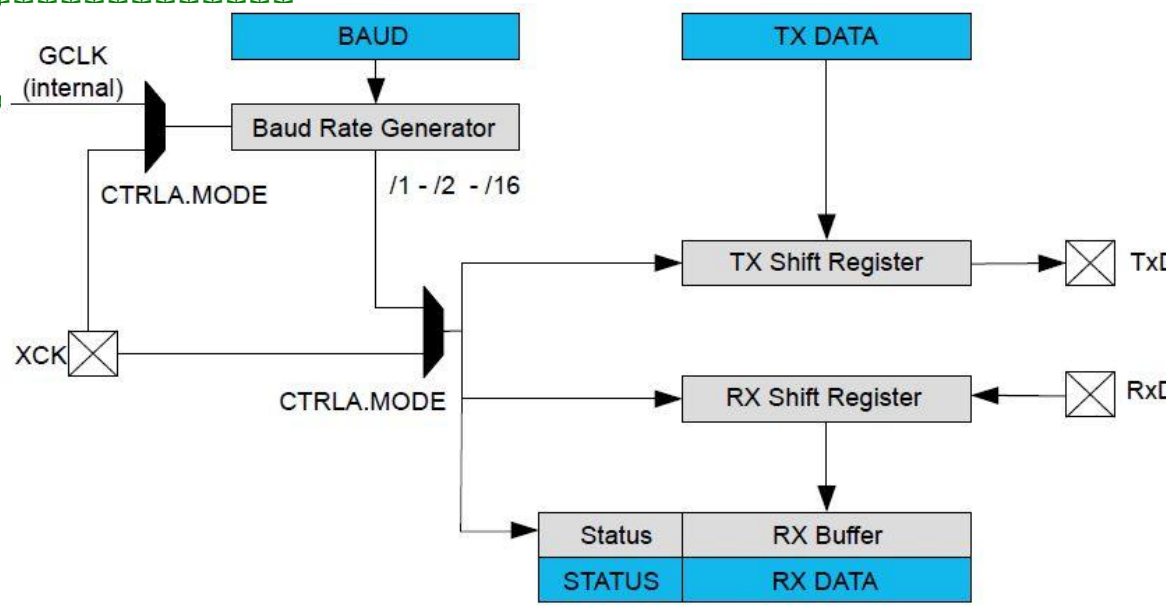
#define UART_RX_PORT          0 // PORTA-GROUP 0
#define UART_RX_PIN          PIN_PA09D_SERCOM2_PAD1
#define UART_RX_MUX          MUX_PA09D_SERCOM2_PAD1

#define UART_SERCOM          SERCOM2
#define UART_SERCOM_GCLK_ID  SERCOM2_GCLK_ID_CORE
#define UART_SERCOM_MASK_REG APBCMASK
#define UART_SERCOM_MASK_BIT MCLK_APBCMASK_SERCOM2_Msk
#define UART_SERCOM_TXPO     SERCOM_USART_CTRLA_TXPO(0/*PAD0*/)
#define UART_SERCOM_RXPO     SERCOM_USART_CTRLA_RXPO(1/*PAD1*/)

#define UART_BAUDRATE        9600

#define BAUD_VAL              (SystemCoreClock / (16 * UART_BAUDRATE))
#define FP_VAL                SystemCoreClock / UART_BAUDRATE - 16 * BAUD_VAL) / 2)

```

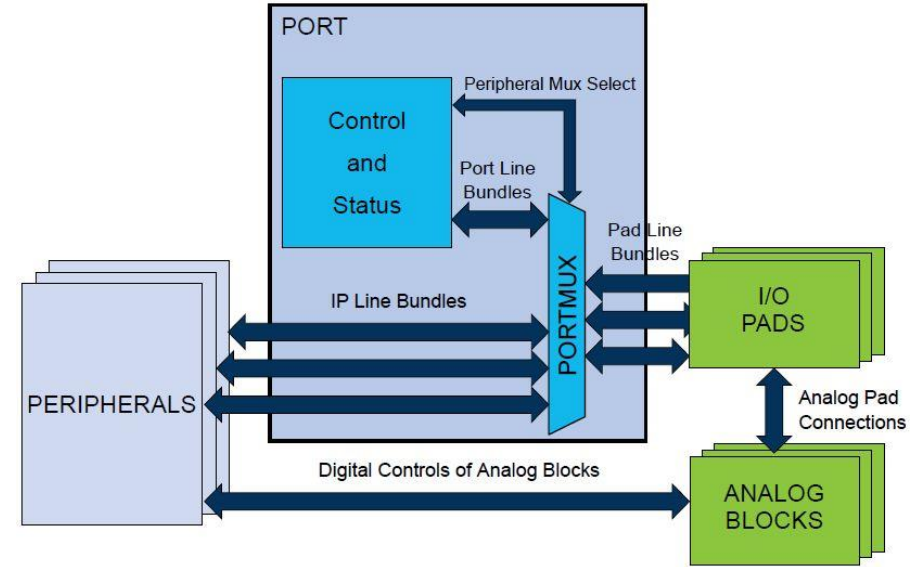




## Programming the SAML11 - UART

```
#define UART_TX_PORT      0 // PORTA-GROUP 0
#define UART_TX_PIN      PIN_PA08D_SERCOM2_PAD0
#define UART_TX_MUX      MUX_PA08D_SERCOM2_PAD0

#define UART_RX_PORT      0 // PORTA-GROUP 0
#define UART_RX_PIN      PIN_PA09D_SERCOM2_PAD1
#define UART_RX_MUX      MUX_PA09D_SERCOM2_PAD1
```



```
#define CONFIGURE_PMUX(dir) \
do { \
    PORT_SEC->Group[UART_##dir##_PORT].PINCFG[UART_##dir##_PIN].reg |= PORT_PINCFG_PMUXEN; \
    if (UART_##dir##_PIN & 1) \
        PORT_SEC->Group[UART_##dir##_PORT].PMUX[UART_##dir##_PIN >> 1].bit.PMUXO = UART_##dir##_MUX; \
    else \
        PORT_SEC->Group[UART_##dir##_PORT].PMUX[UART_##dir##_PIN >> 1].bit.PMUXE = UART_##dir##_MUX; \
} while (0)
```

```
static void uart_init(void)
{
    CONFIGURE_PMUX(RX);
    CONFIGURE_PMUX(TX);

    MCLK->UART_SERCOM_MASK_REG.reg |= UART_SERCOM_MASK_BIT;

