CEC Continuing Education Center

DesignNews

Embedded Studio Primer

DAY 4 : Microchip SAM Series

Sponsored by

Digi-Key ELECTRONICS

informa markets

Webinar Logistics

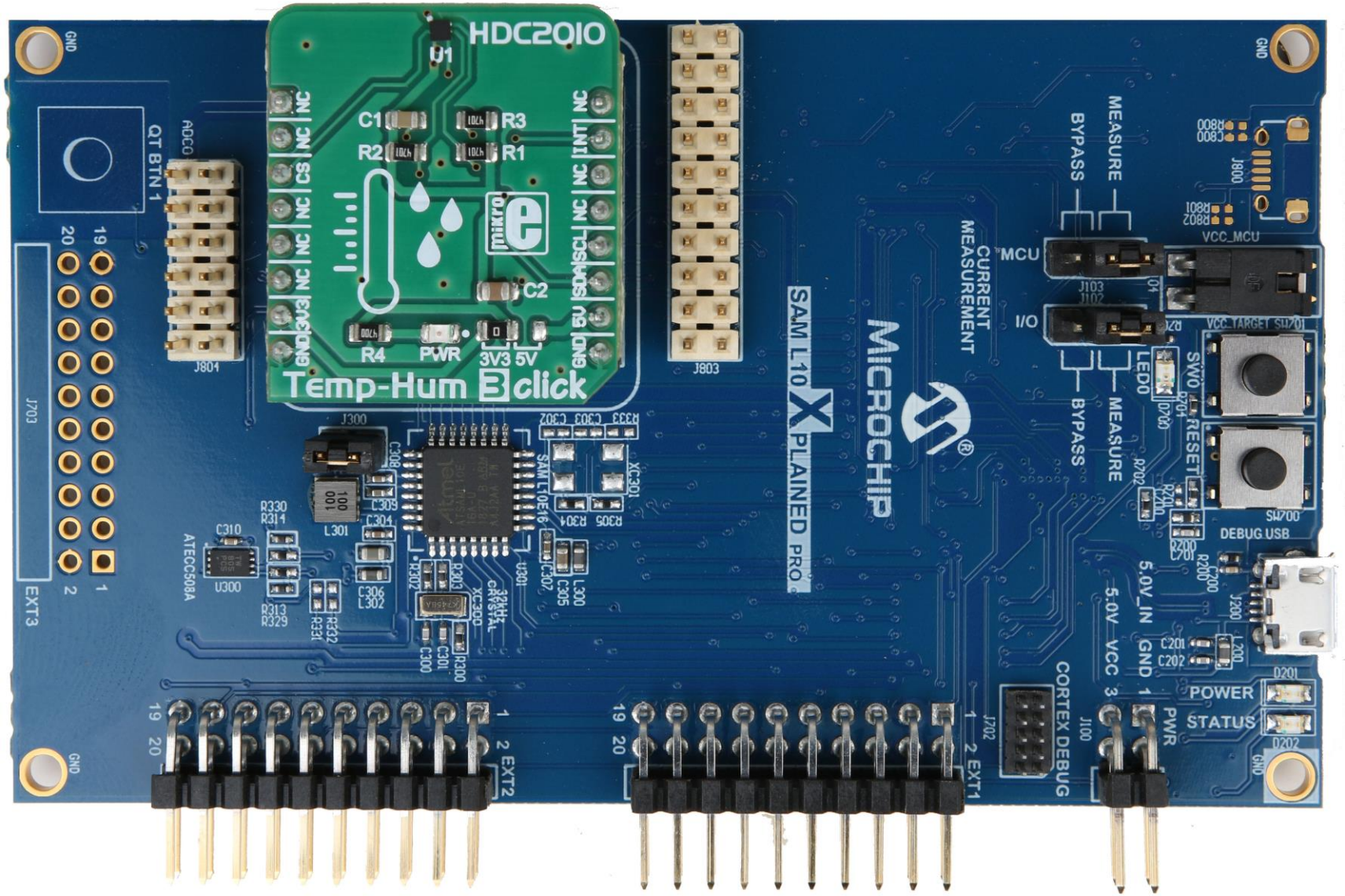
- Turn on your system sound to hear the streaming presentation.
- If you have technical problems, click “Help” or submit a question asking for assistance.
- Participate in ‘Group Chat’ by maximizing the chat widget in your dock.
- Submit questions for the lecturer using the Q&A widget. They will follow-up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

Fred Eady

Visit ‘Lecturer Profile’ in your console for more details.

