HSE

Occupational Health & Safety
and Environmental Protection unit

ROMULUSlib

An autonomous, TCP/IP-based, multi-architecture C networking library for DAQ and Control applications

Amitabh Yadav, Hamza Boukabache, Nicola Gerber, Katharina Ceesay-Seitz, Daniel Perrin

CROME Team

Track: Device Control and Integrating Diverse Systems I

18/10/2021 - ICALEPCS 2021, Beijing, China.

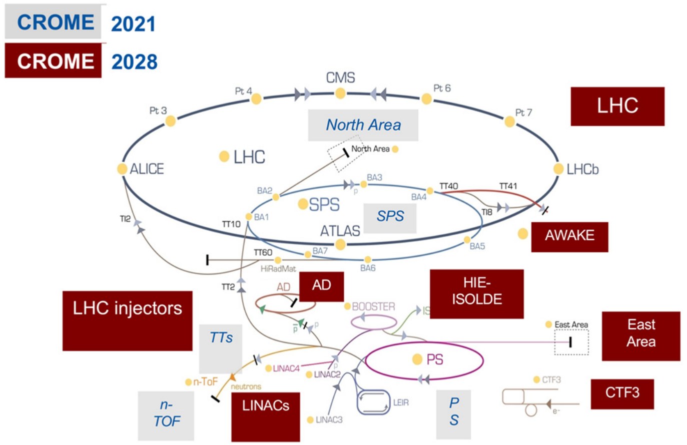
Agenda

- Overview of CROME Radiation Monitors
- Integration of CROME with SCADA supervision
- Motivation for ROMULUSlib
- ROMULUS Protocol
- Architecture and Library Functions
- Applications

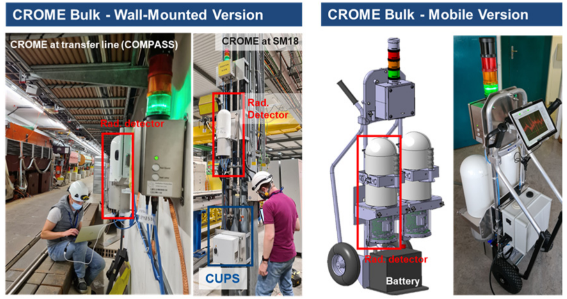
CERN RadiatiOn Monitoring Electronics (CROME)

CROME: The new generation of Radiation Monitoring and Alarm and Interlock generation system.

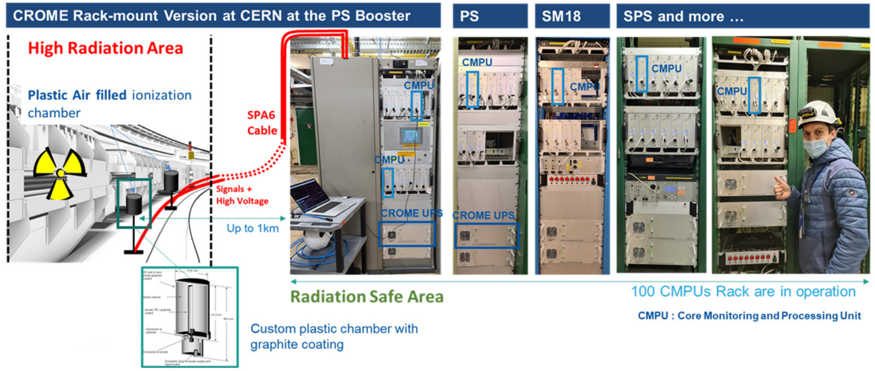
Bulk - Wall Mounted System - Low Radiation levels areas.
Rack System with SPA6 cable - High Radiation levels areas.



