skylabs

picoRTU (ESA contract no. 4000127900/19/NL/BJ/zk)

picoRTU system – Preliminary design: Executive Summary Report

Document revision:	***1.0***
Document status:	***Release***
Issue date:	***18/07/2020***

Decentralization of the system with a modular approach benefits in saving overall mass, harness reduction, lower verification efforts and decreases development times of the S/C. By placing units near the source of measurements the signal quality is improved by reducing noise coupling or placed close to the sink, the overall EMI is improved and consequently simplifies the harness design.

The modular system based on off-the-shelf generic units has additional important advantages. The picoRTU units with proper parametrization of firmware (as in automotive industry) will already fulfil requirements of discrete interfaces for acquisition and actuation. Modular system efficiently copes with late requirement changes, as it is a matter of firmware reconfiguration with changing or adding new units.

During avionics project run, modularity prevents project delays, for instance, requirements can still be updated and this would not be a blocker or hindering project timeline-flow. New module is added to the stack, configured properly and thus enable more agile development flow. Multiple teams can work concurrently on the system. Each team design its own part of the system in the initial stage of the project and later join the work into a seamless integration. This is achieved due to well defined and well-established interfaces and the common system protocol. Modules can be independently upgradable which reduces risk of changing the rest of the system as also additional modules can be easily added and configured. Beside this, during development of the satellite the picoRTU stacks can already be connected to the on-board sensors as also other equipment and start gathering housekeeping data even without OBC or any other data handling system. This truly enables parallel verifications activities of the satellite.

Cabling topology between modules is beneficial, since it is predefined, reduced to minimum and therefore simplified. For example, instead of hundreds of wires between nodes, in distributed system this is reduced to only several due to redundant bus topology communication.

The picoRTU network implements a robust ECSS CANopen communication protocol which increases the Hardware Abstraction Layer of the Remote terminal Unit (RTU) network to the information level and facilitates easier integration of the RTU network into a complete system. The RTU network is based on the request and response messaging model, supporting more complex RTU functionality. In addition, unsolicited telemetry can be enabled in a continuous periodic manner.

System diagnosis is simpler to establish in a distributed system because of fault containment on and between nodes. For instance, if there is a cable fault somewhere between the nodes, system switches to redundant bus.

The following features is the integral part of the RTU network: Master and Slave functionality – a single picoRTU unit on the network is the active data producer with request-response philosophy, with exception to support unsolicited telemetry generation in subscriber producer approach.

Distributed systems pose fragmentation of central intelligence into distributed nodes to provide simple functionality. PicoRTU software is represented in practice as a very simple piece of code. Such a code can be tested and verified on a functional level and thus represents major costs saving in development and testing phase, comparing to SW code validation at S/C main on-board computers.

