AURIX reading SGP30 sensor data and transmit with HMS Anybus EtherCAT demonstration (DEBUG MODE)

Albert Golek,
Microcontroller, RUTRONIK Electronics Worldwide, Ispringen, Germany

Abstract – **Infineon's AURIX MCUs are devices, which work under the most challenging conditions and secure functional safety. Sensirion has developed the SGP30 digital multi-pixel gas sensor with two air quality signals (TVOC and CO_2). With Anybus, HMS provides the solution for all industrial communication protocols.**

Index Terms* — *Microcontroller (MCU), Automotive Real-time integrated Next Generation Architecture (AURIX), Anybus Compact Com (ABCC), Graphical User Interface(GUI), Build and Integration Framework for Automotive Controller Embedded Software (BIFACES), Central Processing Unit (CPU), Sensirion Gas Sensor (SGP30), Total Volatile Organic Components (TVOC), and Carbone Dioxide (CO_2), Programmable Logic Controller (PLC), Universal Asynchronous Receiver Transmitter (UART), Visual Studio (VS), Universal Debug Engine (UDE), Starter Kit (STK).

I. Introduction

This demonstration shows how sensor data (SGP30) is processed by TC397 TFT Kit and then transferred to a PLC on an industrial communications bus (Anybus). The used protocol in this demonstration is EtherCAT. The PLC appears as a desktop PC with Beckhoffs TwinCAT software. Figure 1 shows the block diagram with the connected components.

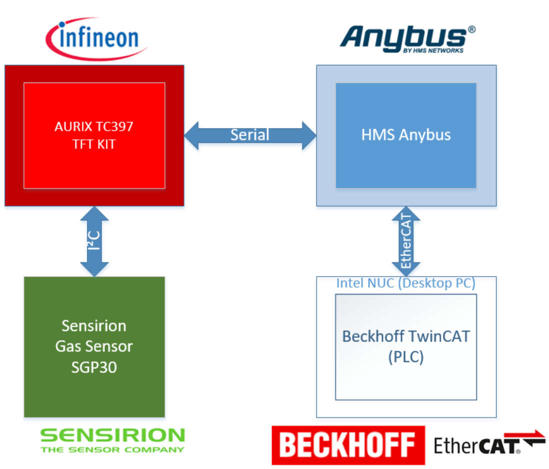

Fig. 1 Block diagram

Actual the demo is in debug mode state. It works fine, but there should be an additional hardware to connect the devices. The developed adapter board will consist of Arduino header and the "ShieldBuddy TC397" consist of Arduino headers. In the debug mode, the Application Kit TC3X7 V2.0 is in use. Figure 2 shows the debug mode setup.

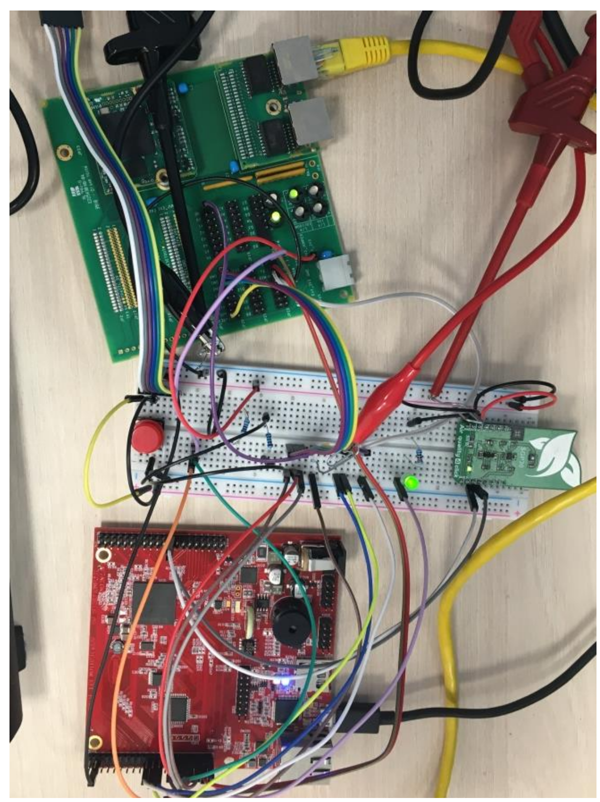

Fig. 2 Debug mode setup

As shown in the picture many cables are needed to realize a communication between the components. The debug version works only in the universal debug engine from pls. The UDE STK 4.10 is the used debug software in this demo.