    GCLK->PCHCTRL[UART_SERCOM_GCLK_ID].reg = GCLK_PCHCTRL_GEN(0) | GCLK_PCHCTRL_CHEN_Msk;
    while (0 == (GCLK->PCHCTRL[UART_SERCOM_GCLK_ID].reg & GCLK_PCHCTRL_CHEN_Msk));

    UART_SERCOM->USART.CTRLA.reg =
        SERCOM_USART_CTRLA_DORD | SERCOM_USART_CTRLA_MODE(1/*USART_INT_CLK*/) |
        SERCOM_USART_CTRLA_FORM(0/*USART*/) | SERCOM_USART_CTRLA_SAMPRT(1) |
        UART_SERCOM_TXPO | UART_SERCOM_RXPO;

    UART_SERCOM->USART.CTRLB.reg = SERCOM_USART_CTRLB_RXEN | SERCOM_USART_CTRLB_TXEN |
    SERCOM_USART_CTRLB_CHSIZE(0/*8 bits*);

    UART_SERCOM->USART.BAUD.reg = SERCOM_USART_BAUD_FRACFP_BAUD(BAUD_VAL) |
    SERCOM_USART_BAUD_FRACFP_FP(FP_VAL);

    UART_SERCOM->USART.CTRLA.reg |= SERCOM_USART_CTRLA_ENABLE;
}
```

## Programming the SAML11 - printf

```

//*****
//* PRINTF STUFF
//*****
typedef struct __printf_tag
{
    size_t charcount;
    size_t maxchars;
    char *string;
    int (*output_fn)(int, struct __printf_tag *ctx);
} __printf_t;

void uart2_putc(char c)
{
    while(!((UART_SERCOM->USART.INTFLAG.reg & SERCOM_USART_INTENCLR_DRE_Msk));
    UART_SERCOM->USART.DATA.reg = c;
}

int uart2_printf(const char *fmt, ...)
{
    int n;
    va_list ap;
    __printf_t iod;
    va_start(ap, fmt);
    iod.string = 0;
    iod.maxchars = INT_MAX;
    iod.output_fn = uart2_putc;
    n = __vfprintf(&iod, fmt, ap);
    va_end(ap);
    return n;
}

```

### SEGGER Embedded Studio for ARM printf-style output

SEGGER Embedded Studio for ARM provides a solution for just this case by using some internal functions and data types in the SEGGER Embedded Studio for ARM library. These functions and types are define in the header file <\_\_vfprintf.h>.

The first thing to introduce is the `__printf_t` type which captures the current state and parameters of the format conversion:

```

typedef struct __printf_tag
{
    size_t charcount;
    size_t maxchars;
    char *string;
    int (*output_fn)(int, struct __printf_tag *ctx);
} __printf_t;

```

This type is used by the library functions to direct what the formatting routines do with each character they need to output. If `string` is non-zero, the character is appended to the string pointed to by `string`; if `output_fn` is non-zero, the character is output through the function `output_fn` with the context passed as the second parameter.

The member `charcount` counts the number of characters currently output, and `maxchars` defines the maximum number of characters output by the formatting routine `__vfprintf`.

We can use this type and function to rewrite `uart0_printf`:

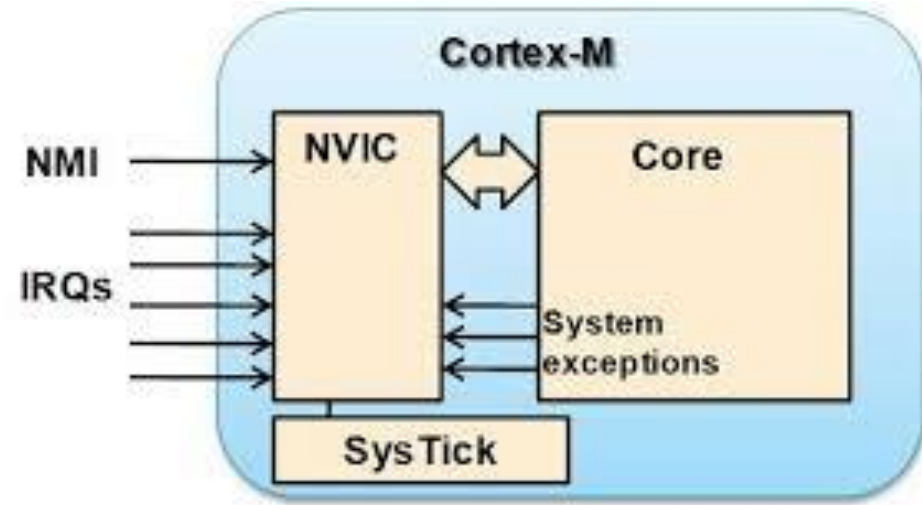
```

int uart0_printf(const char *fmt, ...)
{
    int n;
    va_list ap;
    __printf_t iod;
    va_start(ap, fmt);
    iod.string = 0;
    iod.maxchars = INT_MAX;
    iod.output_fn = uart0_putc;
    n = __vfprintf(&iod, fmt, ap);
    va_end(ap);
    return n;
}

```

## Programming the SAML11 – delay\_ms

```
/** *****  
/** DELAY MILLISECONDS  
/** *****  
void delay_ms(uint32_t dlyTicks)  
{  
    uint32_t curTicks;  
  
    curTicks = msTicks;  
    while ((msTicks - curTicks) < dlyTicks) ;  
}  
/** *****  
/** SYSTICK TIMER INTERRUPT HANDLER  
/** *****  
void SysTick_Handler(void)  
{  
    msTicks++;  
}
```