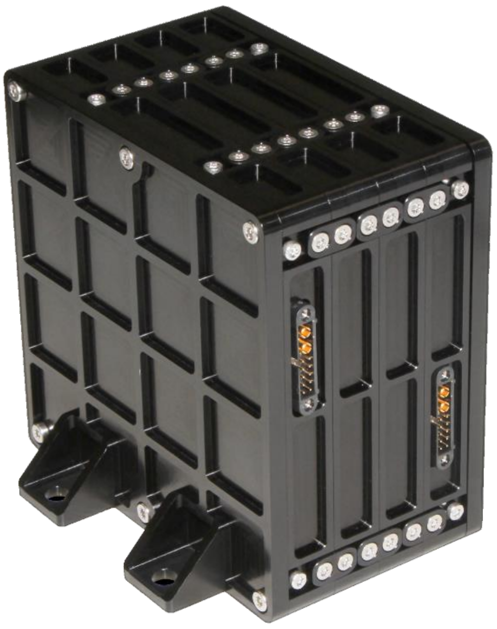

Figure 1: picoRTU system

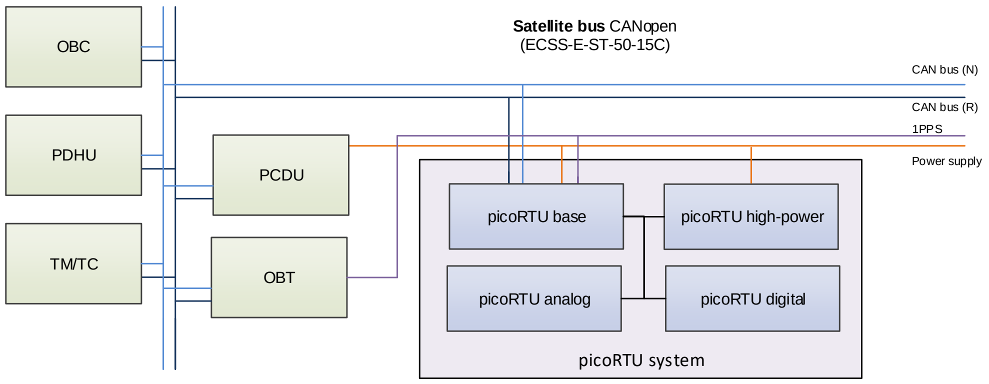

Figure 2: picoRTU system in satellite architecture

Each picoRTU system (or module) is composed of base unit and one or more add-on units attached into homogeneous system. Base unit is designed around PicoSkyFT processor. PicoSkyFT manages communication with external devices (via RTI) and internal communication with add-on units. Data from add-on units are transferred to base unit for further processing and storage to NVM.

PicoRTU system supports two variants:

- **Modular unit** as a standalone unit with stacking capabilities (scalable functionality)
- **Mezzanine unit** as an integrated unit (only PCB with connectors for further integration)

PicoRTU system concept is based on distributed picoRTU system units, where picoRTUs RTIs and master (e.g. OBC) are interconnected via seamless robust redundant CAN bus network. The CAN network is used for gathering TM and executing TC on the picoRTU units. Standardized CANbus extension protocol is supported as an application layer protocol, operating in conjunction with the CAN Network data link layer.

PicoRTU system user interfaces supports various interfaces

picoRTU unit architecture

PicoRTU system is formed out of 4 different units. The following table summarises the configuration of each picoRTU unit. PicoRTU contains a limited set of user interfaces for demonstration purposes and still enable a true distributed on-board data handling architecture.

Table 1: Different picoRTU units

picoRTU variant \ Interfaces	CAPs	PowerIO-HVCs	Digital IOs	RS485/422 interfaces
picoRTU-base (Base unit)	8	0	8	4
picoRTU-analog (Analog acquisition unit)	32	0	0	0
picoRTU-digital (Digital and communication unit)	0	0	64	8
picoRTU-hp (High power unit)	0	12	0	0

PicoRTU system is based on unified firmware architecture. It consists of two main parts – user application and supervision application. User application handles most of the normal processing of the picoRTU. Supervision application is used for handling exceptional cases when processor detects corrupted memory or register data (consequence of radiation effects). In worst case supervisor enters safe state (locked) or restarts processor.

PicoRTU firmware is developed on top of FreeRTOS real-time operating system. Main advantages of FreeRTOS are small and simple design and efficient execution with minimum overhead. The kernel consists of only three C files. Code is readable, maintainable and easy to port. Because of the listed advantages FreeRTOS is one of the most popular real-time operating systems in the world. It has been ported to at least 35 microcontrollers. It has also been featured as number one hard real-time embedded OS in multiple Embedded Markets Studies. Another important advantage of FreeRTOS is free and open source. It is distributed under the MIT license.

PicoSkyFT peripheral driver library is used for interacting with processor peripherals. Peripheral library is thoroughly tested and developed with strict embedded development standards. Main advantage of peripheral library is mature and proven firmware which reduces risks in final product. Code examples are provided with distribution of peripheral library. All major peripheral and processor features are demonstrated in examples.

CAN bus is used for communication between OBC and other picoRTUs on the network. CAN communication is compliant with ECSS-E-50-15C standard. On application layer, CANopen standard is followed. Default CAN bus configuration can host 128 devices on the same network.

User interface devices are configured during parametrization with skyEGSE software. Firmware checks for validity of configuration before applying any changes. Firmware monitors behaviour (e.g. power consumption, bus communication) of user interface devices and triggers appropriate response for faulty device.

Telemetry acquisition is executed by explicit command or by pre-defined telemetry acquisition list. Acquired telemetry data is time tagged and stored on picoRTU NVM storage medium. CAN master (e.g. OBC) can request telemetry data, or data is transferred on specific external (e.g. CAN SYNC frame) or internal (e.g. elapsed timer) trigger. PicoRTU firmware can also pre-process certain data types (depending on configuration and data type), for example temperature recordings, and deliver more meaningful information to OBC. That way less resources are consumed on OBC and CAN network bus is less busy.

SkyEGSE software can configure, parametrize and test picoRTU devices over CAN network. This software simplifies pin configuration, setting and clearing parameters, initializing interfaces, selecting redundancy mode, CAN network parameters, etc.

Configurator software is written in C++. Qt application framework is used for creating graphical user interface. Application is designed in cross-platform manner and runs on Windows and Linux. For communication with CAN network, we used custom UART to CAN bridge. USB FTDI drivers must be installed on system before first use.