CROME Bulk Version – Low radiation levels Areas


CROME Bulk - Wall-Mounted Version
CROME Bulk - Mobile Version
CROME at transfer line (COMPASS)
CROME at SM18
Rad. detector
Rad. Detector
CUPS
Rad. detector
Battery

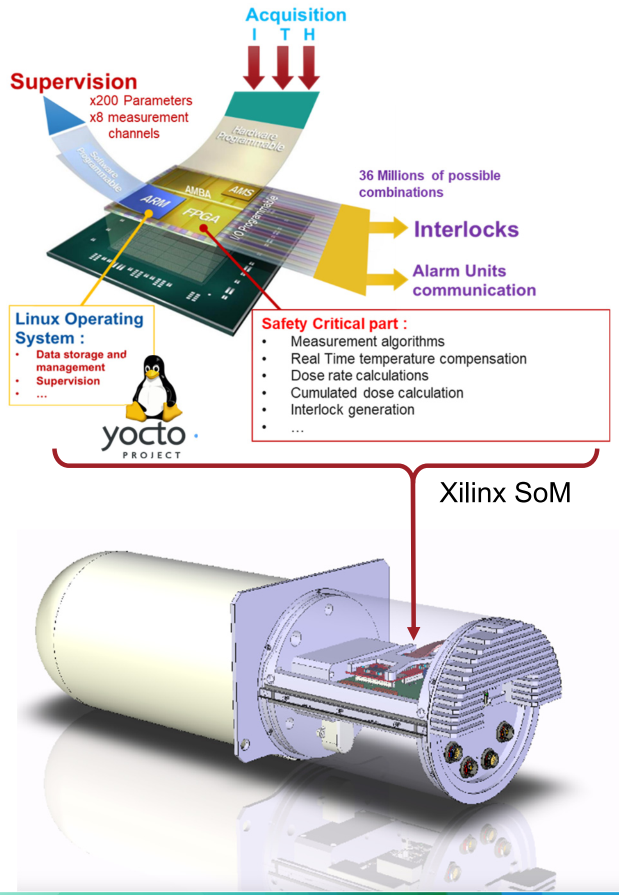
CROME Rack Version – High radiation levels Areas


CROME Rack-mount Version at CERN at the PS Booster
PS
SM18
SPS and more ...
High Radiation Area
Plastic Air filled ionization chamber
SPA6 Cable
Signals + High Voltage
Up to 1km
CMPU
CROME UPS
Radiation Safe Area
Custom plastic chamber with graphite coating
100 CMPUs Rack are in operation
CMPU : Core Monitoring and Processing Unit

H. Boukabache et al., “Towards a novel modular architecture for cern radiation monitoring,”Radiation protection dosimetry, vol. 173, no. 1-3, pp. 240–244, 2017.
C. Toner et al., “Fault resilient fpga design for 28 nm zynq system-on-chip based radiation monitoring system at cern,”Microelectronics Reliability, vol. 100, p. 113 492, 2019.

CERN RadiatiOn Monitoring Electronics (CROME)

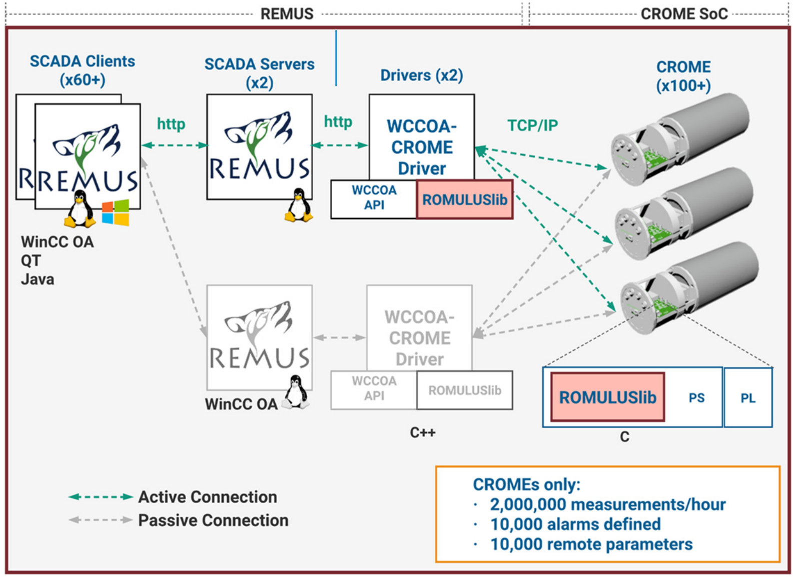
- Employs Radiation Detector connected to the DAQ electronics based on Zynq-7000 Series SoC
- A custom-built embedded linux OS on 32-bit ARM PS core. PS-PL integration implemented on shared-BRAM memory.
- PS is responsible for communication of more than 200 parameters across 8 measurement channels with REMUS/SCADA supervision system via TCP/IP.
- Safety critical operations including Alarms and Interlocks implemented on PL FPGA fabic.