```

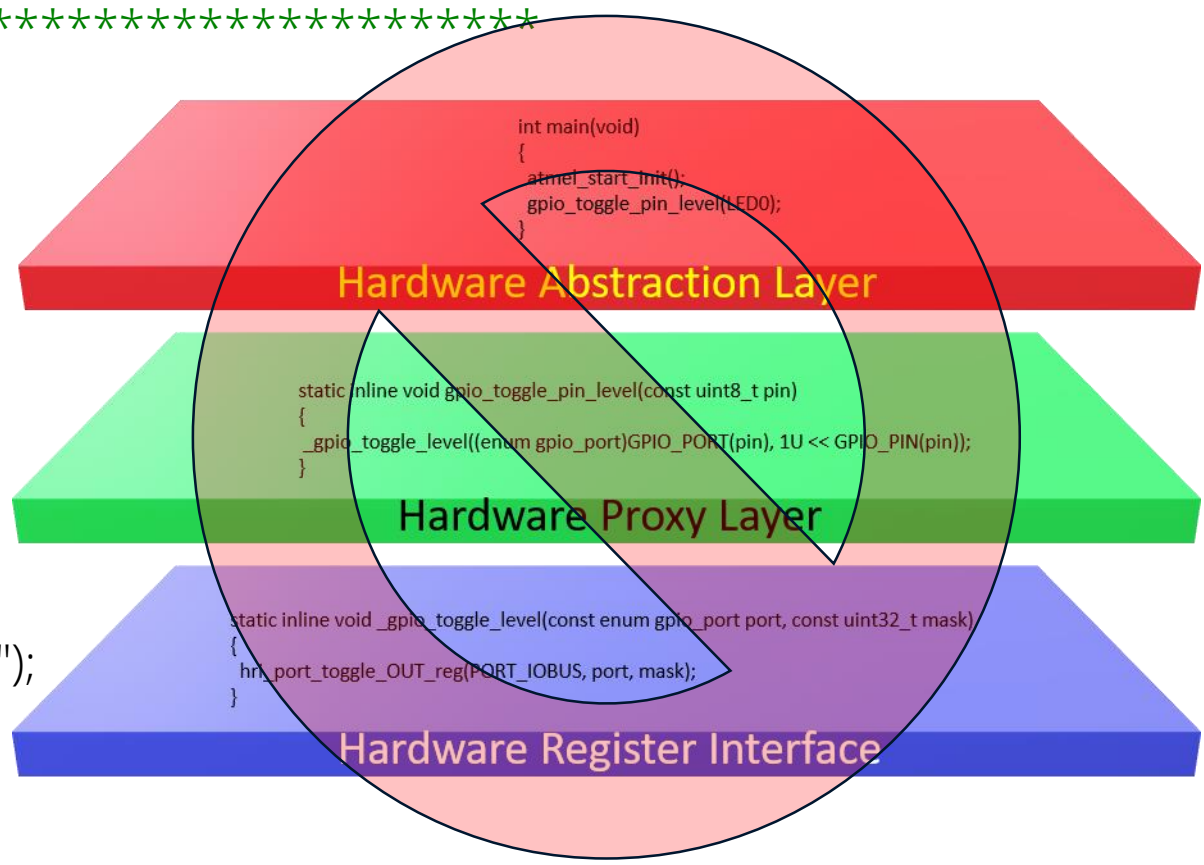
//*****
//* MAIN
//*****
int main(void)
{
    uart_init();

    SysTick_Config(SystemCoreClock/1000);

    PORT_SEC->Group[0].DIRSET.reg = PORT_PA07;

    do{
        PORT_SEC->Group[0].OUTCLR.reg = PORT_PA07;
        uart2_printf("Segger Embedded Studio Primer - SAML11\r\n");
        delay_ms(500);
        PORT_SEC->Group[0].OUTSET.reg = PORT_PA07;
        delay_ms(500);
    }while(1);
}

```

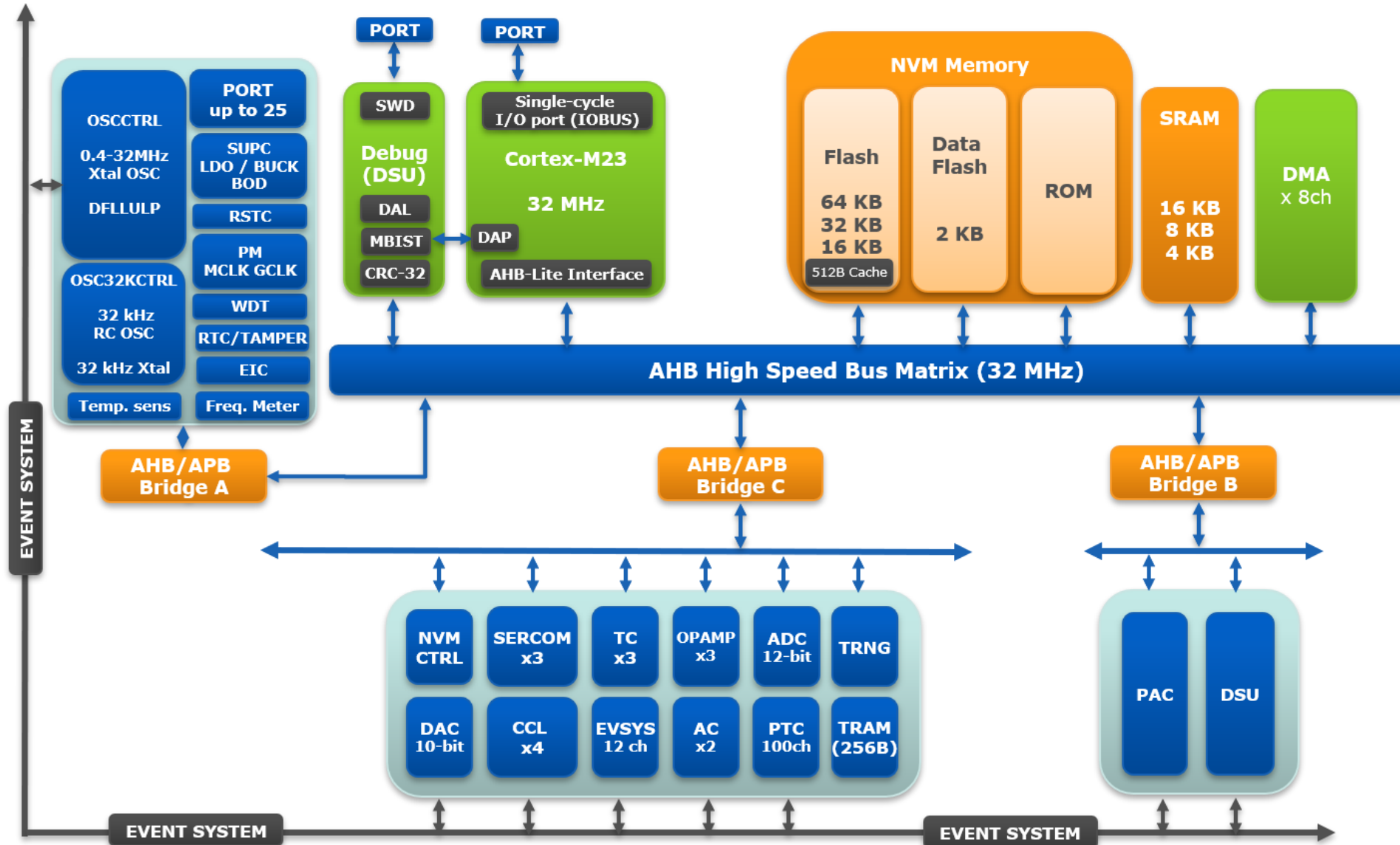






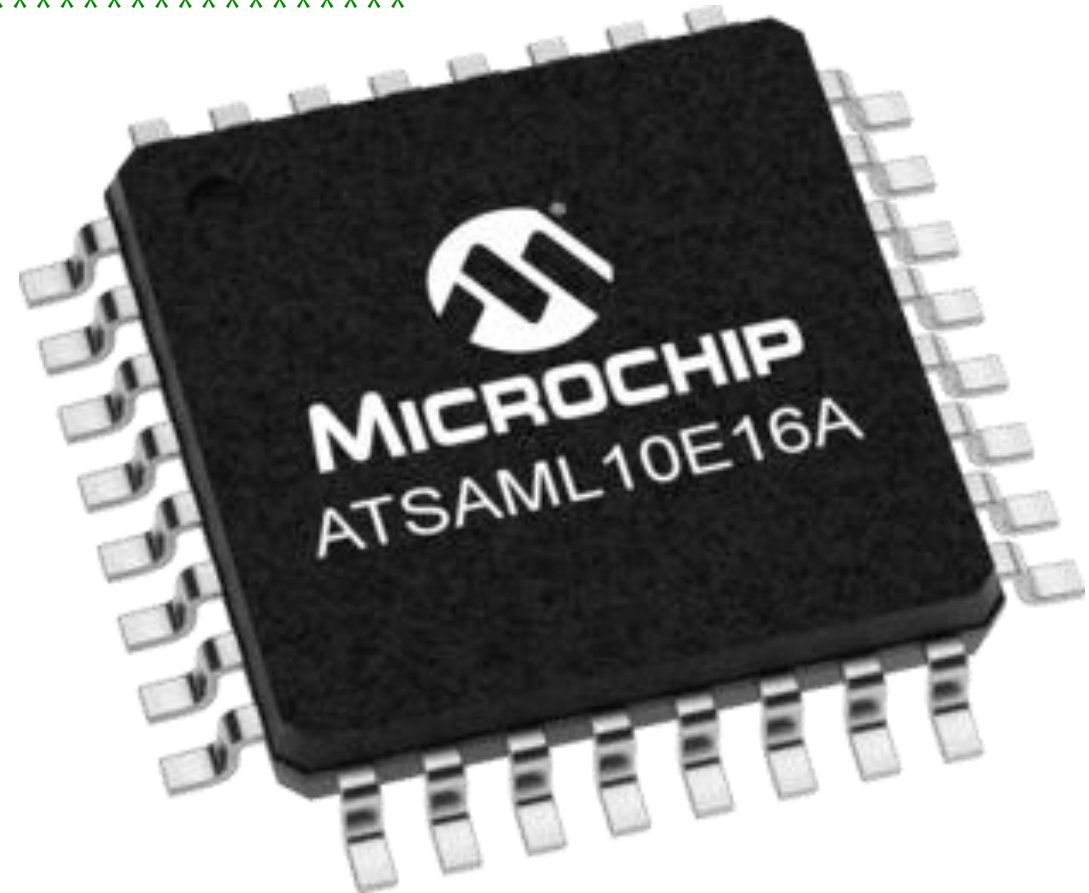
# Embedded Studio Primer

## Programming the SAML10