II. Pin Configuration, Multicore Handling, Communication and Graphical Application

The AURIX MCU demonstration application is configured and programmed with Infineon's BIFACES. (Build and Integration Framework for Automotive Controller Embedded Software). Used Compiler: Hightec gnuc v4.9.1.0-2.0. Before programming, it is necessary to decide which core should perform which task. AURIX will configure the ABCC driver and the AGP30 driver. TwinCAT will do the PLC configuration. The next topics shows the necessary pin connection between the HMS B40-1 NP40 slave and the AURIX TC397 master host system.

A. Pin connection option board [B40-1 (NP40)]

In the HMS appendix manual, Industrial Networks provides examples of possible implementations for the ABCC B40-1 series. Figure 3 shows the implementation of the serial communication.

Necessary pin connections are:

- **Operating Mode** (OM0-3) **pins**: select interface and baud rate.
- **Module Detection** (MD) **pin**: Verify, if the ABCC is detected as module or not, by the host application.
- **Transceiver**: The transmit (TX) and receive (RX) pins are in use to communicate the signals via the UART interface.
- **Reset pin**: Low active - in order to handle the power up, voltage deviations and to be able to support network reset request.
- **Module Identification pins**: Identification, which type of module is connected (e.g. ABCC 40).

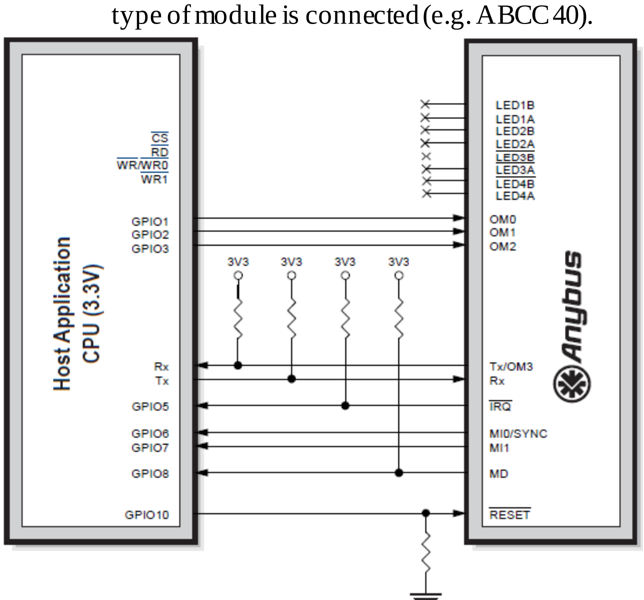

Fig. 3 HMS serial communication example

Consider that the OM3 pin works as the TX pin during the application, only in the startup phase, its function is to configure the desired operating mode.

B. Pin connection TC397

The TC397 application kit uses the following pins for the communication:

- Operating Mode
  - OM0 (PIN:15.7)
  - OM1 (PIN:15.5)
  - OM2 (PIN:15.8)
  - OM3 (PIN:22.3)
- Module Detection
  - MD (PIN:33.8)
- UART
  - Transmit (PIN:30.10)
  - Receive (PIN:15.2)
- Reset
  - RST (PIN:22.1)
- Module Identification
  - MI0 (PIN23.2)
  - MI1 (PIN23.0)
- I²C
  - SCL (PIN:2.5)
  - SDA (PIN:2.4)
- Ground: (GND)