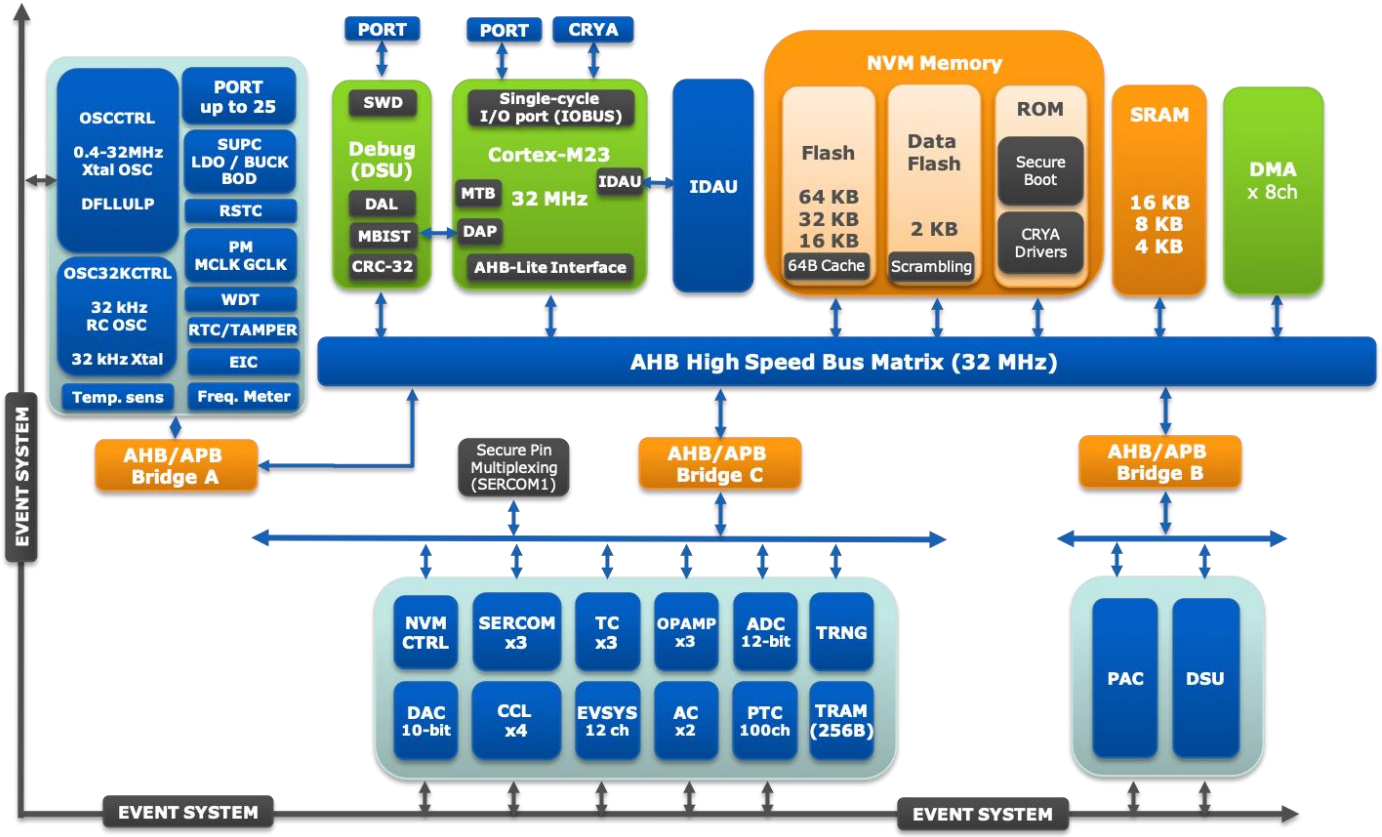
Embedded Studio Primer

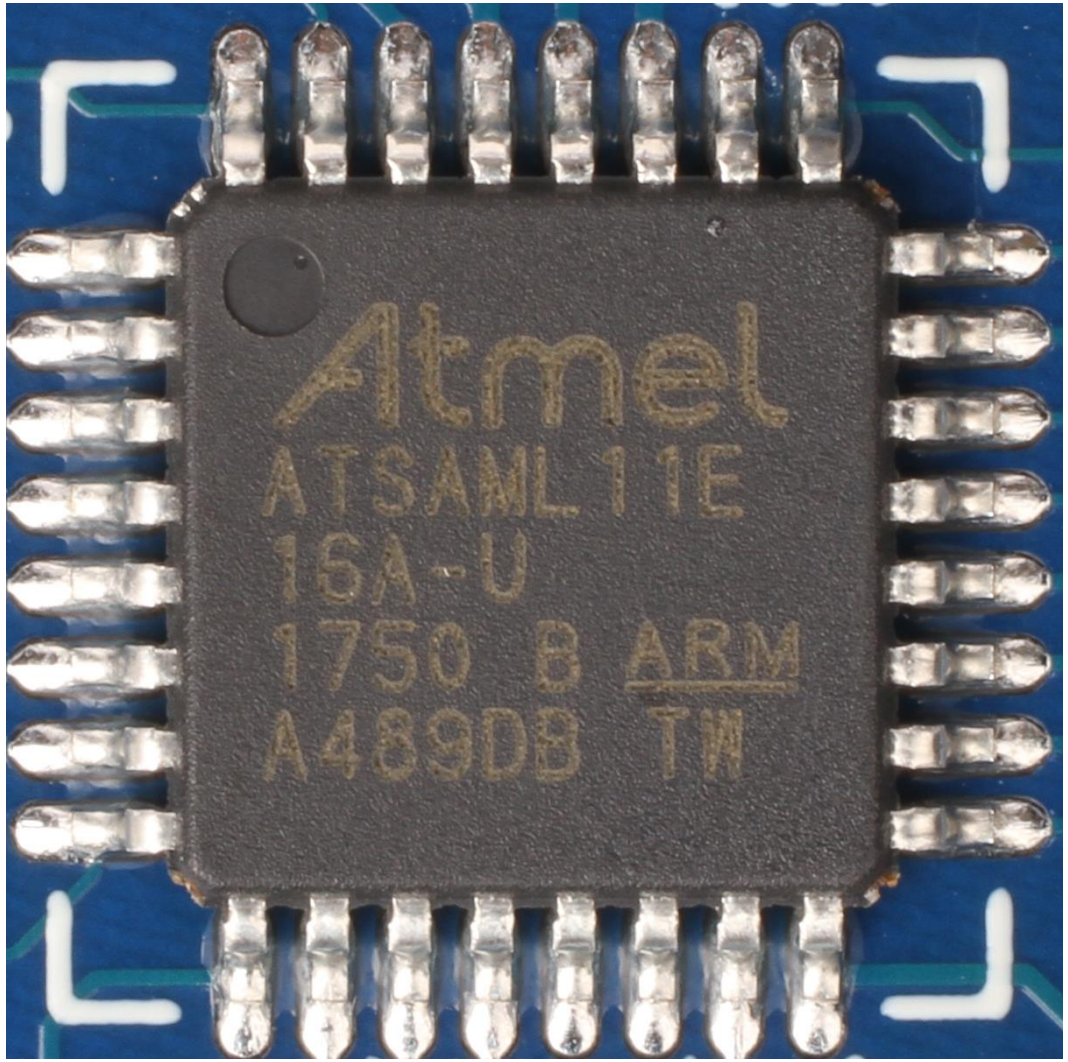
Microchip SAM Series



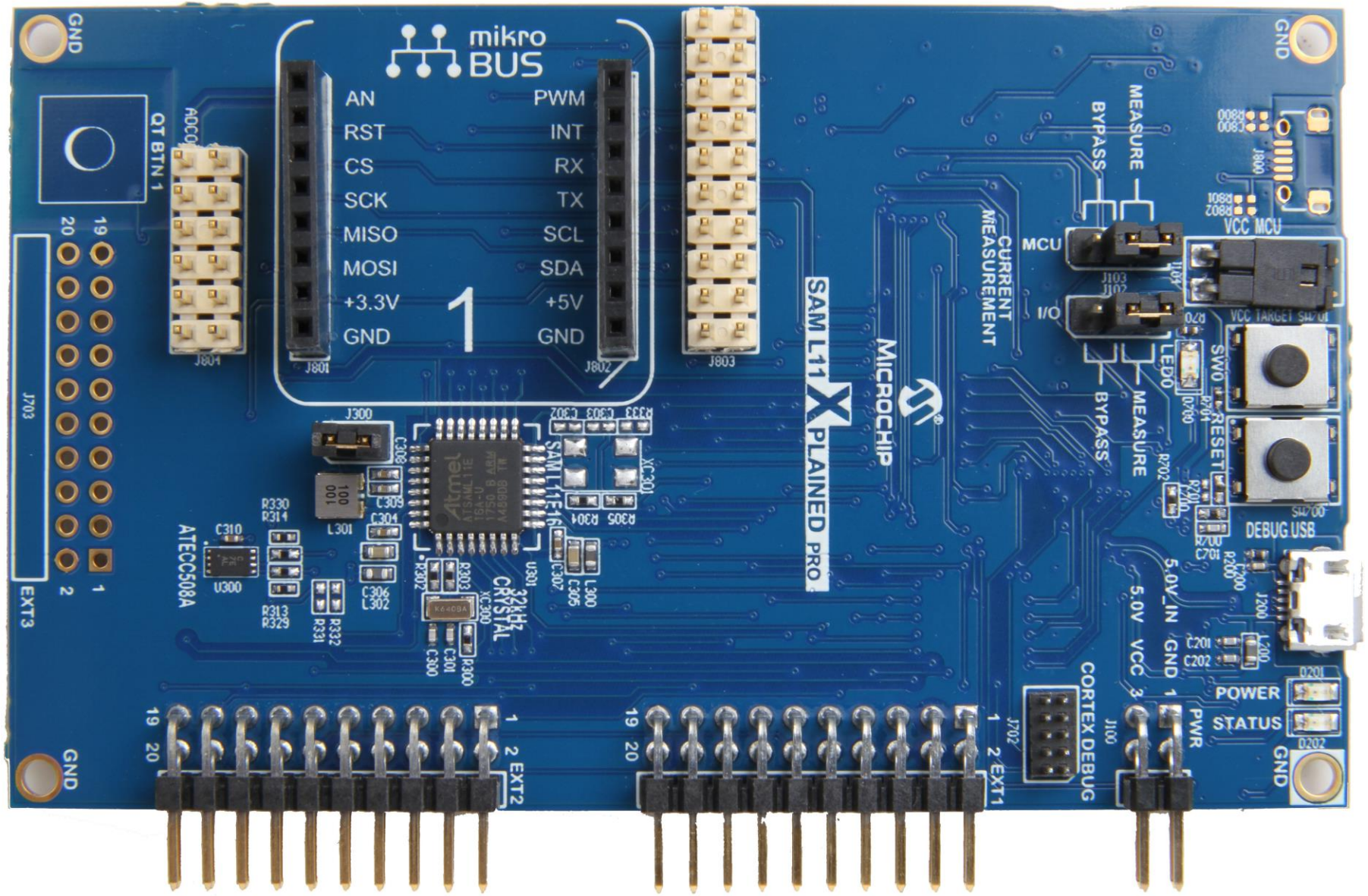
AGENDA

- Programming the SAML11
- Programming the SAML10



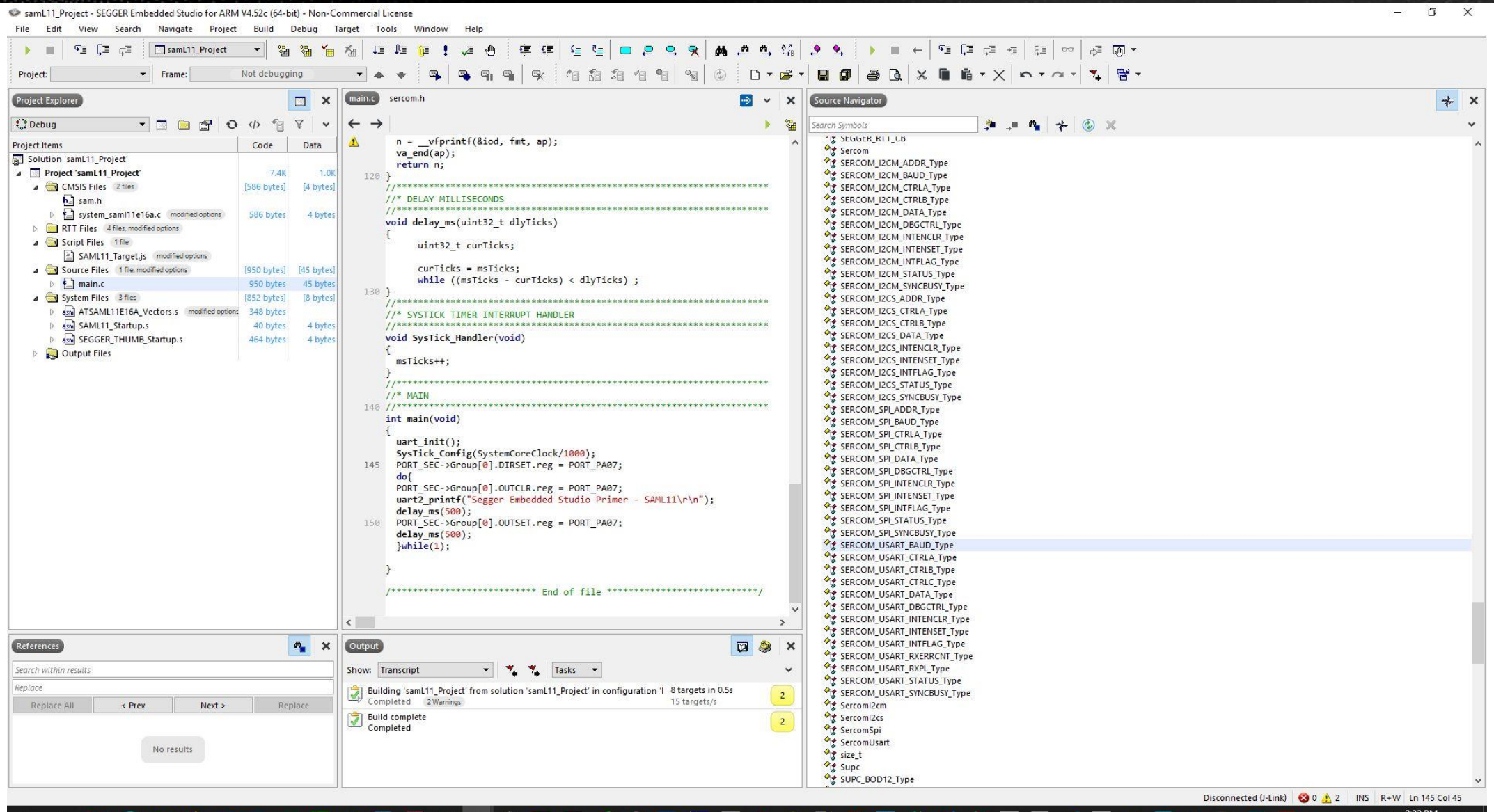


Programming the SAML11

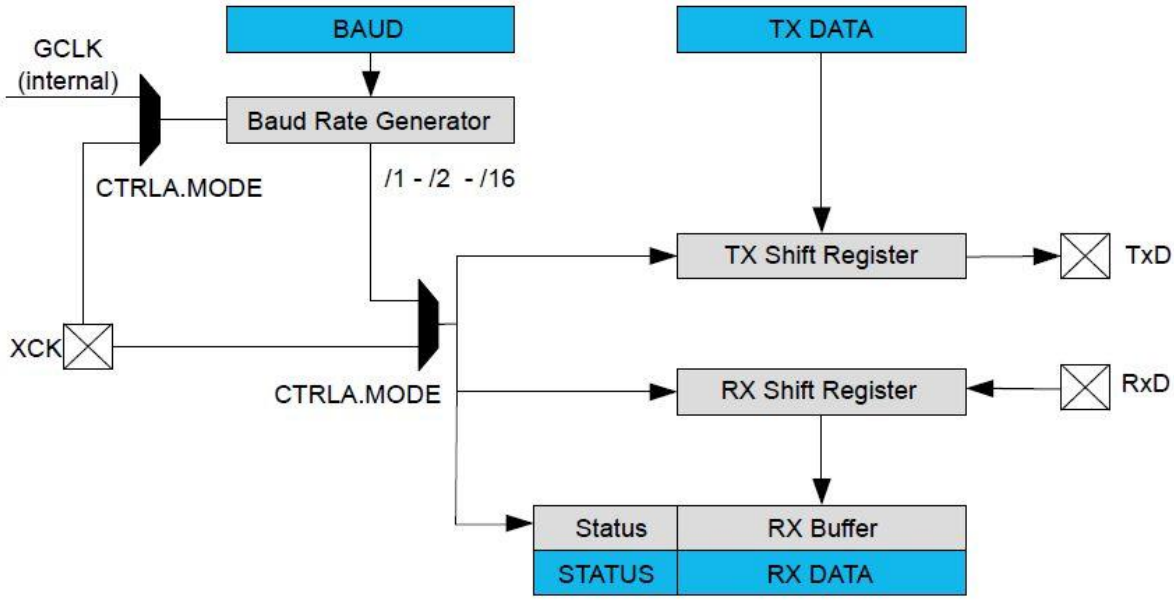


Embedded Studio Primer

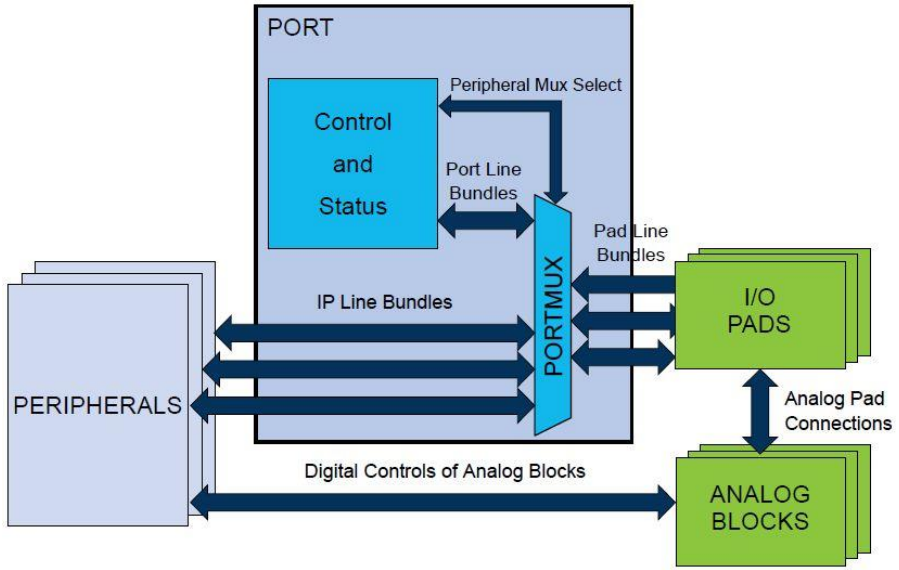
Programming the SAML11



```
//*****************************************************************************
//* 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)
```



```