```
/** *****  
/** FUNCTION PROTOTYPES  
/** *****  
void resetI2C(void);  
void enableI2C(void);  
void disableI2C(void);  
void initClockI2C(void);  
void initI2C(uint32_t baudrate);  
bool startTransmission(uint8_t addr, READ_WRITE_BIT rwBit);  
bool isBusIdle(void);  
bool isBusOwner(void);  
bool isArbLost(void);  
bool isBusBusy(void);  
bool isDataReady(void);  
bool isStopDetected(void);  
bool isRXNackReceived(void);  
uint8_t available(void);  
void setup_nackI2C(void);  
void setup_ackI2C(void);  
void setup_cmdI2C(uint8_t cmd);  
bool sendData(uint8_t data);  
uint8_t readDataToBuf(uint8_t addr, uint8_t quantity, bool stopBit);
```

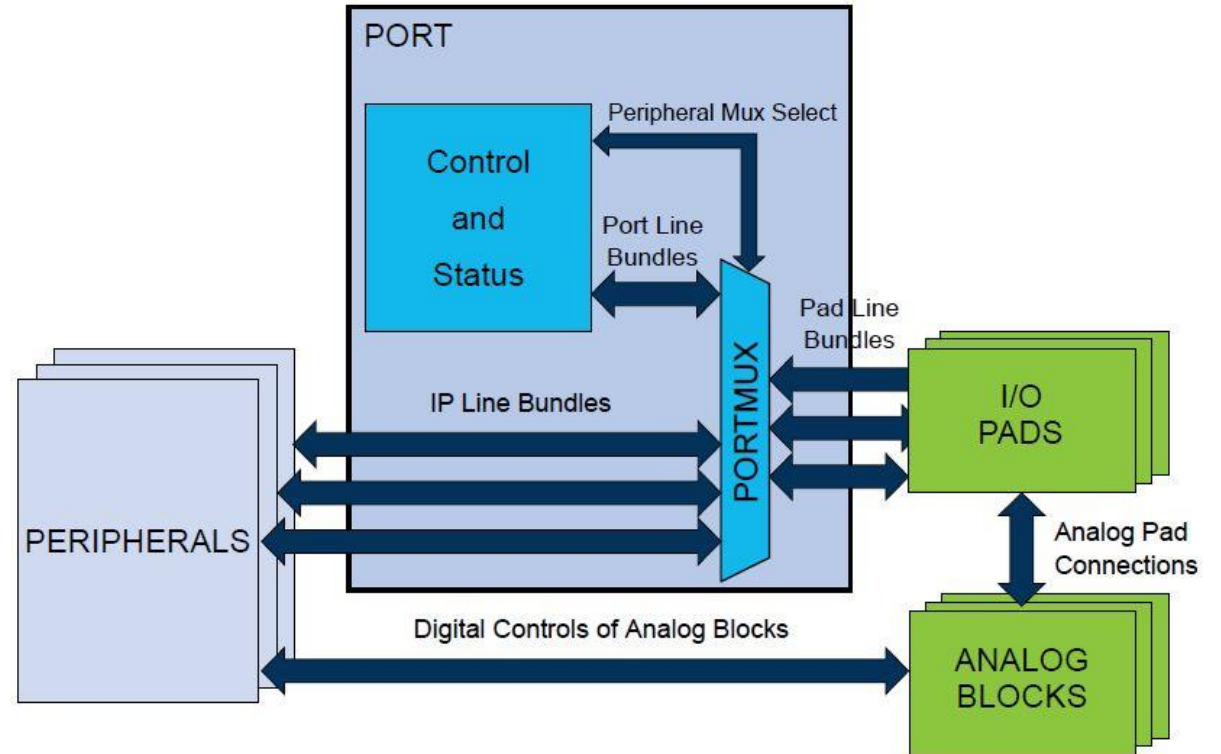