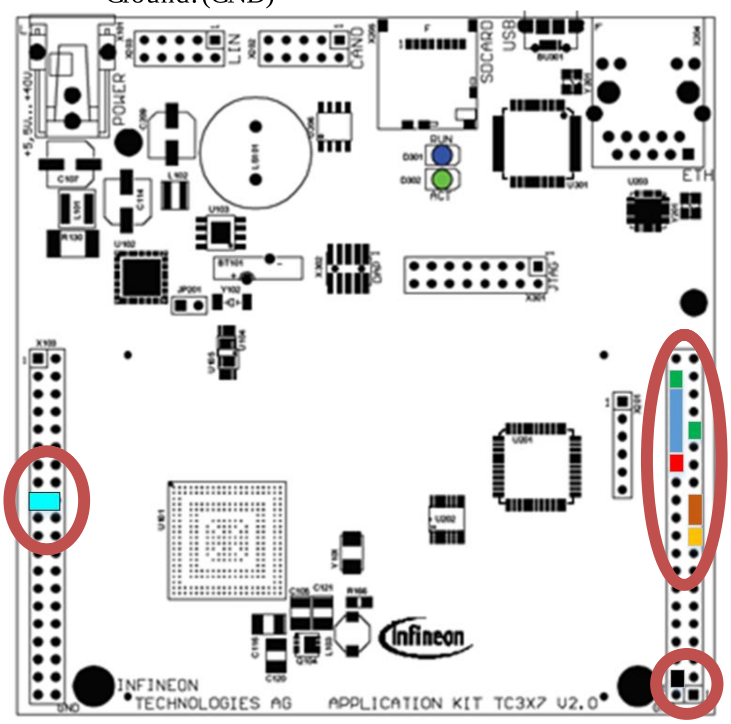

Fig. 4 TC397 pin placement on the application kit

The pin connections occur in the startup initialization of the TC397 modules witch the ABCC driver to guarantee a stabile communication between the two devices. After the initialization, the state machine of the NP40 begins with a command via the serial interface witch starts to perform. In the debug mode, the TC397 manually launches the application from the UDE STD 4.10.

C. Multicore handling

The TC397 MCU contains six cores and four checker cores who are running at the same time and are available over one memory mapping. It is important to make rules of the memory access timing, for example because of race conditions.

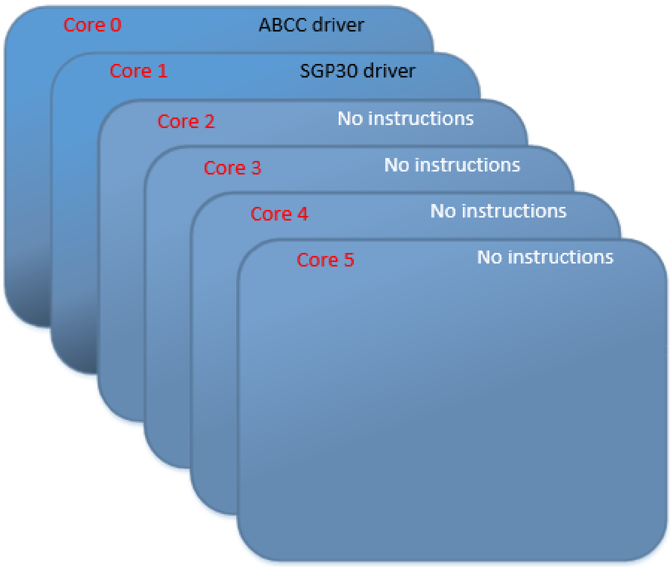

Fig. 5 Task core delegation

Figure 5 shows in a simple way which core handles which task. It is important to define the distribution of tasks before the actual programming so that the programming effort cannot become too large later. Even with such a small project, the planning is indispensable. You have to consider that the CPUs could access the same peripheral at the same time and this would lead to unwanted effects. It helps to program multicore controllers like a real-time operating system with multiple tasks. Remember that in a real safety application you have to decide which core will execute and monitor the safety function.

D. Communication setup hierarchies

The demonstration consist of two Master-Slave hierarchies, inside the first device: the TC397 and the NP40. The master is the AURIX component and the slave is the HMS component. The communicating interface is UART in the debug mode demo. The prepared data is available as object value in the Ethernet protocol, which is available as slave EtherCAT protocol. The SGP30 data must be mapped in the ABCC driver configuration to be available in the protocol procedure. On the other hand, the EtherCAT frame is controlled by the master device, which is simulated here on a PC as PLC with the TwinCAT software. Figure 6 shows the Master-Slave hierarchy.

