CEC Continuing Education Center

DesignNews

Techniques for Interfacing with Modern Sensors

DAY 4 : Sensor Driver Techniques Part 2

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.

THE SPEAKER

Jacob Beningo

Visit ‘Lecturer Profile’

Beningo Embedded Group - President

Focus: Embedded Software Consulting

An independent consultant who specializes in the design of real-time, microcontroller based embedded software.
He has published two books:

- Reusable Firmware Development
- MicroPython Projects

Writes a weekly blog for DesignNews.com focused on embedded system design techniques and challenges.

Visit www.beningo.com to learn more …

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

Course Sessions

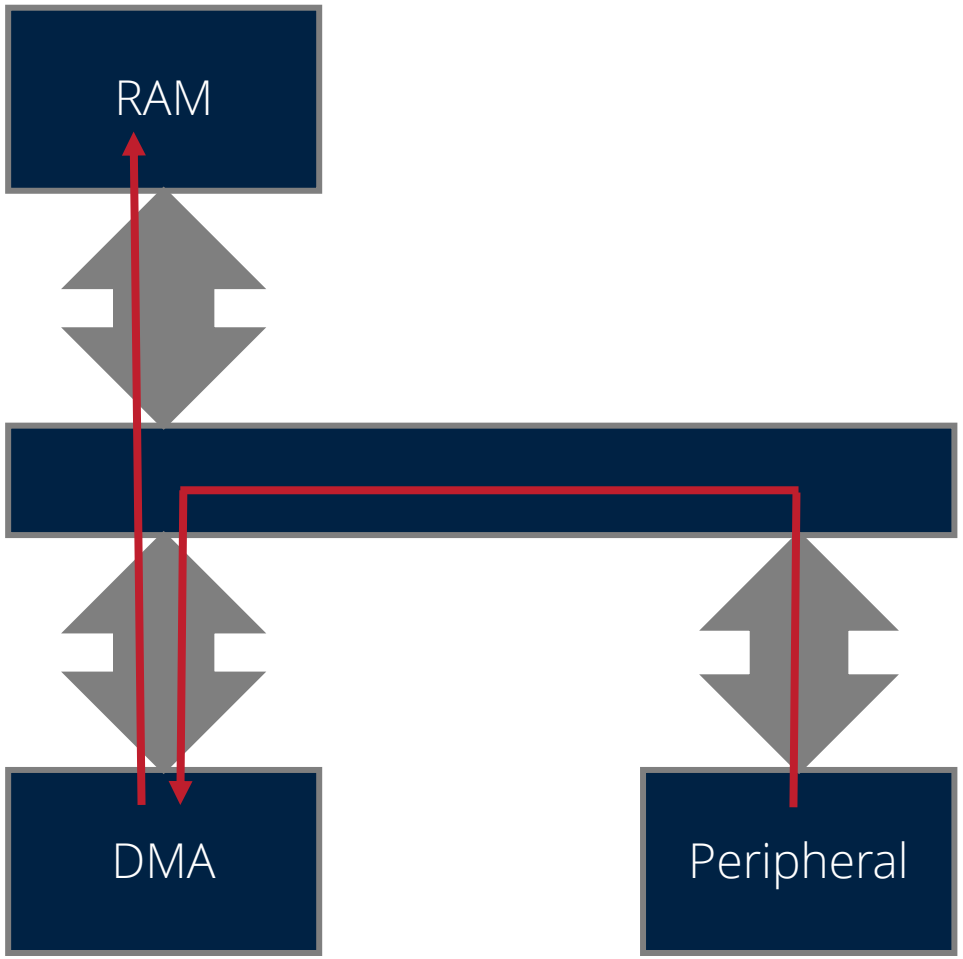
- Introduction to Modern Sensor Interfacing
- Designing Sensor Interfaces
- Sensor Driver Techniques Part 1
- Sensor Driver Techniques Part 2
- Leveraging C++ in Sensor Interfacing

Where are we at?