#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_SAMPR(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;
}
```

```
//************************************************************************
//* 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;
```

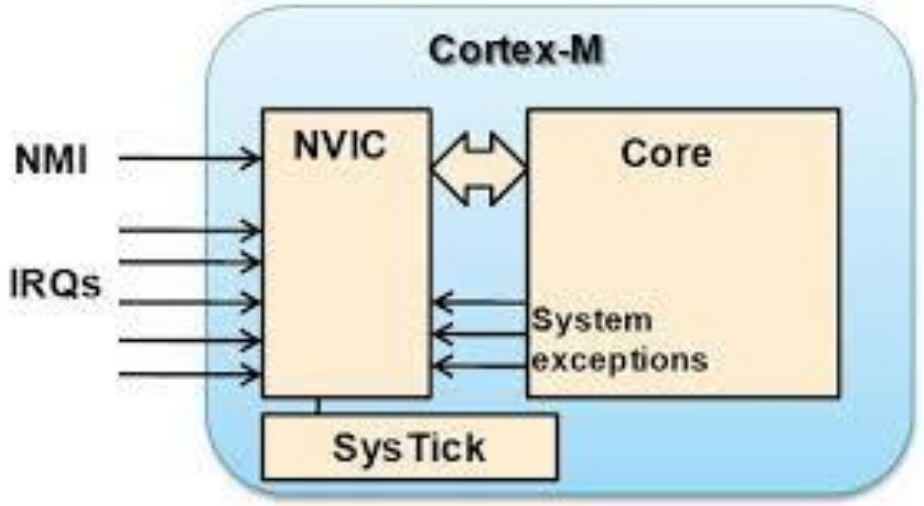
This type is used by the library functions to direct what the formatting routines do with each character they need to output. If `string` is non-zero, the character is appended is appended to the string pointed to by **string**; if `output_fn` is non-zero, the character is output through the function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