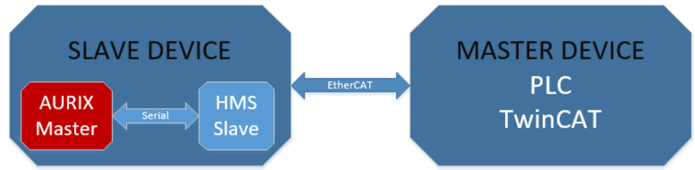

Fig. 6 Master-Slave hierarchy

E. PLC Object mapping and GUI design via VS

TwinCAT detects automatically the EtherCAT protocol when it's correctly configured in Box 1 (CompactCom 40 EtherCAT). Figure 5 shows the TwinCAT project tree with mapped objects.

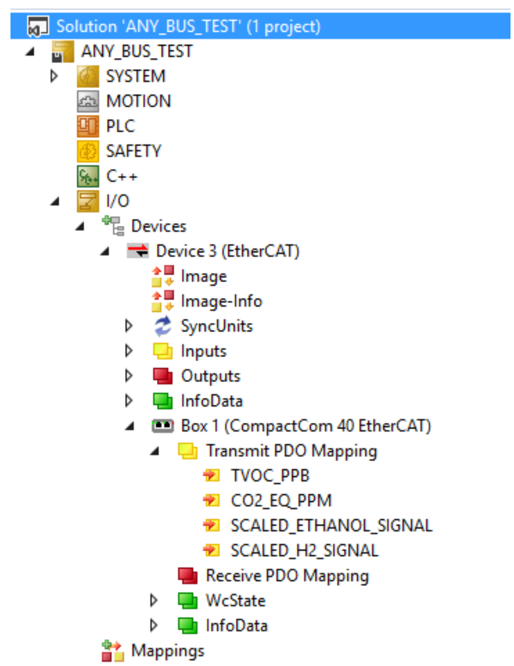

Fig. 5 TwinCAT Project tree

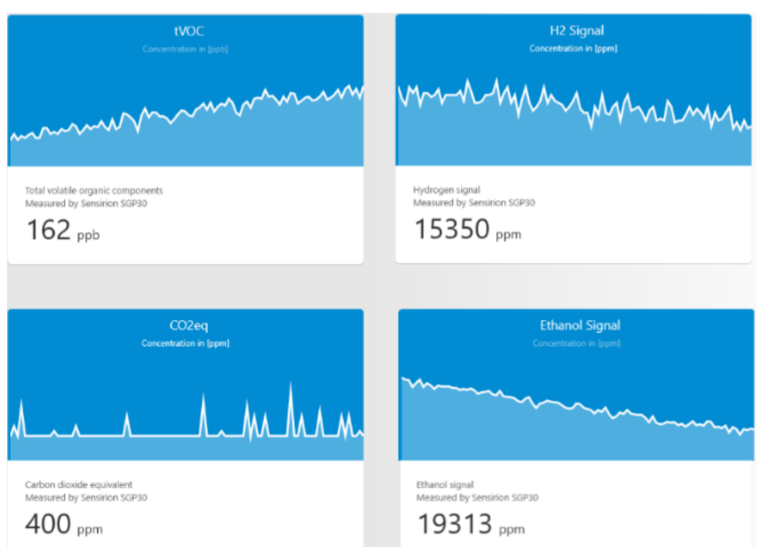

Fig. 7 Gas Sensor Graphical Demonstration

Figure 7 shows the diagramms of data from the sensor.
The CO_2 values are displayed in three different colors: blue, orange and red. The application will change the diagrams color from blue to orange if the CO_2 value is bigger than 1000 ppm (parts per million). If the value is even bigger than 2000 ppm, the color will change from orange to red. Guide range limits values for carbon dioxide concentrations in indoor air areas:

- Source: https://wissenwiki.de/Kohlenstoffdioxid

III. Programming

This demonstration consists of the assembly of several CPUs, which are managed by a central C-Module to use the same global variables for easier communication between them. The name of the C-Module is “smu_ command.c” with its header “smu_demo.h”.