SkyEGSE software consist of two main parts. First part is configuration part, in which user defines all devices and its parameters on external user interface. Second part is used for testing configured picoRTU. User can display telemetry and command actuation directly from skyEGSE user interface. See Figure 3 for telemetry screen from skyEGSE software.

Another useful feature of skyEGSE is download of housekeeping telemetry data. Errors, warnings, logs, current system configuration and status can be downloaded from device and displayed on UI.

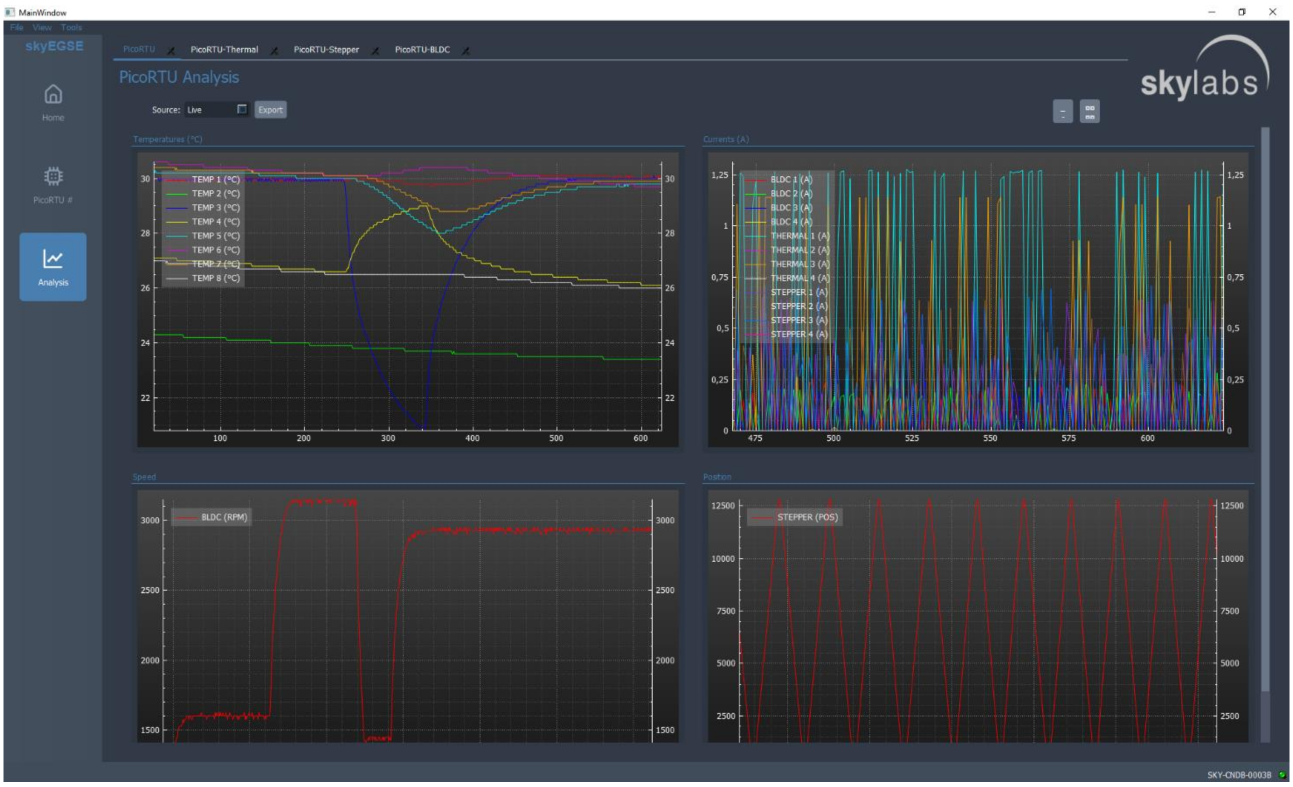

Figure 3: skyEGSE software screenshot

The picoRTU is based on a PicoSkyFT SoC design in Microsemi ProASIC3E FPGA. Detailed configuration of PicoSkyFT picoRTU SoC is provided in the following block diagram:

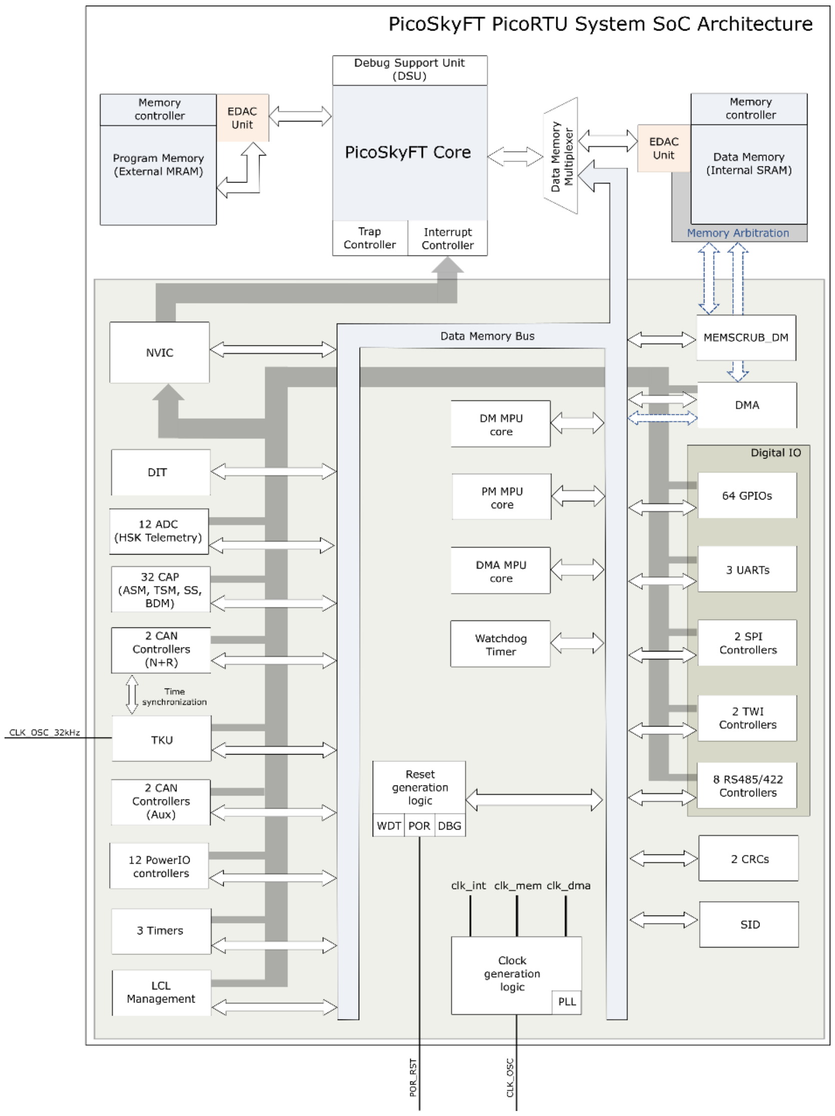

Figure 4: PicoSkyFT picoRTU SoC Architecture

Each PicoRTU module SoC comes with SkyLabs peripherals, consisting of a DMA core, NVIC core, DIT core, TKU, WDR, reset core, multiple timers, LCL management cores, Memory Scrubber on internal SRAM, CRC cores, MPU core for each memory and SID SoC-ID core. Exact number of additional individual peripherals (communication cores, GPIOs, AD converters and power IO cores) is based on functionality of each module (see Table 1 and on block diagram as of Figure 4). Keep note, that not all can be used at the same time on a specific board, since this is related to the board in question. For instance, on picoRTU-hp (Table 1) there are no communication interfaces available, therefore those are unavailable to select and use.

Remote Terminal Interface

Several interfaces have been evaluated for the picoRTU design such as CAN, SpW (EIA-644), TTTech ETH, M-LVDS (EIA-899), RS422/485, FlexRay, JESD204B, MIL-1553B and others. Due to the design concept, selection was further limited only to multi-point interfaces that support adequate number of units on the bus without additional routers or control devices. Based on the trade-offs, derived through the first phase of the project, the CAN interface has been selected for remote terminal interface for picoRTU system. The CAN interface is compliant with CANbus extension protocol defined by ECSS-E-ST-50-15C.

User interfaces

Supported interfaces for picoRTU defined by ECSS (ECSS-E-ST-50-14C) are: analogue signal monitor, temperature sensors monitor (TSM1), bi-level discrete input, and high-power commands. The picoRTU provides additional interfaces as UART (TTL), SPI, I2C, and digital GPIOs, all with 3.3V voltage level logic.

Conclusion

The activity results showed that picoRTU would present the next generation of RTU systems for avionics platform, due to many before mentioned advantages after alignment with real customer specifications. The project has successfully demonstrated a feasibility of having RTU system in a distributed manner, while providing at least the same degree of functionality as conventional centralised RTU solutions.