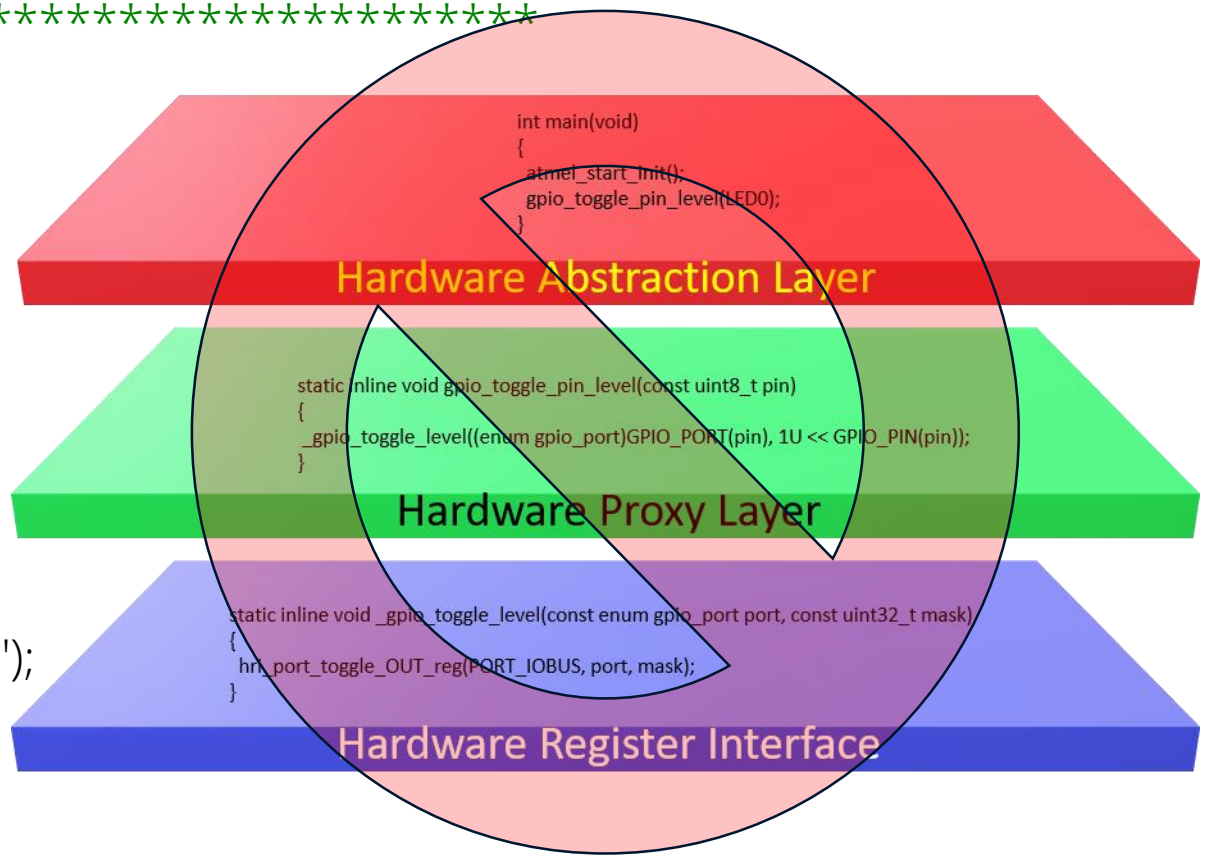
We can use this type and function to rewrite **uart0_printf**:

```
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;
}
```

```
//*****************************************************************
//* 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++;
}
```



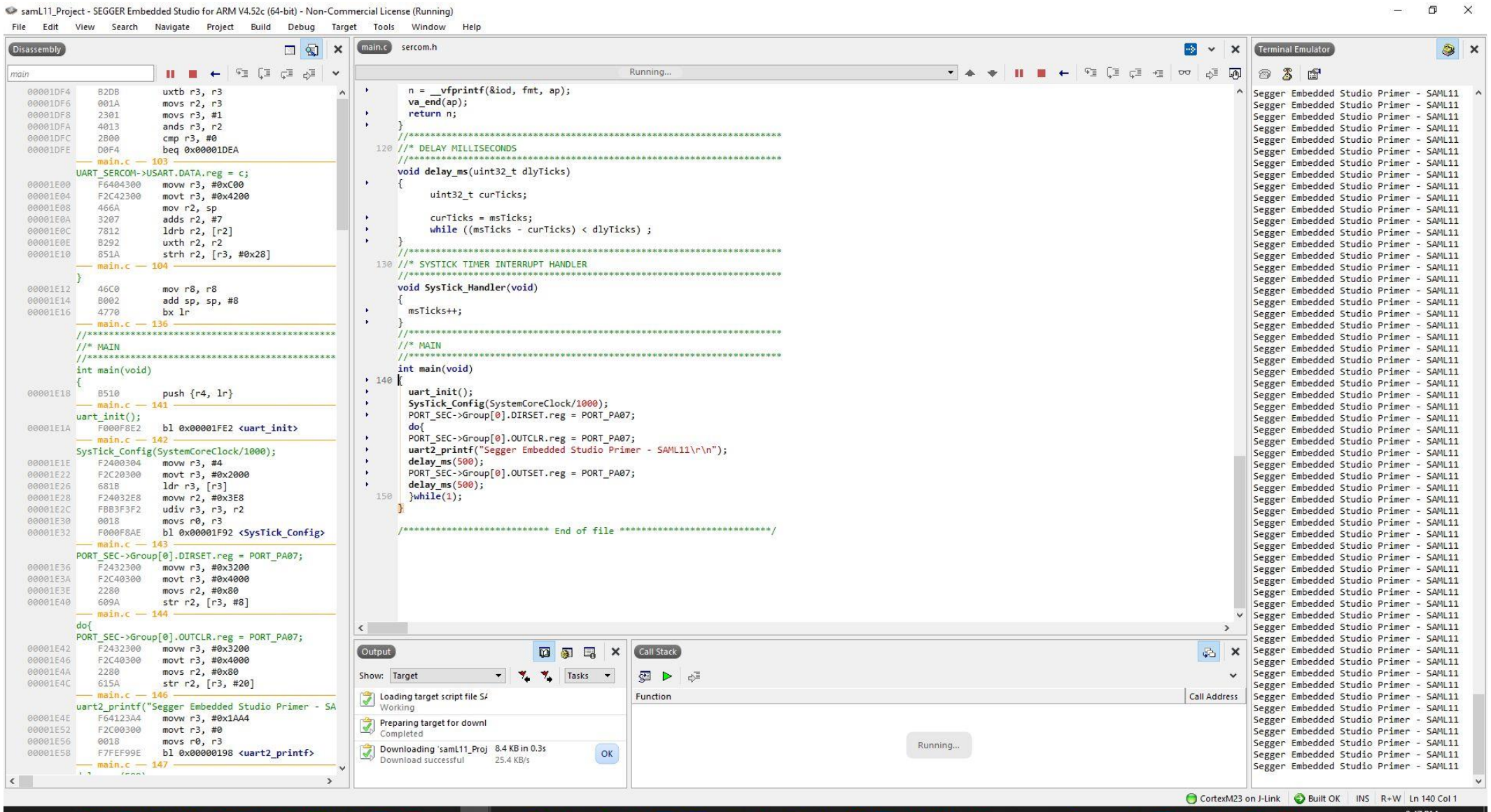
```
//*****************************************************************
//* MAIN
//*****************************************************************
int main(void)
{
  uart_init();

  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);
}
```