A. Main C-Module

A C-Module with the name “control_command.c” acts as a central unit for the multicore handling tasks. The C-Module contains all functions the CPUs will call. Figure 6 shows the program hierarchy with the main C module.

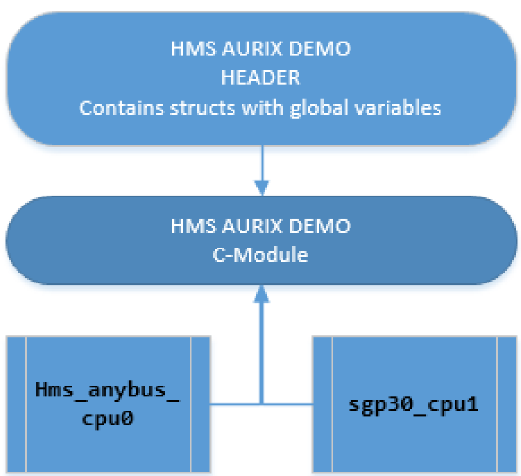

Fig. 8 Program hierarchy

The following functions have the following short described tasks:

- `hms_anybus_cpu0`
  ➔ Initialize Anybus Compact Com drivers and configures the EtherCAT protocol. Communicates with the PLC as Slave.

- `Sgp30_cpu1`
  ➔ Initialize Sensirions SGP30 Gas sensor. Provide data for CPU0 and the hms_anybus_cpu0 function.

B. CPU0 performance

First CPU0 will initialize the other CPUs. After the start, the system timer 0 will be initialized. The timer is necessary for the ABCC driver startup phase and the applications run state. After the system timer 0 setup, the Port Pins configuration takes place. Here it is important to choose the right settings for the B40 option board to achieve the desired communications interface. In addition, the NP40 reset pin and the UART communications pins must be considered. After the Port settings, the initialization will reach the UART Interface. This part will usually take place in the ABCC driver initialization, but here it is done first and will consider the ABCC driver later. After all necessary configurations, the ABCC drivers will initialize and start the handler in the hms_anybus_CPU0() function. Figure 7 shows the flowchart of CPU0. The ABCC handler works in a state machine manner. In the run state, the driver expects data from the SGP30 gas sensor, which belongs to CPU1.