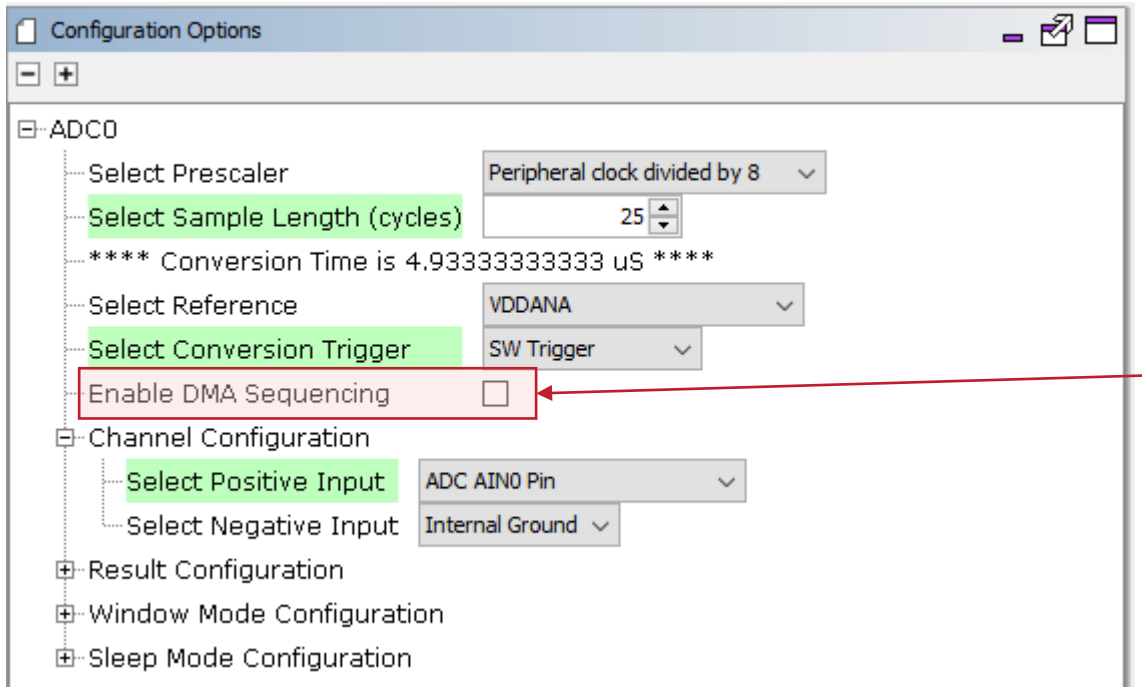
- Polling
- Interrupt
- DMA
- Other

Technique	Complexity	Efficiency
Polling	Low	Low
Interrupt	Medium	Medium
DMA Driven	Medium	High