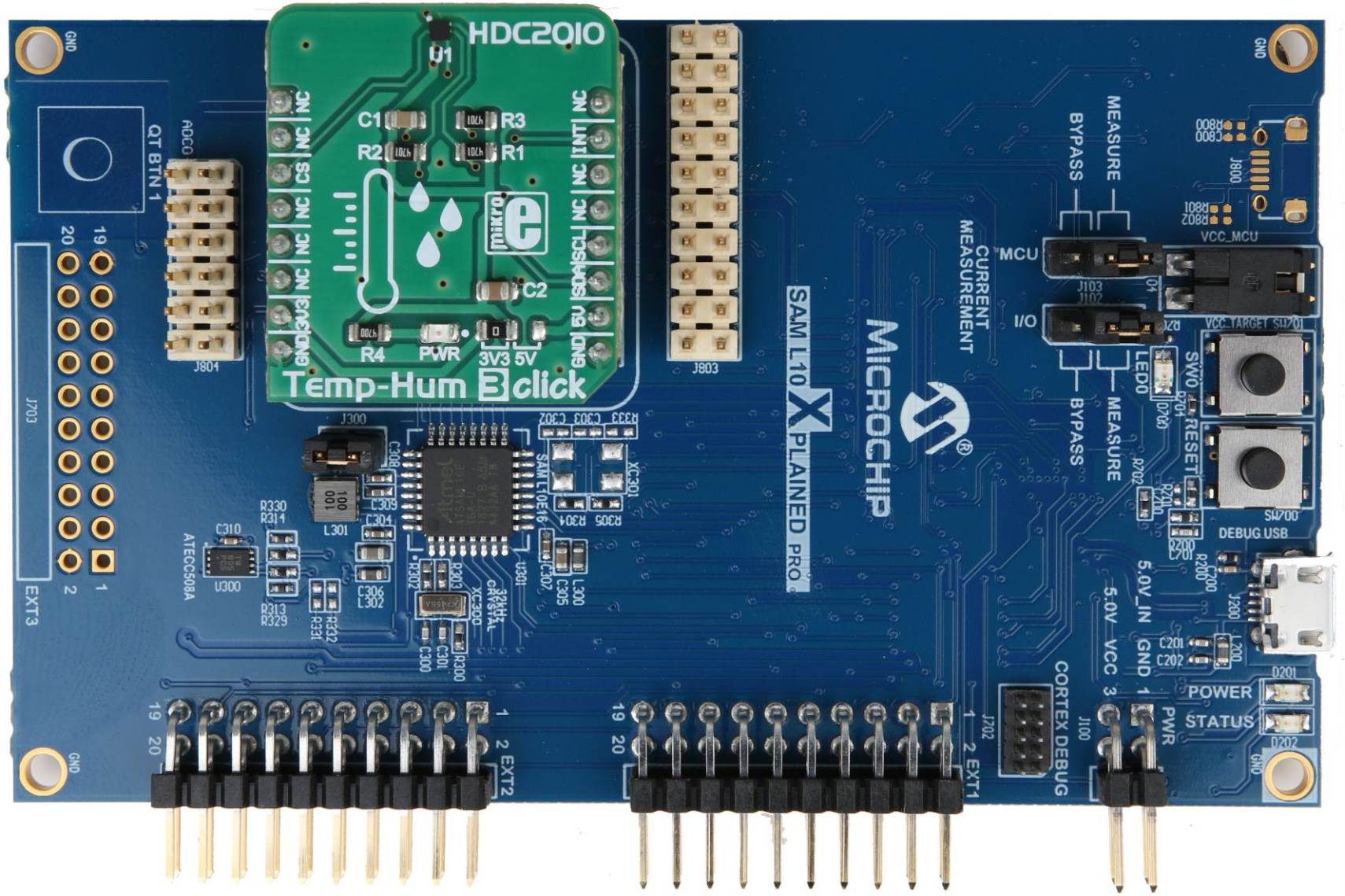


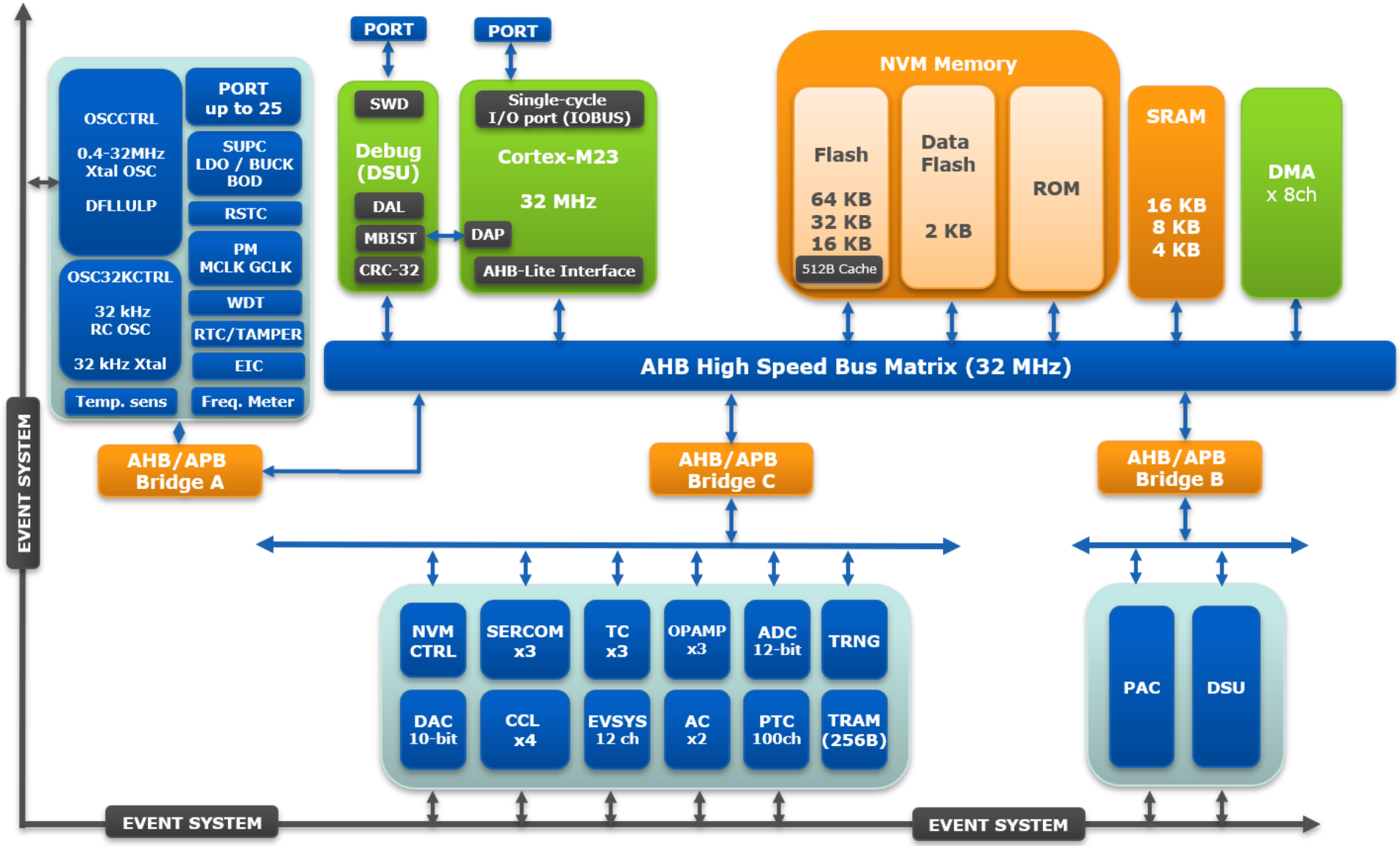
Embedded Studio Primer

Programming the SAML11 – Debugger

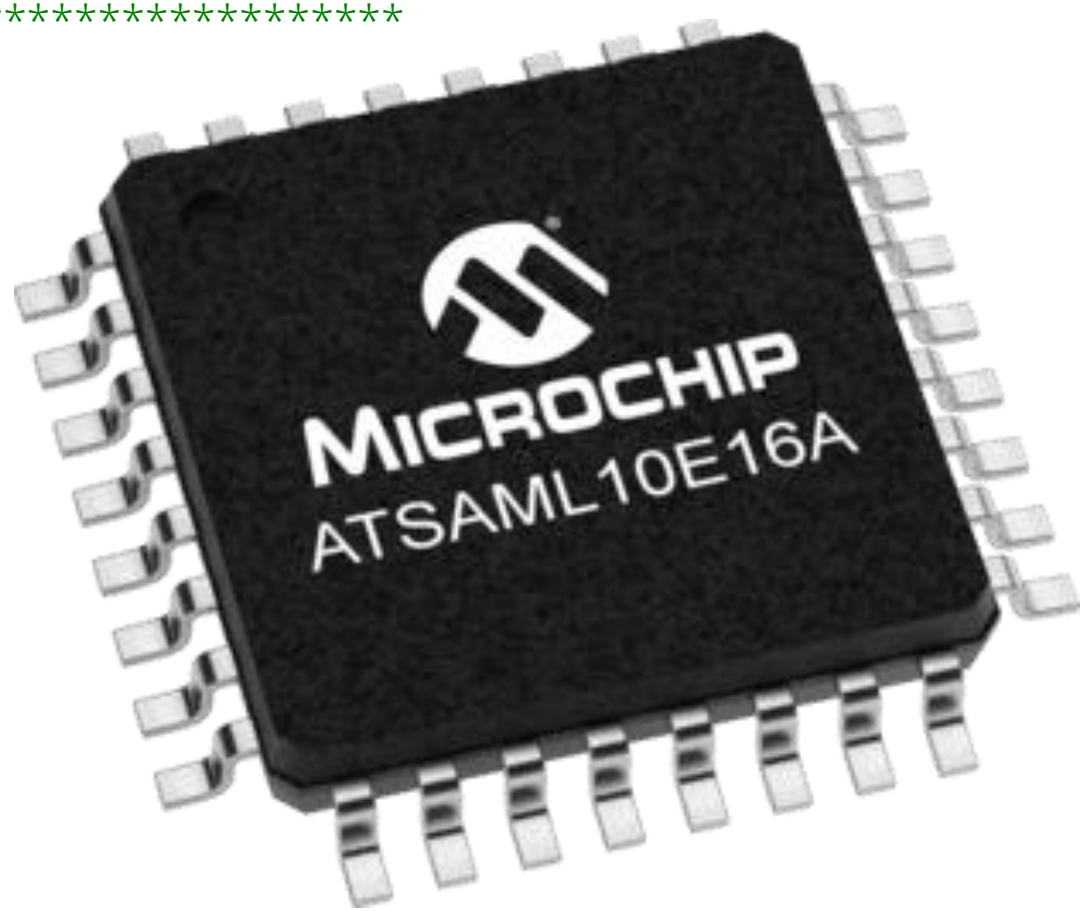
Sponsored By



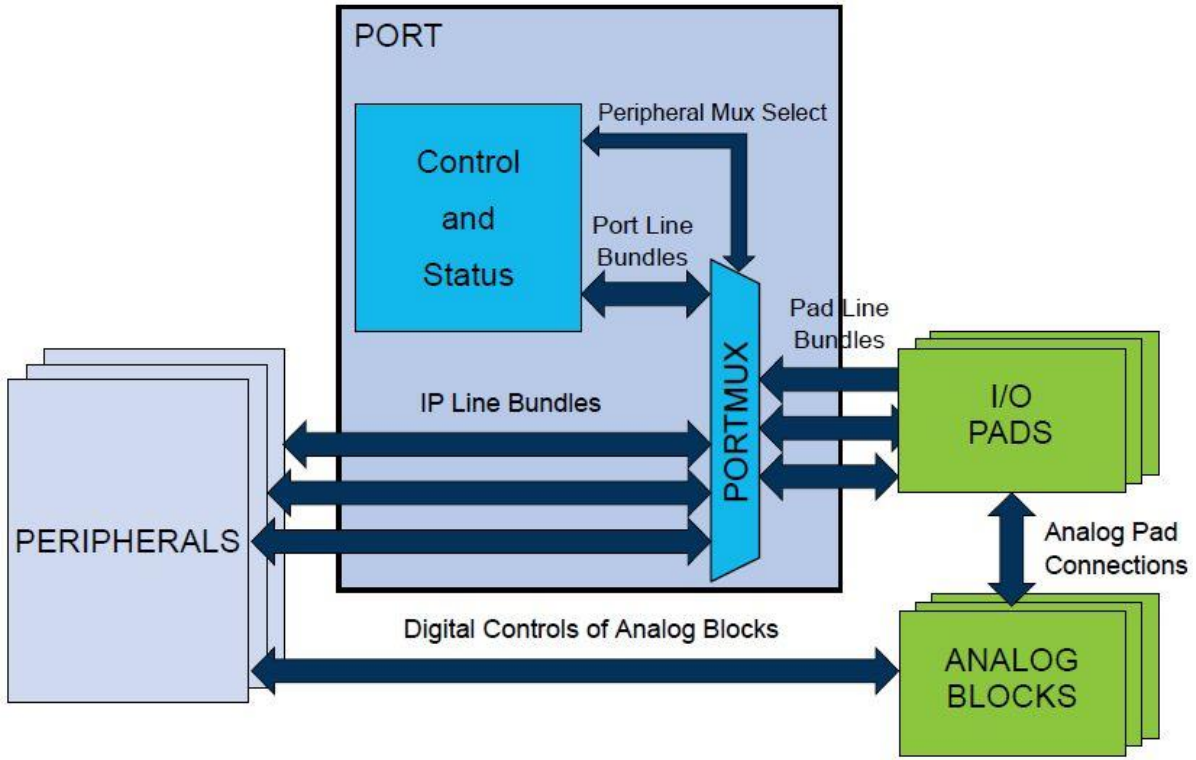

HDC2010
Temp-Hum Click
SAM L10 XPLAINED PRO
MICROCHIP
CURRENT MEASUREMENT
MEASURE
BYPASS
MCU
I/O
VCC_MCU
VCC_TARGET
SW0
RESET
LED0
DEBUG USB
5.0V_IN
5.0V_VCC
GND
POWER
STATUS
CORTEX DEBUG
QT BTN 1
EXT1
EXT2
EXT3
ATECC508A
SAML10E16


PORT
PORT
PORT up to 25
OSCCTRL
0.4-32MHz Xtal OSC
DFLLULP
SUPC
LDO / BUCK
BOD
RSTC
PM
MCLK GCLK
OSC32KCTRL
32 kHz RC OSC
32 kHz Xtal
WDT
RTC/TAMPER
EIC
Temp. sens
Freq. Meter
SWD
Debug (DSU)
DAL
MBIST
CRC-32
DAP
Single-cycle I/O port (IOBUS)
Cortex-M23
32 MHz
AHB-Lite Interface
NVM Memory
Flash
64 KB
32 KB
16 KB
512B Cache
Data Flash
2 KB
ROM
SRAM
16 KB
8 KB
4 KB
DMA
x 8ch
AHB High Speed Bus Matrix (32 MHz)
EVENT SYSTEM
AHB/APB Bridge A
AHB/APB Bridge C
AHB/APB Bridge B
NVM CTRL