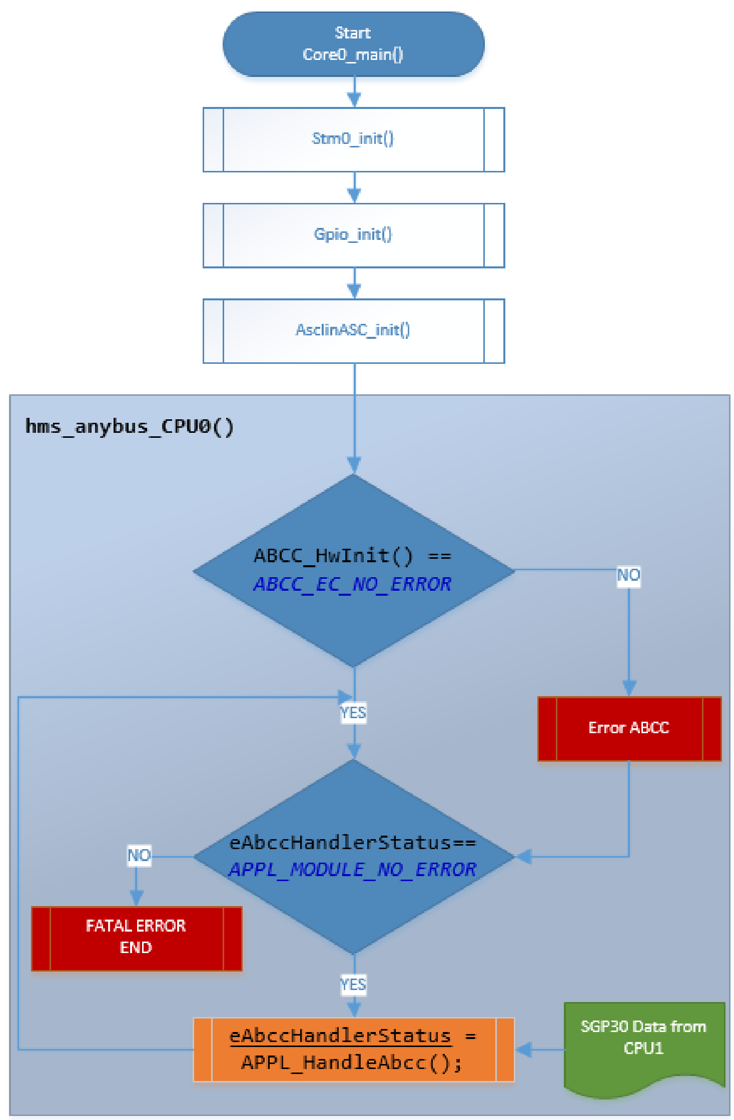

Fig. 9 CPU0 flowchart

C. *CPU1 performance*

CPU1 will be triggered by CPU0 in the software startup phase. Like CPU0 the CPU1 needs a system timer. Here the initialization starts with the system timer 1. The timer is necessary for the SGP30 driver and to the I²C Interface to coordinate the request response data frames between the sensor and the AURIX at the right time. After the timer setup, the I²C initialization for the sensor communication will take place. After all necessary configurations, the sgp30_CPU0() function will start and collect data from the SGP30 gas sensor. The received data will be provided for CPU0. Figure 8 shows the flowchart of CPU1.