```

//*****
//* SAML10 I2C PINS AND PORTS
//*****

#define I2C_SDA_PORT      0 // PORTA IN GROUP 0
#define I2C_SDA_PIN      PIN_PA16C_SERCOM1_PAD0
#define I2C_SDA_MUX      MUX_PA16C_SERCOM1_PAD0

#define I2C_SCL_PORT     0 // PORTA IN GROUP 0
#define I2C_SCL_PIN     PIN_PA17C_SERCOM1_PAD1
#define I2C_SCL_MUX     MUX_PA17C_SERCOM1_PAD1
    
```



```

//*****
/* I2C INIT CLOCK
//*****

```

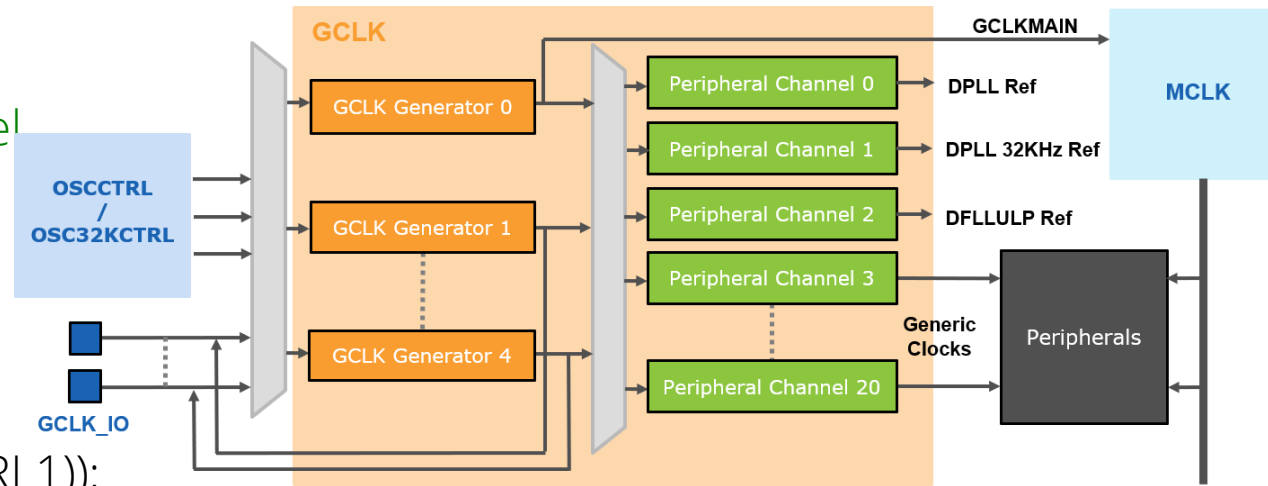
```

void initClockI2C(void)
{
//Set clock generator, clock source and peripheral channel
GCLK->GENCTRL[1].reg = GCLK_GENCTRL_DIV(1) |
    GCLK_GENCTRL_SRC_OSC16M |
    GCLK_GENCTRL_GENEN |
    GCLK_GENCTRL_OE ;

while((GCLK->SYNCBUSY.reg & GCLK_SYNCBUSY_GENCTRL1));

GCLK->PCHCTRL[SERCOM1_GCLK_ID_CORE].reg = (GCLK_PCHCTRL_GEN_GCLK1 | GCLK_PCHCTRL_CHEN);
}

```



## Programming the SAML10 – initI2C

```

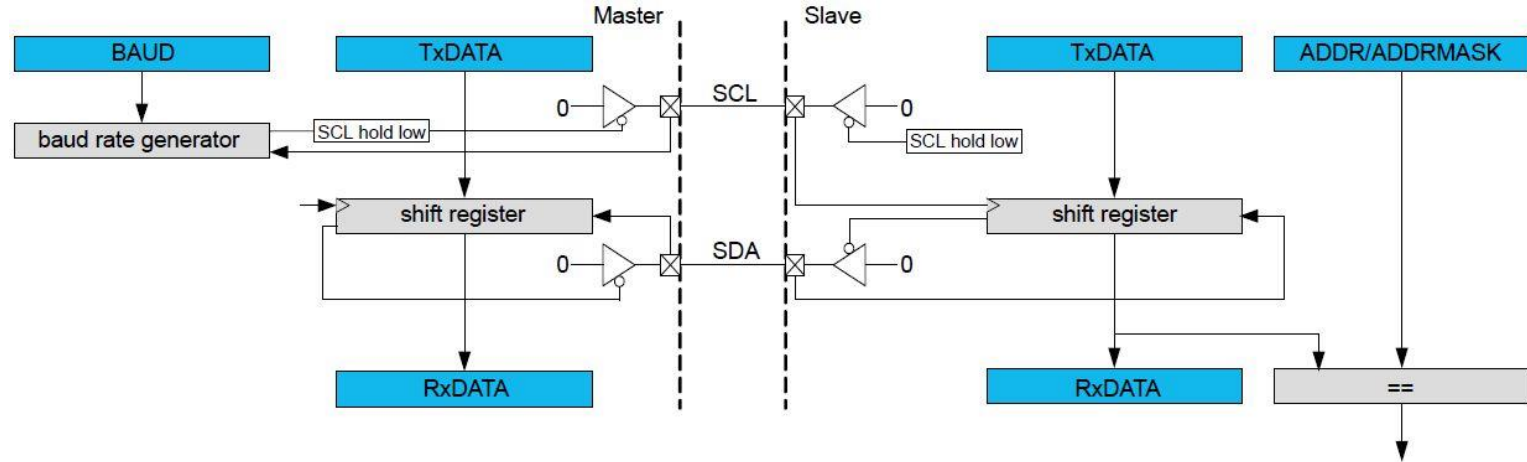
#define I2C_Master 0x05
#define riseTime_nS 125
//*****
//* I2C INIT MASTER - SERCOM1
//*****
void initI2C(uint32_t baudrate)
{
//Initialize the I2C peripheral clock
initClockI2C();

resetI2C();