SERCOM x3
TC x3
OPAMP x3
ADC 12-bit
TRNG
DAC 10-bit
CCL x4
EVSYS 12 ch
AC x2
PTC 100ch
TRAM (256B)
PAC
DSU
EVENT SYSTEM
EVENT SYSTEM

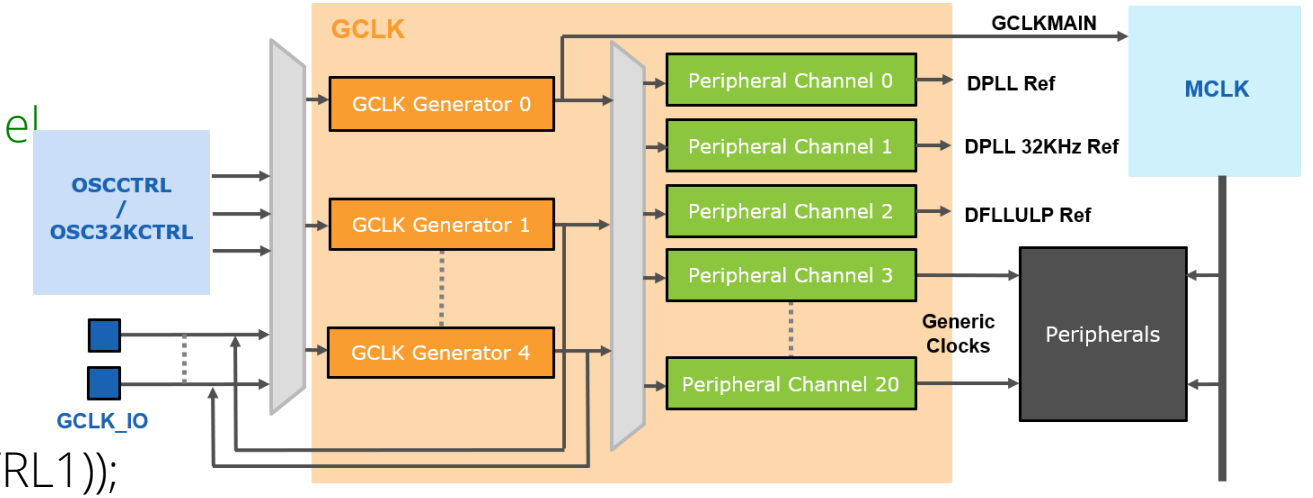
```
//*****************************************************************
//* FUNCTION PROTOTYPES
//*****************************************************************
void resetI2C(void);
void enableI2C(void);
void disableI2C(void);
void initClockI2C(void);
void initI2C(uint32_t baudrate);
bool startTransmission(uint8_t 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);
```



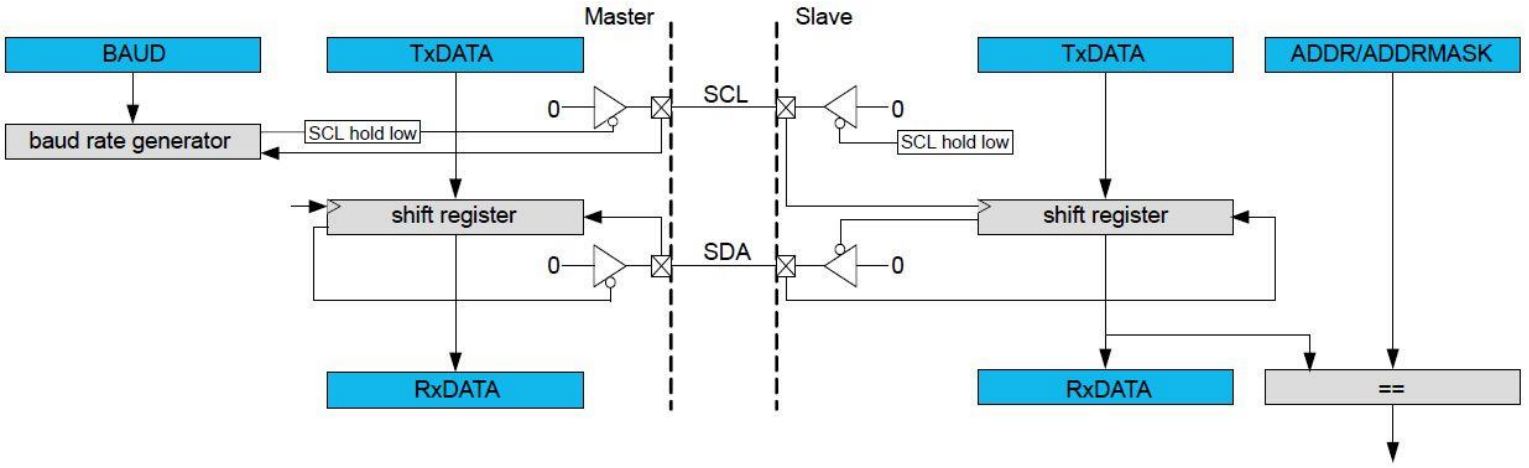
```
//*************************************************************************
//* SAML10 I2C PINS AND PORTS
//*************************************************************************
#define I2C_SDA_PORT      0 // PORTA IN GROUP 0
#define I2C_SDA_PIN       PIN_PA16C_SERCOM1_PAD0
#define I2C_SDA_MUX       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
```