Technique #3 - Direct Memory Access



Example

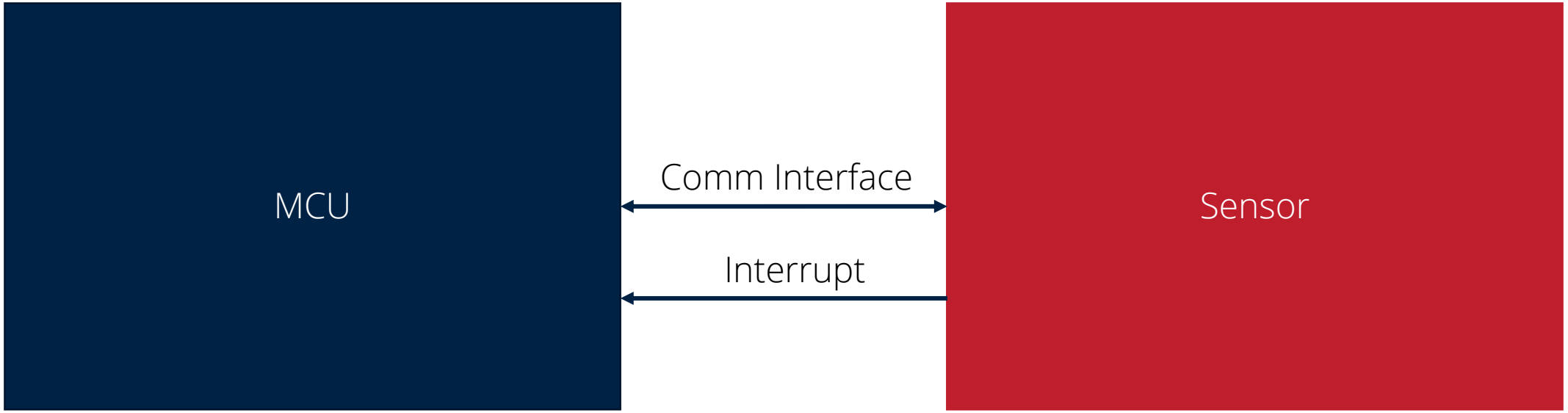


"Enable DMA"

DMA Channels are often limited in MCU's. When do you prefer to use a DMA transfer?

- Any time I can
- For activities that require high-speed data transfer
- Activities that interrupt the processor frequently
- Other

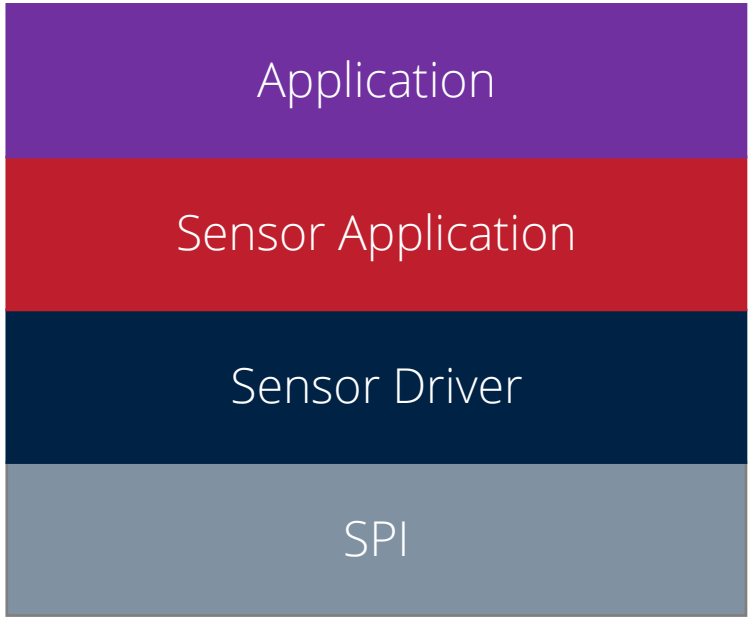
External Digital Sensor Interfacing



What driver pattern is best used for the sensor interface on the previous page:

- Polled
- Interrupt driven
- DMA
- Not sure yet

The recommended software stack



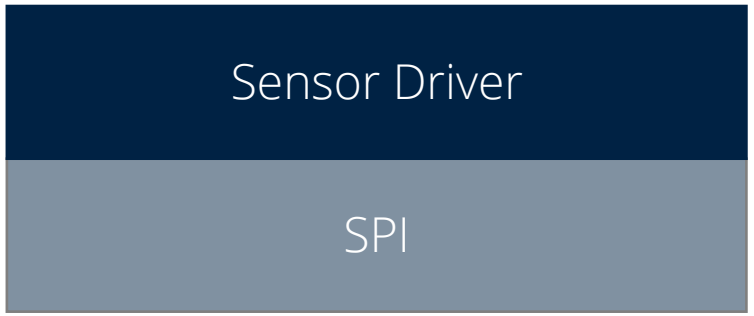
Example External Sensor on SPI Bus

```
void SpiTransfer(void)
{
// Pull the slave line low
    GPIO_SS_Clear();

    // Send and receive the data
    SERCOM1_SPI_WriteRead(&TxBuffer, TxSize, &RxBuffer[0], RxSize);

    // Wait until the transfer is complete.
    while (SERCOM1_SPI_IsBusy() == true)
    {
        // Wait here till the transfer is done.
    }

    // Clear the chip select line
   GPIO_SS_Set();

}
```