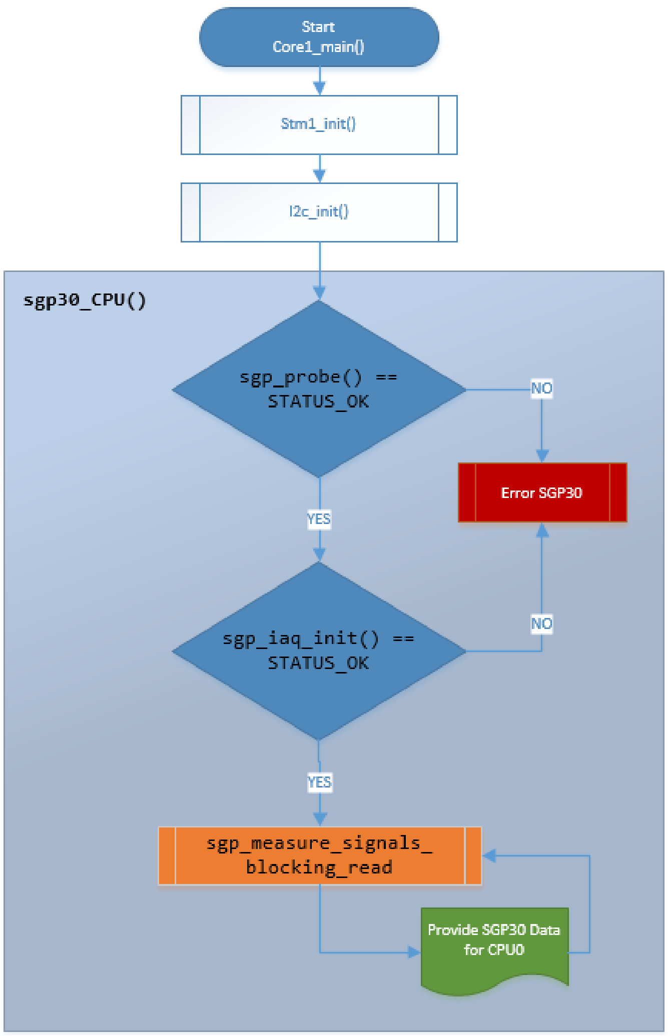

Fig. 10 CPU1 flowchart

IV. Developer Debug Version Benefits

A. *Debug mode software verify*

During the initialization of the ABCC and the SGP30 drivers, a software verification is very helpful in every step of the projects development phase. The software verification takes place in every critical program function, which concerns the drivers. To verify the development steps the printf() function and a breakpoint strategy is placed in the application. Figure 11 shows the highlighted functions in a green frame. Every if-else-operation is monitored and helps to find quickly code issues.

```
APPL_AbccHandlerStatusType eAbccHandlerStatus = APPL_MODULE_NO_ERROR;
ABCC_ErrorCodeType hardwareInit = ABCC_EC_NO_ERROR;

printf("Begin with ABCC driver initialization \n");// Start ABCC
hardwareInit = ABCC_HwInit();
if (hardwareInit == ABCC_EC_NO_ERROR) //Check if the HW is available
{
    printf("ABCC_HwInit() OK -> next step \n");
}
else
{
    printf("Show ABCC ErrorCodeType number:%i\n",hardwareInit);
    show_ABCC_ErrorCodeType(hardwareInit);// show error text
}
printf("HW INIT done\n");// No Error
```

Fig. 11 driver monitoring example via printf()

B. *Debug mode test simulation*

In debug mode the testing of all driver steps are monitored via the function printf(). In UDE STK 4.10 all steps can be monitored trough the "Simulated I/O" window. This necessary workflow shows the important NP40 states during the communications setup steps and the TC397 initializations steps. Figure 12 shows as demonstration of some prints from the function in the UDE STD 4.10.

```
Simulated I/O - connected via simulated I/O
ABP_ANB_STATE_PROCESS_ACTIVE
tVOC Concentration:    13 [ppb]     CO2eq Concentration:   400 [ppm]
tVOC Concentration:    10 [ppb]     CO2eq Concentration:   410 [ppm]
INT
{-> 0, -> 0, -> 0, -> 0, -> 0, -> 0, -> 0, -> 0, -> 0, -> 0, -> 0, ->
OK:Received followed data INT
{-> 0, -> 0, -> 0, -> 0, -> 0, -> 0, -> 0, -> 0, -> 0, -> 0, -> 0, ->
OK:Received followed data INT
{-> 0, -> 0, -> 0, -> 0, -> 0, -> 0, -> 0, -> 0, -> 0, -> 0, -> 0, ->
ABP_ANB_STATE_PROCESS_ACTIVE
tVOC Concentration:    10 [ppb]     CO2eq Concentration:   410 [ppm]
```

Fig. 12 driver monitoring via printf()

V. GUI Presentation

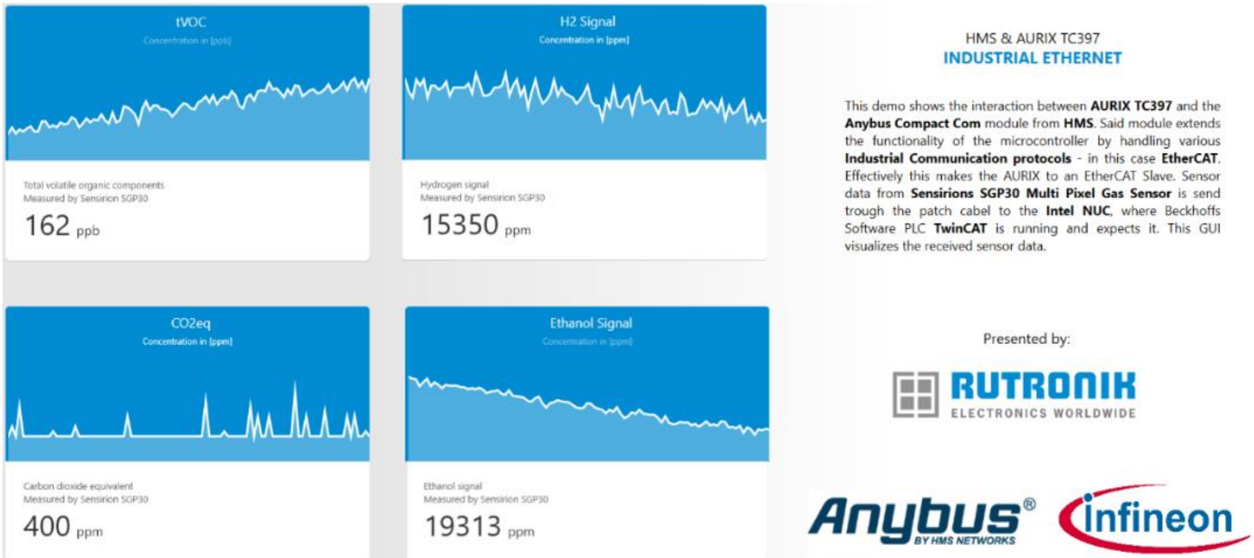

Figure 13. GUI

Figure 11 shows the whole GUI from the Desktop PC where two applications are running at the same time.