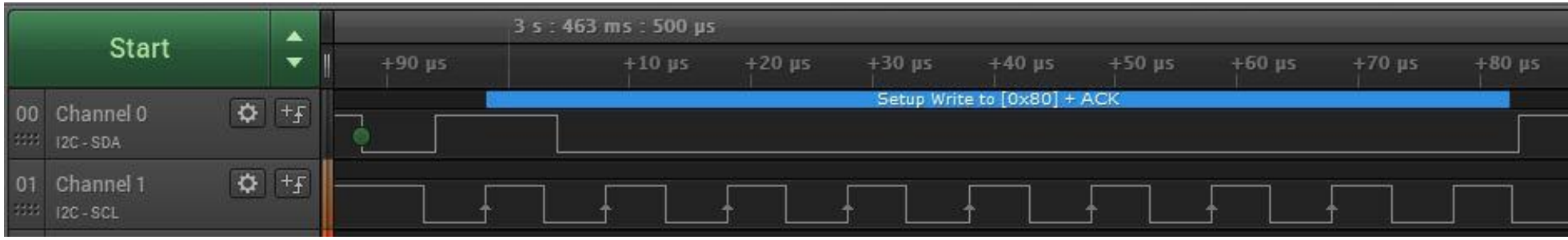
```
//*********************************************************************
//* I2C INIT CLOCK
//*********************************************************************
void initClockI2C(void)
{
  //Set clock generator, clock source and peripheral channel
  GCLK->GENCTRL[1].reg =  GCLK_GENCTRL_DIV(1)     |
                          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);
}
```



```
#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));
}
```

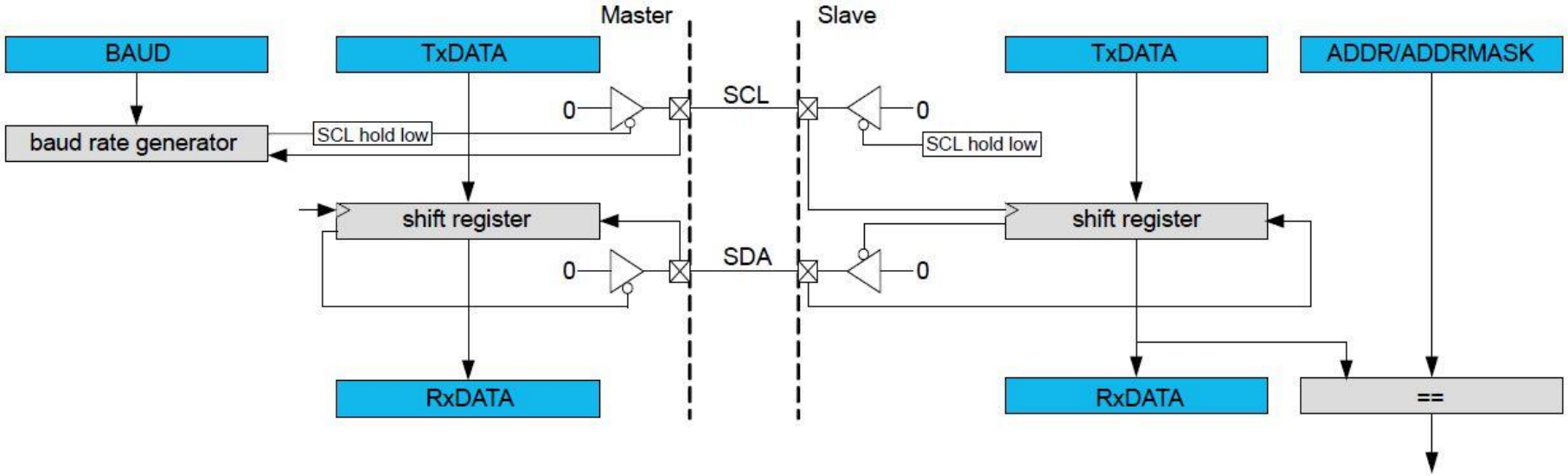


```
//*****************************************************************************
//* I2C START TRANSMISSION
//*****************************************************************************
bool startTransmission(uint8_t addr, READ_WRITE_BIT rwBit)
{
  // 7-bits address + rwBit
  addr = (addr << 0x01) | rwBit;

  if(!isBusOwner())
  {
    if( isBusBusy() || (isArbLost() && !isBusIdle()) )
    {
      return false;
    }
  }

  // Send start and address
  SERCOM1->I2CM.ADDR.bit.ADDR = addr;
```

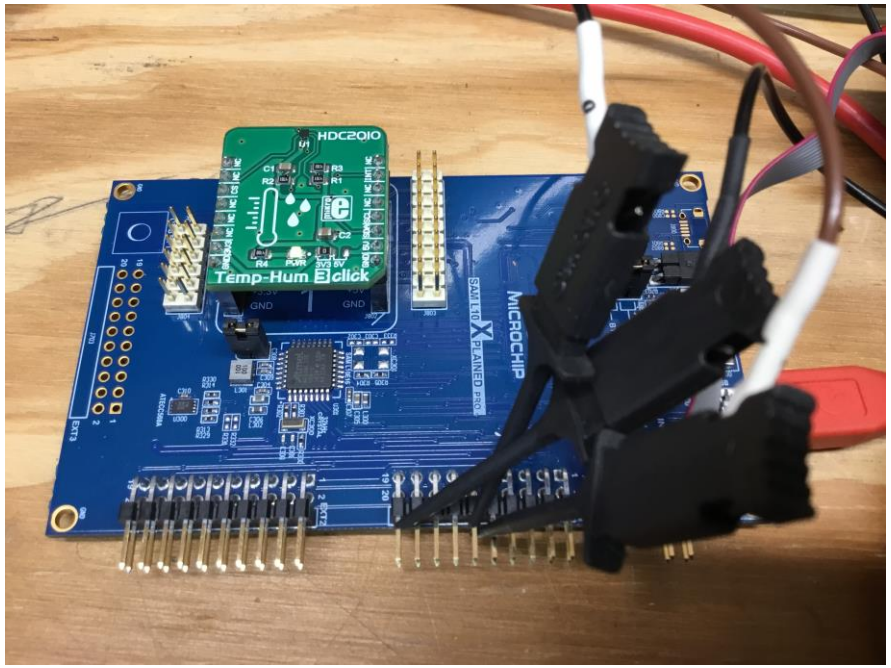


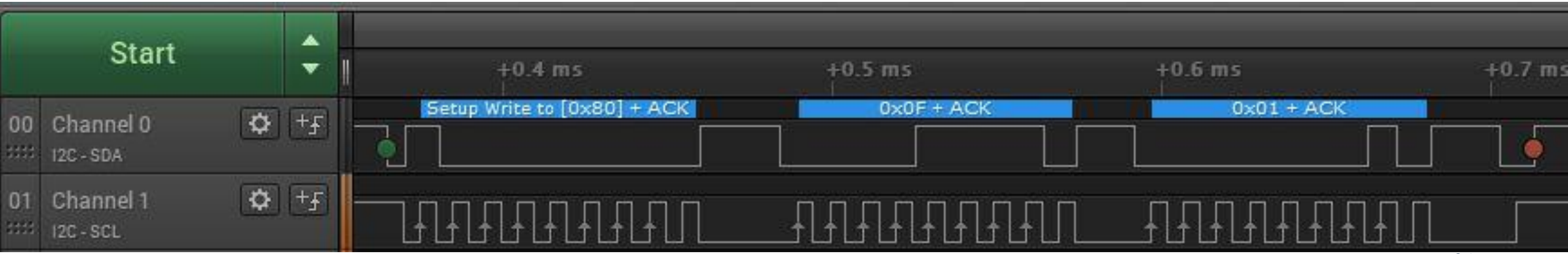
```
// Address Transmitted
 if ( rwBit == writeBit ) // Write mode
 {
   // Wait for transmission to complete
   while( !SERCOM1->I2CM.INTFLAG.bit.MB );
   // Check for loss of arbitration (multiple masters starting communication at the same time)
   if(!isBusOwner())
   {
     // Restart communication
     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;
      }
   }
 }
```

```
//*****************************************************************
//* 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);
```