C. Toner, H Boukabache et al., “Fault resilient fpga design for 28 nm zynq system-on-chip based radiation monitoring system at cern,”Microelectronics Reliability, vol. 100, p. 113 492, 2019.

Integration of CROME with REMUS/SCADA Supervision

- REMUS is based on WinCC Open Architecture allows for display of near real-time measurements, alarms statuses and operational states of connected devices.
- CROME communicates with REMUS via TCP/IP protocol over the CERN Technical Network.
- The need for reliable connectivity and logging capabilities motivated the development of a dedicated TCP/IP-based C networking library, ROMULUSlib.
- REMUS currently handles 2 millions measurements/hour using ROMULUSlib from the CROME devices currently operational at CERN.
- ROMULUSlib rests on top of WinCC OA API and forms the interface between CROME devices and REMUS.



Courtesy of Adrien Ledeul

A. Ledeul et al. Data streaming with apache kafka for cern supervision, control and data acquisition system for radiation and environmental protection, 2019.
A. R. Ledeul, "Integration of CROME into REMUS," SoC Interest Group Meeting, 2020, https://indico.cern.ch/event/882283/

ROMULUSlib: A portable C networking library for SoC to SCADA communication

Motivation for ROMULUSlib:

- PLC-to-SCADA control systems for LHC Cryogenics control in accelerator and experimental control, Gas systems,Cooling systems, HVAC etc.
- SoC-to-SCADA systems employed in DAQ and Control systems in ATLAS, CMS, ALICE and other experiments.
- Some libraries for communication over Ethernet:
  Profinet, Ethernet IP, Quasar (based on OPC UA), MODBUS TCP/IP etc.

ROMULUSlib: A portable C networking library for SoC to SCADA communication

Motivation for ROMULUSlib:

- PLC-to-SCADA control systems for LHC Cryogenics control in accelerator and experimental control, Gas systems,Cooling systems, HVAC etc.
- SoC-to-SCADA systems employed in DAQ and Control systems in ATLAS, CMS, ALICE and other experiments.
- Some libraries for communication over Ethernet:
  Profinet, Ethernet IP, Quasar (based on OPC UA), MODBUS TCP/IP etc.

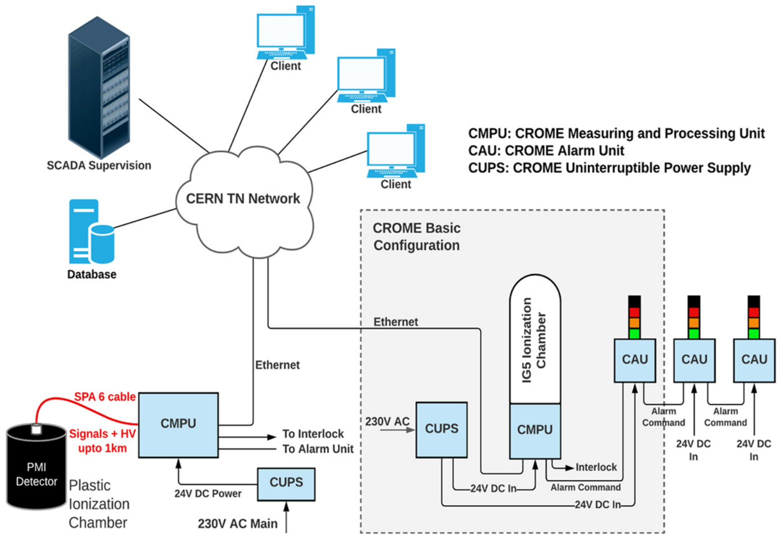
ROMULUSlib offers:

- A full-fledged reliable variable frame-length TCP/IP communication.
- Built completely in C and uses sockets, a part of POSIX standard for compliant OS for multi-architecture support.
- Allows for multi-user full-duplex communication with CROME devices through SCADA supervision and/or an ‘expert application’ connected to the devices through the network via Ethernet connection.
- ROMULUSlib is built and tested successfully with:
  - gcc 4.8.5 on x84_64,
  - arm32, and
  - with Apple clang version 11.0.3 on x86_64 Darwin MacOS Kernel 19.4.0.

ROMULUS TCP/IP Communication Protocol (1)

ROMULUSlib provides:

- Functions to perform full-fledged TCP/IP Networking.
- TCP/IP frame construction for different measurement channels.
- Multiple Communication Modes to transmit single, multiple and (possibly) infinite frames for streaming data over multiple channels.
- Multiple utilities:
  - Functions for log reporting of warnings and errors,
  - Print functions: ROMULUS frame, Measurement Struct etc.
  - Checksum functions etc.

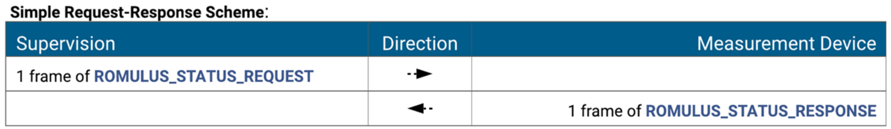


ROMULUS TCP/IP Communication Protocol (2)

Modes of Communication:

- **Simple Request-Response Scheme**

  Single request frame responded to with single response frame.



Simple Request-Response Scheme:

Supervision	Direction	Measurement Device
1 frame of ROMULUS_STATUS_REQUEST	→	
	←	1 frame of ROMULUS_STATUS_RESPONSE

ROMULUS TCP/IP Communication Protocol (2)

Modes of Communication:

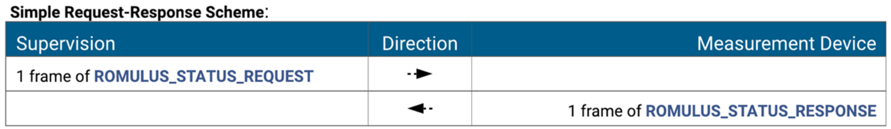
- **Simple Request-Response Scheme**

  Single request frame responded to with single response frame.

- **Complex Request-Response Scheme**

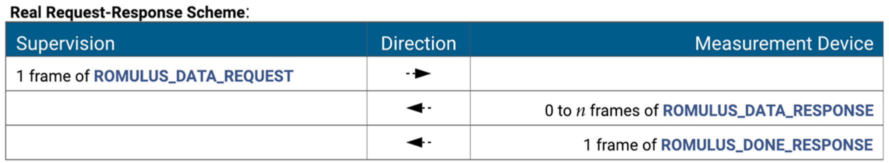
  1. Real Scheme:
  - Single request frame responded to with finite multiple response frames.
  - The last response frame is always fixed to indicate the end of the communication.

  2. Streaming-like Scheme:
  - Single request frame responded to with possibly infinite response frames.
  - Breaks to the connection or stop command by supervision terminates the stream.



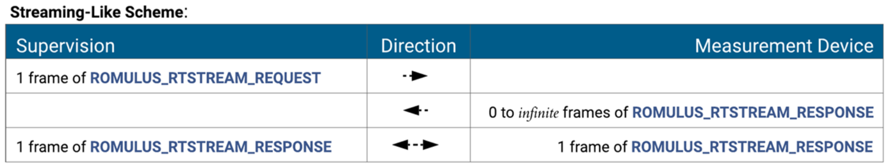
Simple Request-Response Scheme:

Supervision	Direction	Measurement Device
1 frame of ROMULUS_STATUS_REQUEST	→	
	←	1 frame of ROMULUS_STATUS_RESPONSE