//Set I2C master mode - I2C_Master = 0x05
SERCOM1->I2CM.CTRLA.reg = SERCOM_I2CM_CTRLA_MODE(I2C_Master);

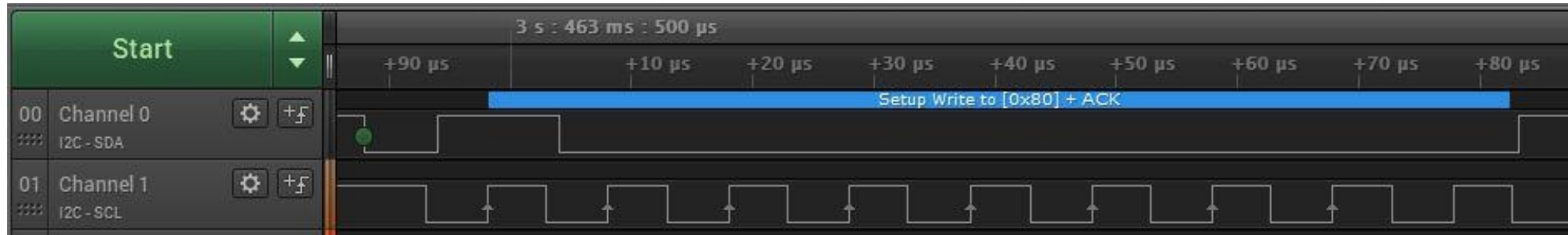
// Synchronous arithmetic baudrate
SERCOM1->I2CM.BAUD.bit.BAUD = SystemCoreClock / ( 2 * baudrate) - 5 - (((SystemCoreClock / 1000000) *
riseTime_nS) / (2 * 1000));
}

```



## Programming the SAML10 – startTransmission

```
/** *****  
/** I2C START TRANSMISSION  
/** *****  
bool startTransmission(uint8_t addr, READ_WRITE_BIT rwBit)  
{  
    // 7-bits address + rwBit  
    addr = (addr << 0x01) | rwBit;  
  
    if(!isBusOwner())  
    {  
        if( isBusBusy() || (isArbLost() && !isBusIdle()) )  
        {  
            return false;  
        }  
    }  
  
    // Send start and address  
    SERCOM1->I2CM.ADDR.bit.ADDR = addr;
```



```
// Address Transmitted
if ( rwBit == writeBit ) // Write mode
{
    // Wait for transmission to complete
    while( !SERCOM1->I2CM.INTFLAG.bit.MB );
    // Check for loss of arbitration (multiple masters starting communication at the same time)
    if(!isBusOwner())
    {
        // Restart communication
        startTransmission(addr >> 1, rwBit);
    }
}
else // Read mode (rwBit = readBit)
{
    while( !SERCOM1->I2CM.INTFLAG.bit.SB )
    {
        // If the slave NACKS the address, the MB bit will be set.
        if (SERCOM1->I2CM.INTFLAG.bit.MB)
        {
            SERCOM1->I2CM.CTRLB.bit.CMD = 3; // Stop condition
            return false;
        }
    }
}
```

## Programming the SAML10 – sendData

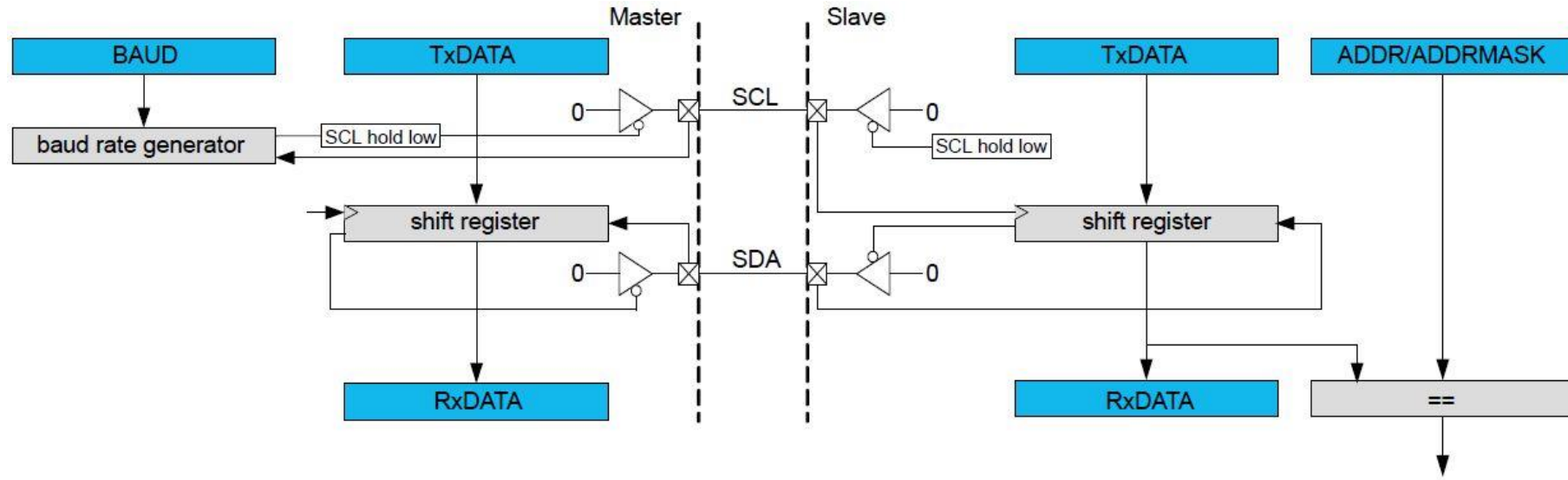
```

//*****
/* I2C SEND DATA
//*****
bool sendData(uint8_t data)
{
  //Send data
  SERCOM1->I2CM.DATA.bit.DATA = data;

  //Wait for successful transmission
  while(!SERCOM1->I2CM.INTFLAG.bit.MB)
  {
    if (SERCOM1->I2CM.STATUS.bit.BUSERR)
    {
      return false;
    }
  }