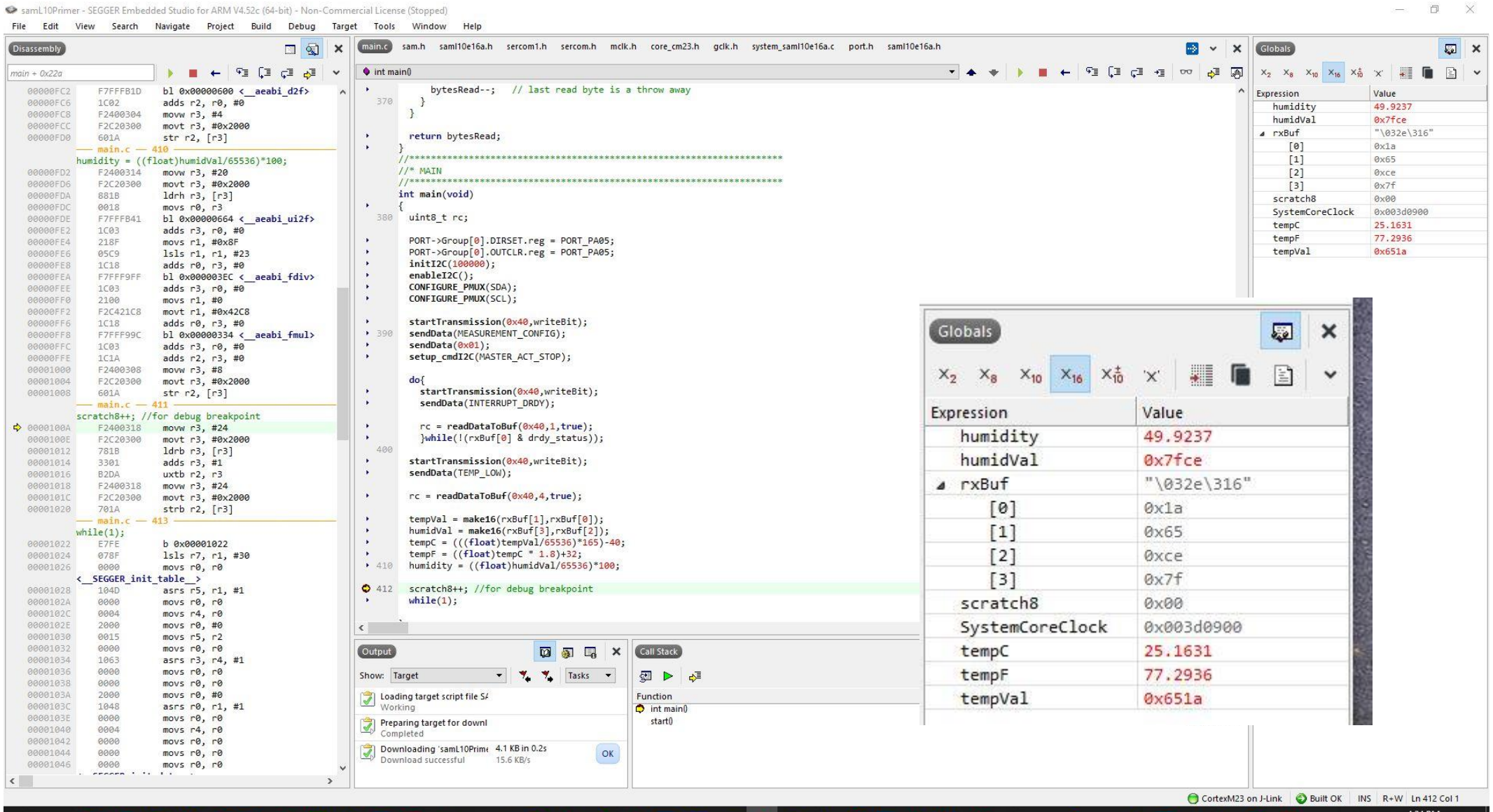


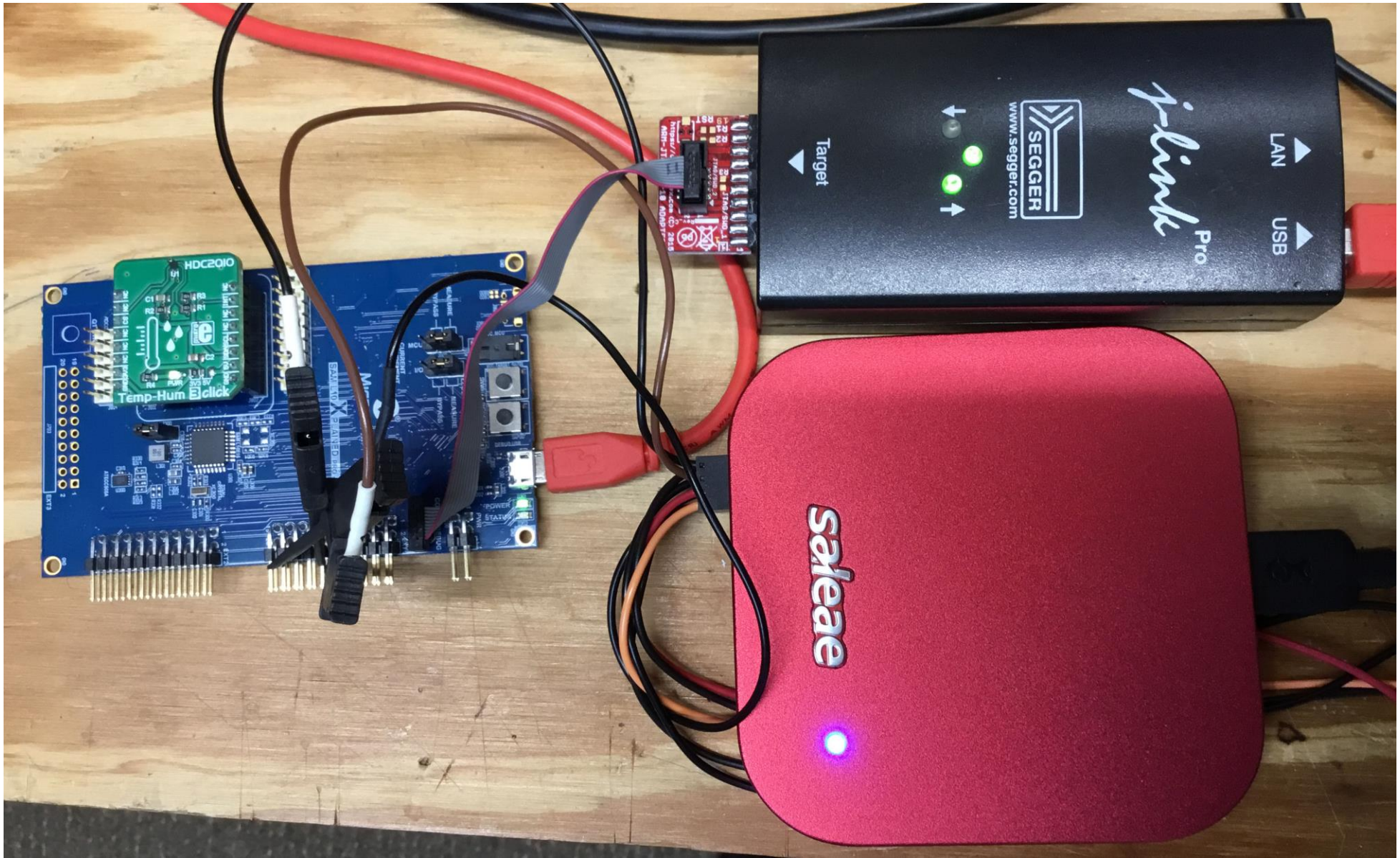


Embedded Studio Primer

Programming the SAML10 – Get Temp/Humidity

Sponsored By




Target
LAN
USB
J-Link
Pro
SEGGER
www.segger.com
HDC2010
Temp-Hum Click
saleae

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/

CEC Continuing Education Center

DesignNews

Thank You

Sponsored by

Digi-Key ELECTRONICS

informa markets