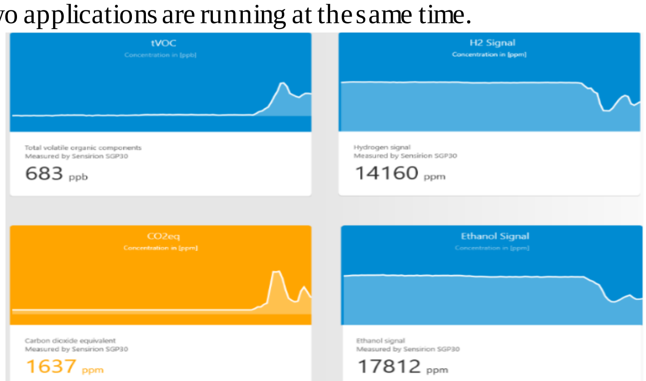

Figure 14. CO_2 value > 1000

Figure 11 shows a warning state because the bad air quality rises above 1000 ppm. Figure 12 shows a higher danger scenario witch CO_2 value >2000 ppm

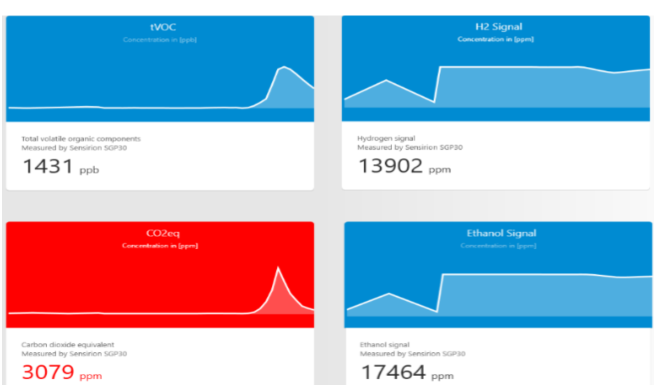

Figure 15. CO_2 value > 2000

VI. Summary

This demonstration shows how the AURIX MCU is able to work witch the HMS ABCC device via the UART interface. In addition, the AURIX MCU transmits in a very efficient I²C process the SGP30 gas data. For further safety applications, the intern AURIX Module Safety management Unit (SMU) can be very useful. The AURIX MCU can also communicate via the SPI interface, which transmits data faster in a full duplex mode. All interfaces can be monitored and reach a high safety standard by the AURIX controller.

References

[1] "Application Kit TC3X7 V2.0" Edition 2018-06, Published by Infineon Technology AG. (Version 2.0). https://www.infineon.com

[2] "AURIX User's Manual TC3xx part 1" Edition 2019-09, Published by Infineon Technology AG. (Version 1.3.0). https://www.infineon.com

[3] "AURIX User's Manual TC3xx part 2" Edition 2019-09, Published by Infineon Technology AG. (Version 1.3.0). https://www.infineon.com

[4] "Appendix TC3x-B" Edition 2019-09, Published by Infineon Technology AG. (Version 1.3.0). https://www.infineon.com

[5] "Host Application Implementation Guide" Edition HMSI-27-334 1.6, Published HMS Industrial Networks. (Version1.6) mailto:info@hms.se

[6] "DESIGN GUIDE" Edition HMSI-27-230 3.4, Published by HMS Industrial Networks. (Version 3.4) mailto:info@hms.se

[7] "SOFTWARE DESIGN GUIDE" Edition HMSI-216-125 3.9, Published by HMS Industrial Networks. (Version 3.9) mailto:info@hms.se

Contact:

Albert Golek
Technical Support Engineer
RUTRONIK Elektronische Bauelemente GmbH
Industriestr. 9
75228 ISPRINGEN
Germany
Albert.Golek@rutronik.com
www.rutronik.com