  //nack received?
  if(SERCOM1->I2CM.STATUS.bit.RXNACK)
    return false;
  else
    return true;
}

```



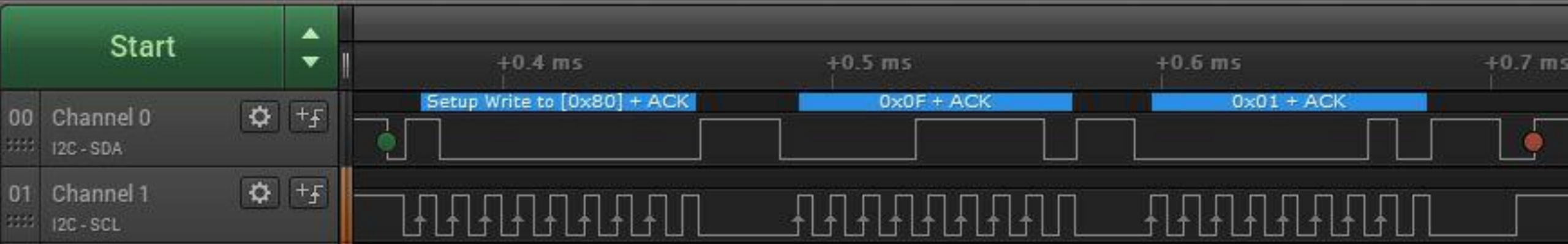
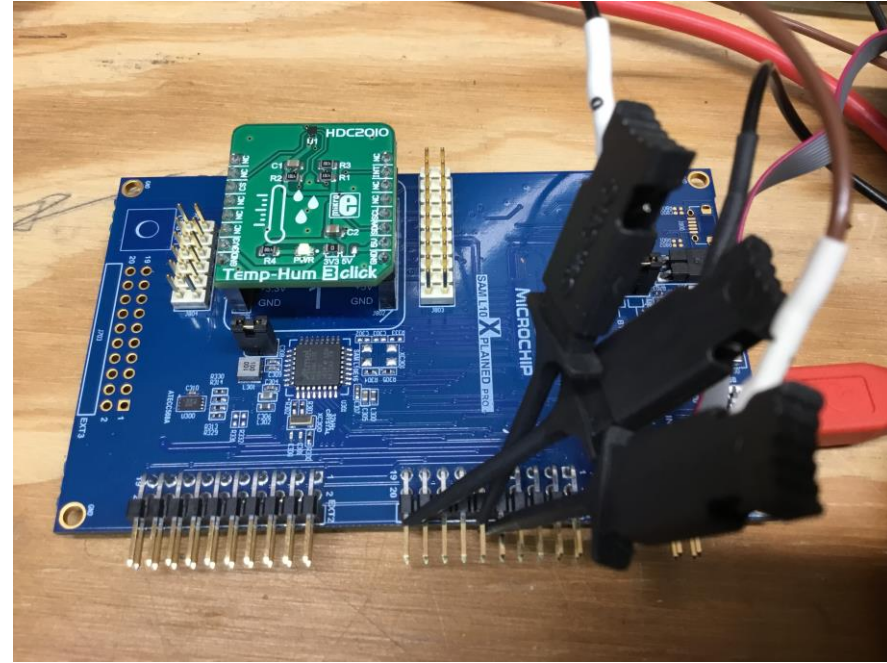
```

//*****
/* MAIN
//*****
int main(void)
{
  uint8_t rc;

  PORT->Group[0].DIRSET.reg = PORT_PA05;
  PORT->Group[0].OUTCLR.reg = PORT_PA05;
  initI2C(100000);
  enableI2C();
  CONFIGURE_PMUX(SDA);
  CONFIGURE_PMUX(SCL);

  startTransmission(0x40,writeBit);
  sendData(MEASUREMENT_CONFIG);
  sendData(0x01);
  setup_cmdI2C(MASTER_ACT_STOP);

```





saml10Primer - SEGGER Embedded Studio for ARM V4.52c (64-bit) - Non-Commercial License (Stopped)

File Edit View Search Navigate Project Build Debug Target Tools Window Help

Disassembly

```

main + 0x22a
00000FC2  F7FFFB1D  bl 0x00000600 <__aeabi_d2f>
00000FC6  1C02      adds r2, r0, #0
00000FC8  F2400304  movw r3, #4
00000FCC  F2C20300  movt r3, #0x2000
00000FD0  601A     str r2, [r3]
-----
main.c -- 410
humidity = ((float)humidVal/65536)*100;
00000FD2  F2400314  movw r3, #20
00000FD6  F2C20300  movt r3, #0x2000
00000FDA  8B1B     ldrh r3, [r3]
00000FDC  0018     movs r0, r3
00000FDE  F7FFFB41  bl 0x00000664 <__aeabi_ui2f>
00000FE2  1C03     adds r3, r0, #0
00000FE4  218F     movs r1, #0x8F
00000FE6  05C9     lsls r1, r1, #23
00000FE8  1C18     adds r0, r3, #0
00000FEA  F7FF99FF  bl 0x000003EC <__aeabi_fdiv>
00000FEE  1C03     adds r3, r0, #0
00000FF0  2100     movs r1, #0
00000FF2  F2C421C8  movt r1, #0x42C8
00000FF6  1C18     adds r0, r3, #0
00000FF8  F7FF99C  bl 0x00000334 <__aeabi_fmuls>
00000FFC  1C03     adds r3, r0, #0
00000FFE  1C1A     adds r2, r3, #0
00010000  F2400308  movw r3, #8
00010004  F2C20300  movt r3, #0x2000
00010008  601A     str r2, [r3]
-----
main.c -- 411
scratch8++; //for debug breakpoint
0001000A  F2400318  movw r3, #24
0001000E  F2C20300  movt r3, #0x2000
00010012  781B     ldrb r3, [r3]
00010014  3301     adds r3, #1
00010016  B2DA     uxtb r2, r3
00010018  F2400318  movw r3, #24
0001001C  F2C20300  movt r3, #0x2000
00010020  701A     strb r2, [r3]
-----
main.c -- 413
while(1);
00010022  E7FE     b 0x00001022
00010024  078F     lsls r7, r1, #30
00010026  0000     movs r0, r0
-----
<_SEGGER_init_table_>
00010028  104D     asrs r5, r1, #1
0001002A  0000     movs r0, r0
0001002C  0004     movs r4, r0
0001002E  2000     movs r0, #0
00010030  0015     movs r5, r2
00010032  0000     movs r0, r0
00010034  1063     asrs r3, r4, #1
00010036  0000     movs r0, r0
00010038  0000     movs r0, r0
0001003A  2000     movs r0, #0
0001003C  1048     asrs r0, r1, #1
0001003E  0000     movs r0, r0
00010040  0004     movs r4, r0
00010042  0000     movs r0, r0
00010044  0000     movs r0, r0
00010046  0000     movs r0, r0

```