Real Request-Response Scheme:

Supervision	Direction	Measurement Device
1 frame of ROMULUS_DATA_REQUEST	→	
	←	0 to n frames of ROMULUS_DATA_RESPONSE
	←	1 frame of ROMULUS_DONE_RESPONSE



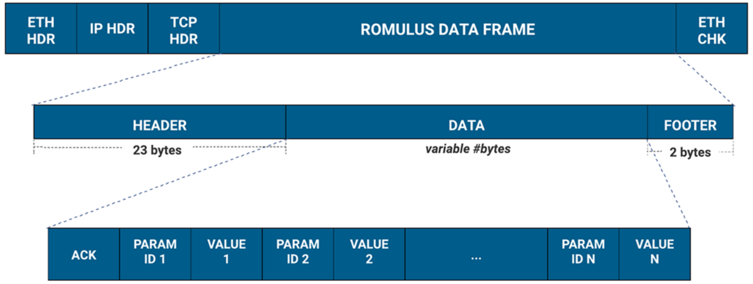
Streaming-Like Scheme:

Supervision	Direction	Measurement Device
1 frame of ROMULUS_RTSTREAM_REQUEST	→	
	←	0 to *infinite* frames of ROMULUS_RTSTREAM_RESPONSE
1 frame of ROMULUS_RTSTREAM_RESPONSE	↔	1 frame of ROMULUS_RTSTREAM_RESPONSE

ROMULUSlib Library Architecture (1)

ROMULUS TCP/IP data frame

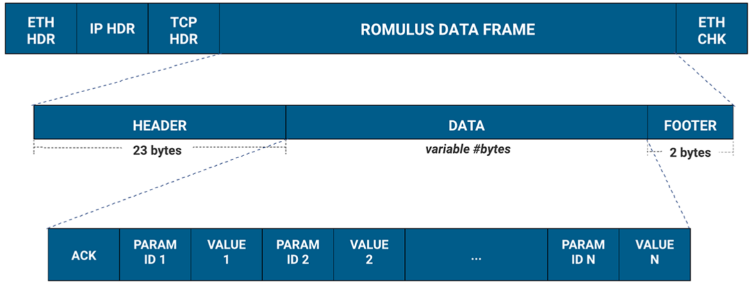
- 1500 bytes of TCP/IP frame.
- TCP and IP headers, ROMULUS data frame is set, and ETH checksum.
- ROMULUS data frame has 23 bytes header for information for device identity and the command ISA.
- The data packet within ROMULUS data frame stores contents that are individually defined for each command and command type.
- ACK acknowledge bits for most response commands.
- CheckSum footer.



ROMULUSlib Library Architecture (1)

ROMULUS TCP/IP data frame

- 1500 bytes of TCP/IP frame.
- TCP and IP headers, ROMULUS data frame is set, and ETH checksum.
- ROMULUS data frame has 23 bytes header for information for device identity and the command ISA.
- The data packet within ROMULUS data frame stores contents that are individually defined for each command and command type.
- ACK acknowledge bits for most response commands.
- CheckSum footer.