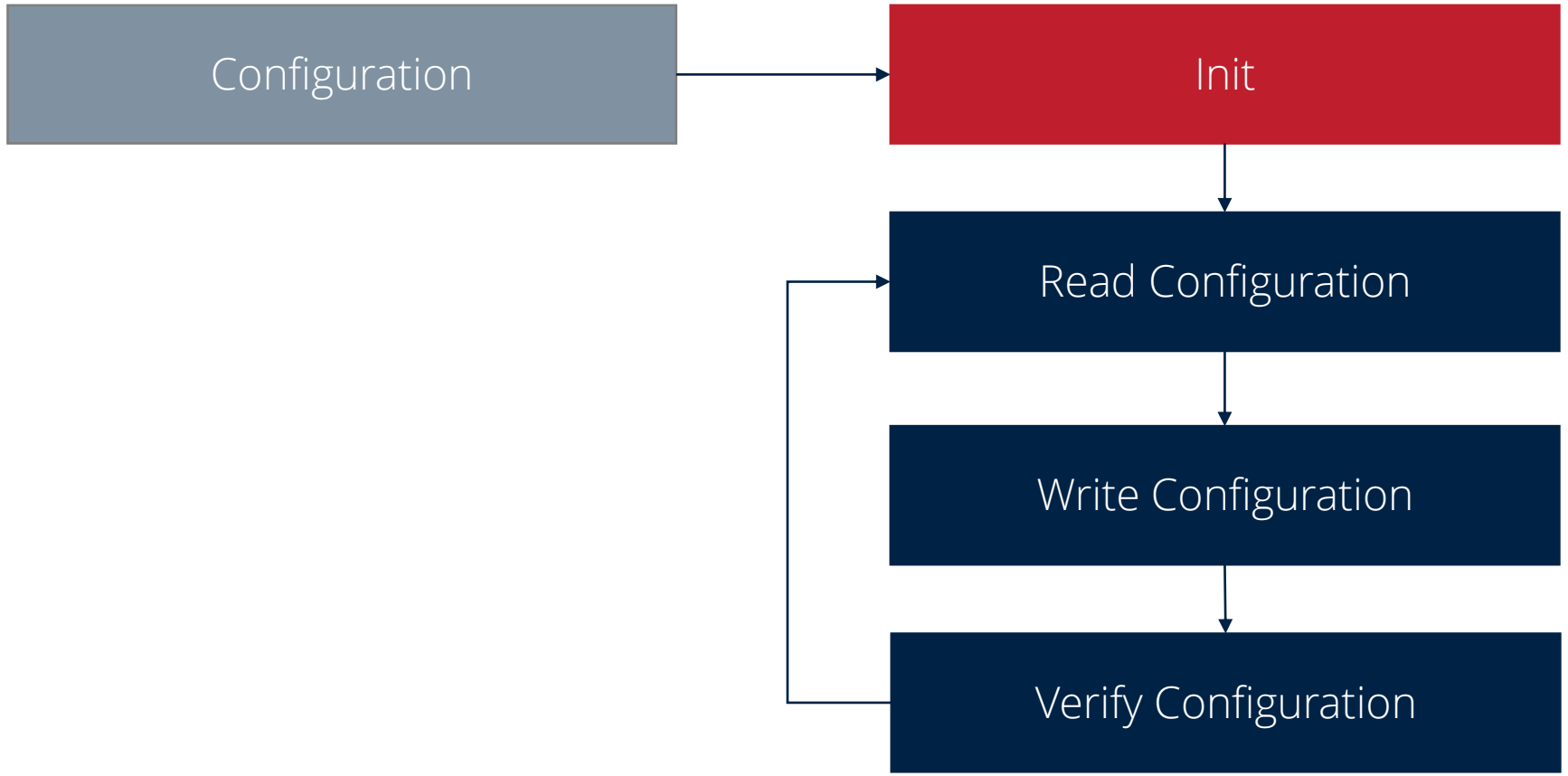
SPI

The Sensor Driver

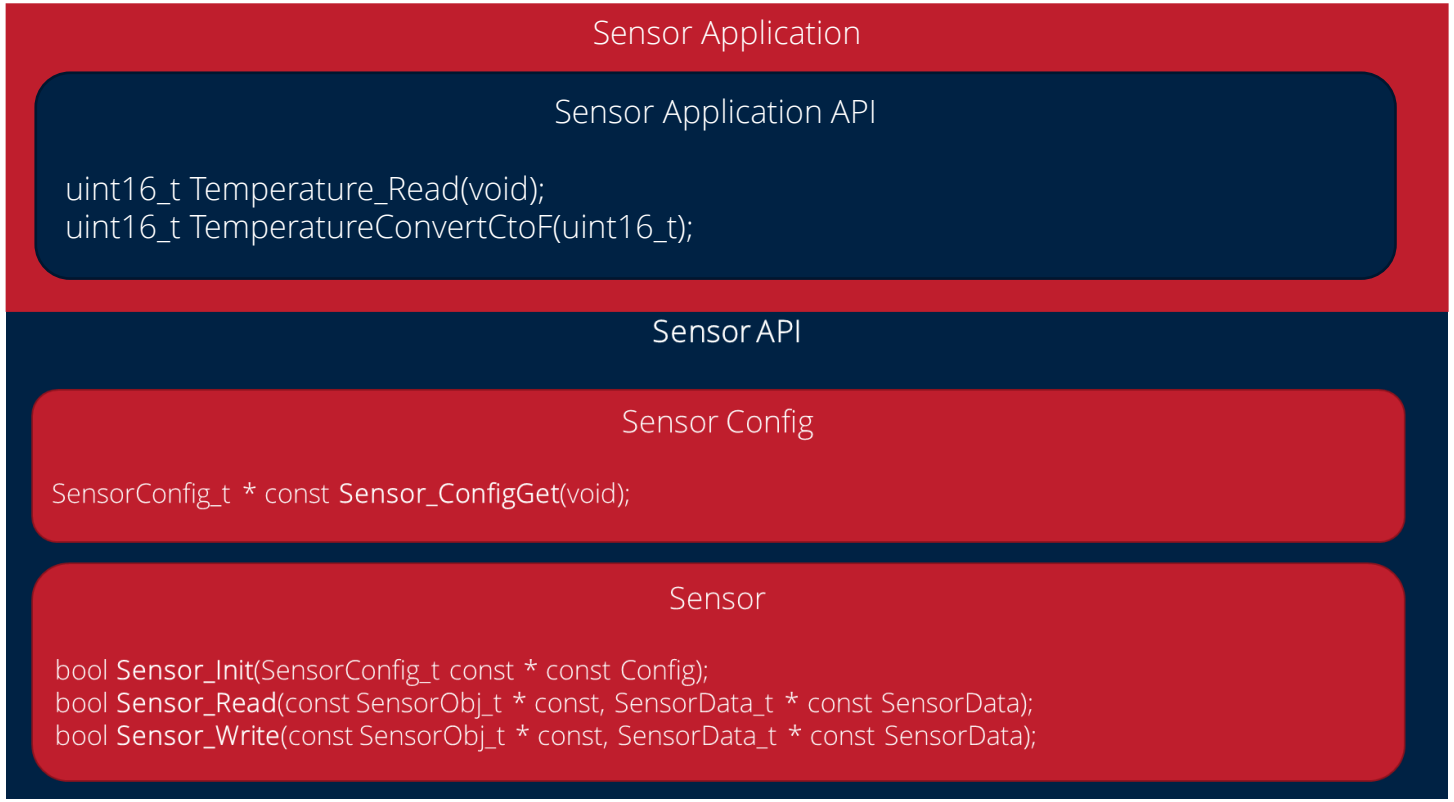
```
bool Init(Config_t const * const  Config);
void RegisterWrite(Reg_t const Register, uint16_t const Data);
bool RegisterRead(Reg_t const Register, uint16_t * const Result);
void RegisterCallBack(void * (Func)(void));
```