```

int main()
{
    bytesRead--; // last read byte is a throw away
}

return bytesRead;
}

/* MAIN
*****
int main(void)
{
    380  uint8_t rc;

    PORT->Group[0].DIRSET.reg = PORT_PA05;
    PORT->Group[0].OUTCLR.reg = PORT_PA05;
    initI2C(100000);
    enableI2C();
    CONFIGURE_PMUX(SDA);
    CONFIGURE_PMUX(SCL);

    startTransmission(0x40, writeBit);
    sendData(MEASUREMENT_CONFIG);
    390  sendData(0x01);
    setup_cmdI2C(MASTER_ACT_STOP);

    do{
        startTransmission(0x40, writeBit);
        sendData(INTERRUPT_DRDY);

        rc = readDataToBuf(0x40, 1, true);
    }while(! (rxBuf[0] & drdy_status));

    400  startTransmission(0x40, writeBit);
    sendData(TEMP_LOW);

    rc = readDataToBuf(0x40, 4, true);

    tempVal = make16(rxBuf[1], rxBuf[0]);
    humidVal = make16(rxBuf[3], rxBuf[2]);
    tempC = (((float)tempVal/65536)*165)-40;
    tempF = ((float)tempC * 1.8)+32;
    410  humidity = ((float)humidVal/65536)*100;

    412  scratch8++; //for debug breakpoint
    while(1);
}

```

Globals

Expression	Value
humidity	49.9237
humidVal	0x7fce
rxBuf	"\032e\316"
[0]	0x1a
[1]	0x65
[2]	0xce
[3]	0x7f
scratch8	0x00
SystemCoreClock	0x003d0900
tempC	25.1631
tempF	77.2936
tempVal	0x651a

Output

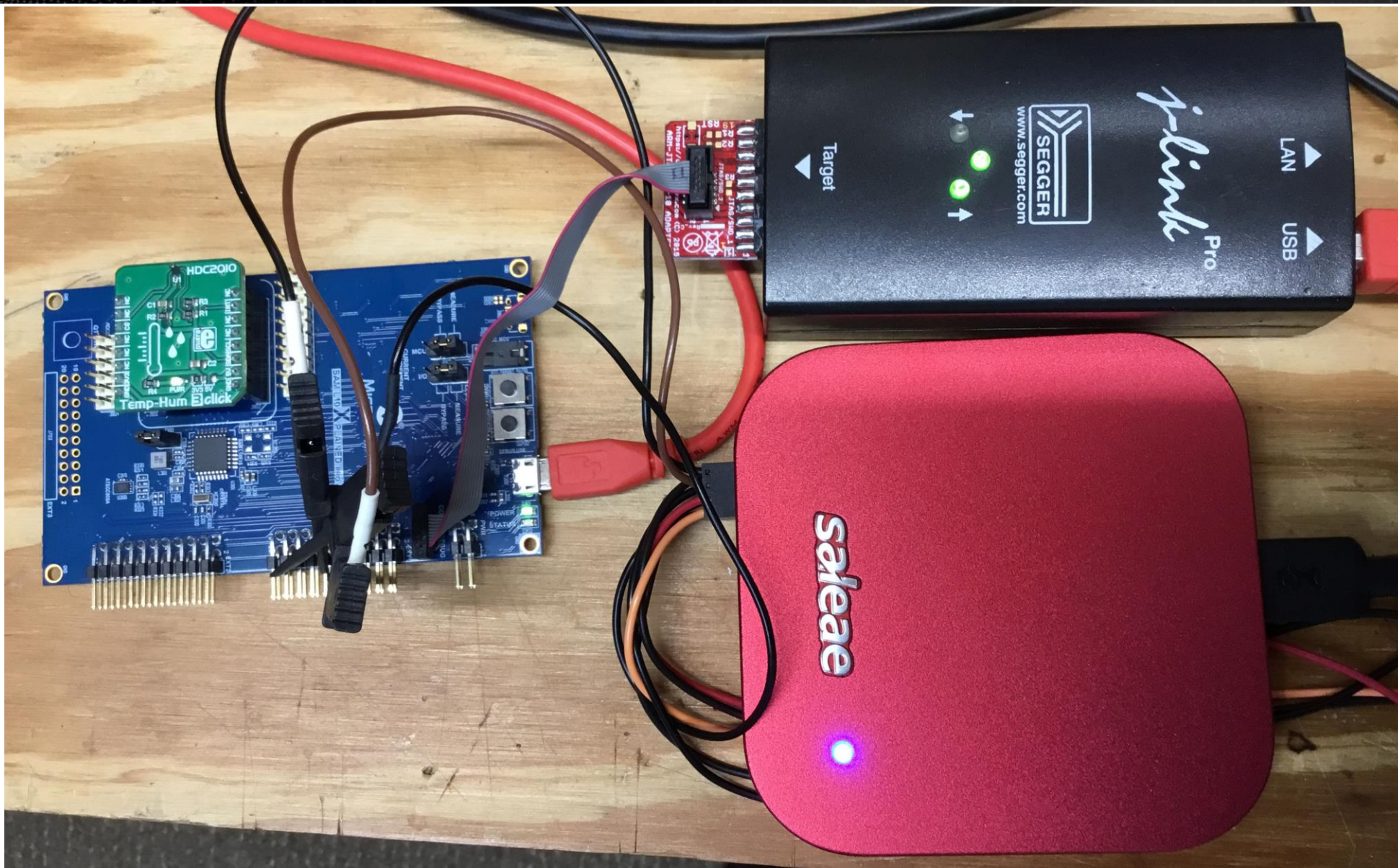
Show: Target

- Loading target script file S4 Working
- Preparing target for download Completed
- Downloading 'saml10Primer' 4.1 KB in 0.2s Download successful 15.6 KB/s

Call Stack

Function

- int main() start()



# Thank you for attending

Please consider the resources below:

- <http://ww1.microchip.com/downloads/en/DeviceDoc/70005359B.pdf>
- <https://www.mikroe.com/temp-hum-3-click>
- <https://www.saleae.com/>



Thank You

Sponsored by