ROMULUS Measurement Channels

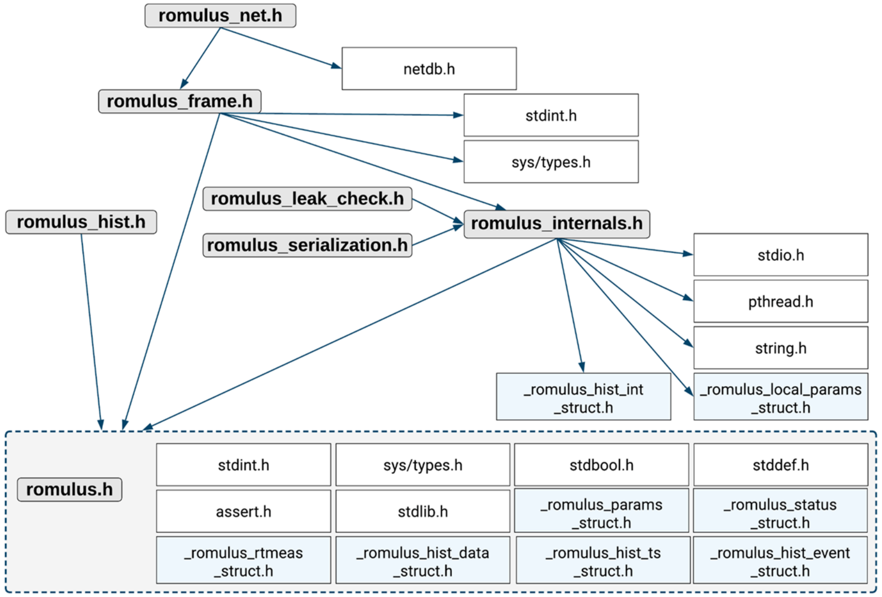
- C Structs to define a measurement channel.
- Supports multiple variables of nearly all C data types.
- Easy update of Struct table for different use cases in DAQ and control.
- 8 defined measurement channels in CROME.

ROMULUSlib Library Architecture (2)

romulus.h
- initialization functions and timestamp
- all data types, structs and debug functions.

romulus_frame.h
- request-response frame construction functions
- frame validity check
- device ID generation

romulus_net.h
- networking
- send, receive functions for TCP/IP



romulus_internals.h
- internal data structures management.
- printing and I/O
- checksum

romulus_hist.h
- historics data/event management.

romulus_leak_check.h
- optional header file for memory leak check functions.

romulus_serialization.h
- data read functions.

ROMULUSlib build instructions and Utility functions

To use ROMULUSlib for your DAQ and Control applications, follow the following 3 simple steps:

1. Update measurement channels in .gen files.

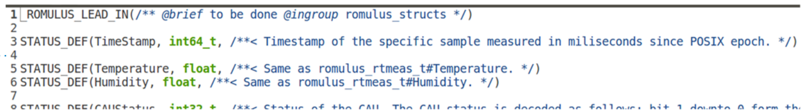
1. Run the Make script for after specifying target architecture for `cross-compile`.
   This automatically generates necessary C Struct files for your use case and generates executables and symbolic link files to be installed in `/usr/lib`.

1. Import ROMULUSlib into your Embedded User Space application to make use of ROMULUSlib's TCP/IP networking functions, starting with `romulus_init()`.

ROMULUSlib build instructions and Utility functions

To use ROMULUSlib for your DAQ and Control applications, follow the following 3 simple steps:

1. Update measurement channels in .gen files.



```
ROMULUS_LEAD_IN(/** @brief to be done @ingroup romulus_structs */)

STATUS_DEF(TimeStamp, int64_t, /**< Timestamp of the specific sample measured in miliseconds since POSIX epoch. */)

STATUS_DEF(Temperature, float, /**< Same as romulus_rtmeas_t#Temperature. */)
STATUS_DEF(Humidity, float, /**< Same as romulus_rtmeas_t#Humidity. */)

STATUS_DEF(CAUStatus, int32_t, /**< Status of the CAU. The CAU status is decoded as follows: bit 1 downto 0 form th
```

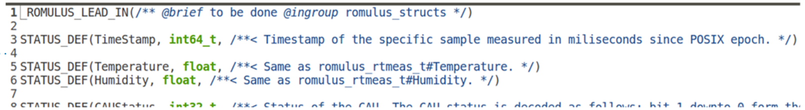
1. Run the Make script for after specifying target architecture for `cross-compile`.
   This automatically generates necessary C Struct files for your use case and generates executables and symbolic link files to be installed in `/usr/lib`.

1. Import ROMULUSlib into your Embedded User Space application to make use of ROMULUSlib's TCP/IP networking functions, starting with `romulus_init()`.