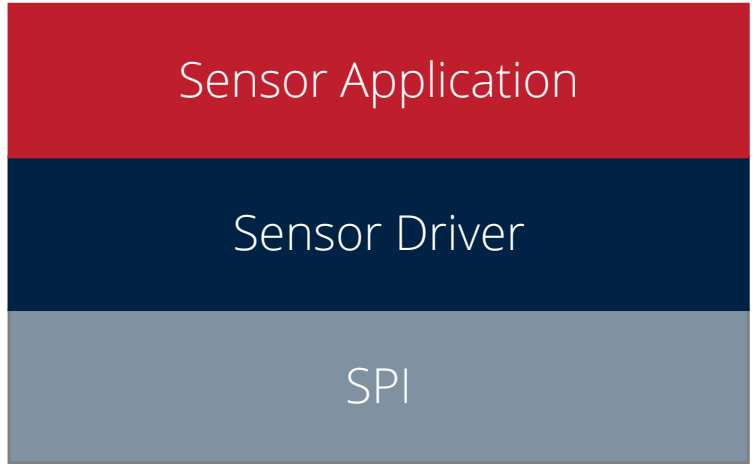


The Sensor Driver – Init Pattern



Sensor Application APIs

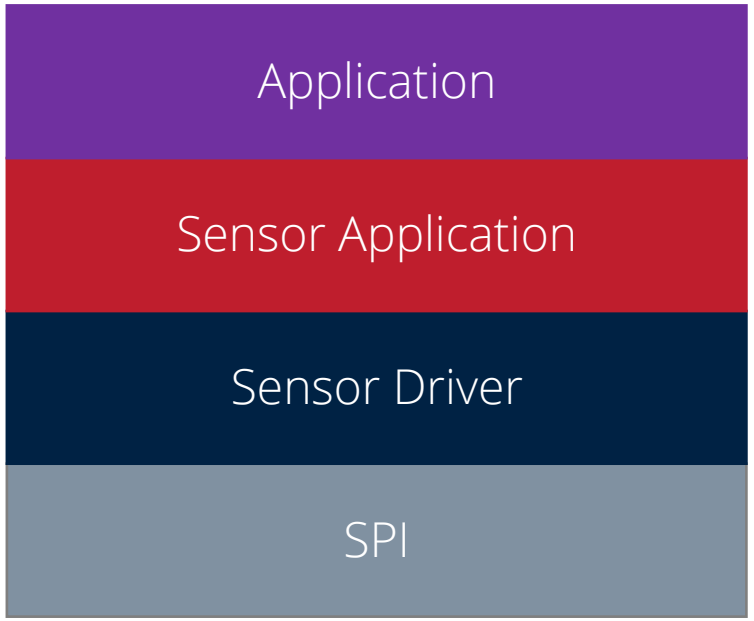




The Product Application

Application Task

```
void Task_Sensor(void)
{
    SensorConfig_t * Config = Sensor_ConfigGet();

    Sensor_Init(Config);

    for(;;)
    {
        Sensor_AppRun();
        vTaskDelayUntil(TASK_SENSOR_PERIOD);
    }
}
```



Is it obvious now how these sensor techniques can be used to interface sensor in a configurable, scalable and flexible manner?

- Yes
- No
- Mostly

Thank you for attending

Please consider the resources below:

- www.beningo.com
  - Blog, White Papers, Courses
  - Embedded Bytes Newsletter
    - http://bit.ly/1BAHYXm

From www.beningo.com under

- Blog > CEC – Techniques for Interfacing with Modern Sensors

CEC Continuing Education Center

DesignNews

Thank You

Sponsored by

Digi-Key ELECTRONICS

informa markets