ROMULUSlib build instructions and Utility functions

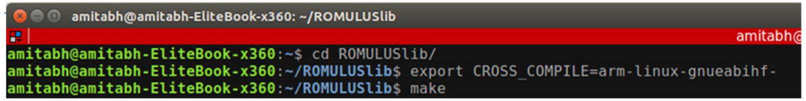
To use ROMULUSlib for your DAQ and Control applications, follow the following 3 simple steps:

1. Update measurement channels in .gen files.



```
ROMULUS_LEAD_IN(/** @brief to be done @ingroup romulus_structs */)

STATUS_DEF(TimeStamp, int64_t, /**< Timestamp of the specific sample measured in miliseconds since POSIX epoch. */)

STATUS_DEF(Temperature, float, /**< Same as romulus_rtmeas_t#Temperature. */)
STATUS_DEF(Humidity, float, /**< Same as romulus_rtmeas_t#Humidity. */)

STATUS_DEF(CAUStatus, int32_t, /**< Status of the CAU. The CAU status is decoded as follows: bit 1 downto 0 form th
```

1. Run the Make script for after specifying target architecture for `cross-compile`.
   This automatically generates necessary C Struct files for your use case and generates executables and symbolic link files to be installed in `/usr/lib`.



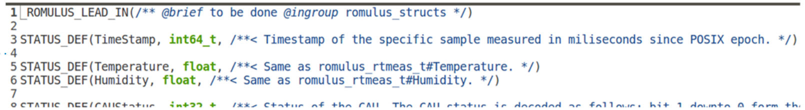
```
amitabh@amitabh-EliteBook-x360: ~/ROMULUSlib
amitabh@
amitabh@amitabh-EliteBook-x360:~$ cd ROMULUSlib/
amitabh@amitabh-EliteBook-x360:~/ROMULUSlib$ export CROSS_COMPILE=arm-linux-gnueabihf-
amitabh@amitabh-EliteBook-x360:~/ROMULUSlib$ make
```

1. Import ROMULUSlib into your Embedded User Space application to make use of ROMULUSlib's TCP/IP networking functions, starting with `romulus_init()`.

ROMULUSlib build instructions and Utility functions

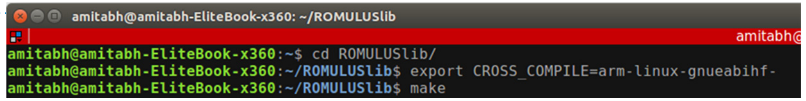
To use ROMULUSlib for your DAQ and Control applications, follow the following 3 simple steps:

1. Update measurement channels in .gen files.



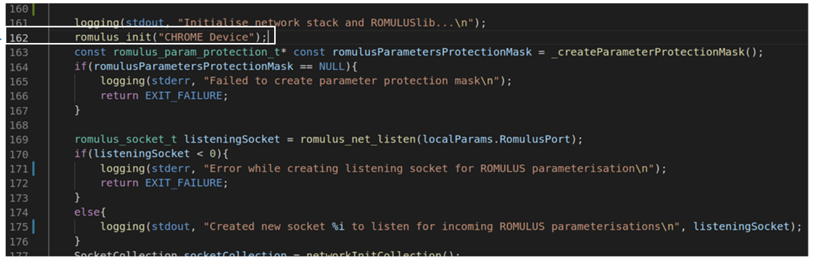
```
ROMULUS_LEAD_IN(/** @brief to be done @ingroup romulus_structs */)

STATUS_DEF(TimeStamp, int64_t, /**< Timestamp of the specific sample measured in miliseconds since POSIX epoch. */)

STATUS_DEF(Temperature, float, /**< Same as romulus_rtmeas_t#Temperature. */)
STATUS_DEF(Humidity, float, /**< Same as romulus_rtmeas_t#Humidity. */)

STATUS_DEF(CAUStatus, int32_t, /**< Status of the CAU. The CAU status is decoded as follows: bit 1 downto 0 form th
```

1. Run the Make script for after specifying target architecture for `cross-compile`.
   This automatically generates necessary C Struct files for your use case and generates executables and symbolic link files to be installed in `/usr/lib`.



```
amitabh@amitabh-EliteBook-x360: ~/ROMULUSlib
amitabh@
amitabh@amitabh-EliteBook-x360:~$ cd ROMULUSlib/
amitabh@amitabh-EliteBook-x360:~/ROMULUSlib$ export CROSS_COMPILE=arm-linux-gnueabihf-
amitabh@amitabh-EliteBook-x360:~/ROMULUSlib$ make
```

1. Import ROMULUSlib into your Embedded User Space application to make use of ROMULUSlib's TCP/IP networking functions, starting with `romulus_init()`.

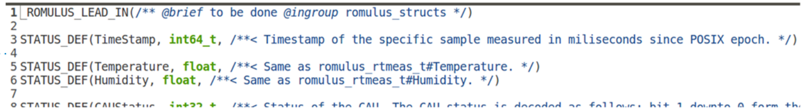


```

    logging(stdout, "Initialise network stack and ROMULUSlib...\n");
    romulus_init("CHROME Device");
    const romulus_param_protection_t* const romulusParametersProtectionMask = _createParameterProtectionMask();
    if(romulusParametersProtectionMask == NULL){
        logging(stderr, "Failed to create parameter protection mask\n");
        return EXIT_FAILURE;
    }

    romulus_socket_t listeningSocket = romulus_net_listen(localParams.RomulusPort);
    if(listeningSocket < 0){
        logging(stderr, "Error while creating listening socket for ROMULUS parameterisation\n");
        return EXIT_FAILURE;
    }
    else{
        logging(stdout, "Created new socket %i to listen for incoming ROMULUS parameterisations\n", listeningSocket);
    }
    SocketCollection socketCollection = networkInitCollection();
```

ROMULUSlib build instructions and Utility functions

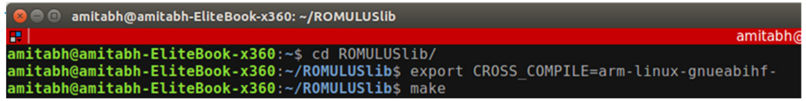
To use ROMULUSlib for your DAQ and Control applications, follow the following 3 simple steps:

1. Update measurement channels in .gen files.



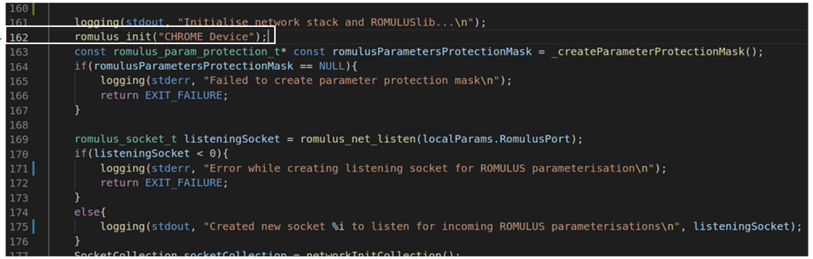
```
ROMULUS_LEAD_IN(/** @brief to be done @ingroup romulus_structs */)

STATUS_DEF(TimeStamp, int64_t, /**< Timestamp of the specific sample measured in miliseconds since POSIX epoch. */)

STATUS_DEF(Temperature, float, /**< Same as romulus_rtmeas_t#Temperature. */)
STATUS_DEF(Humidity, float, /**< Same as romulus_rtmeas_t#Humidity. */)

STATUS_DEF(CAUStatus, int32_t, /**< Status of the CAU. The CAU status is decoded as follows: bit 1 downto 0 form th
```

1. Run the Make script for after specifying target architecture for `cross-compile`.
   This automatically generates necessary C Struct files for your use case and generates executables and symbolic link files to be installed in `/usr/lib`.



```
amitabh@amitabh-EliteBook-x360: ~/ROMULUSlib
amitabh@
amitabh@amitabh-EliteBook-x360:~$ cd ROMULUSlib/
amitabh@amitabh-EliteBook-x360:~/ROMULUSlib$ export CROSS_COMPILE=arm-linux-gnueabihf-
amitabh@amitabh-EliteBook-x360:~/ROMULUSlib$ make
```

1. Import ROMULUSlib into your Embedded User Space application to make use of ROMULUSlib's TCP/IP networking functions, starting with `romulus_init()`.



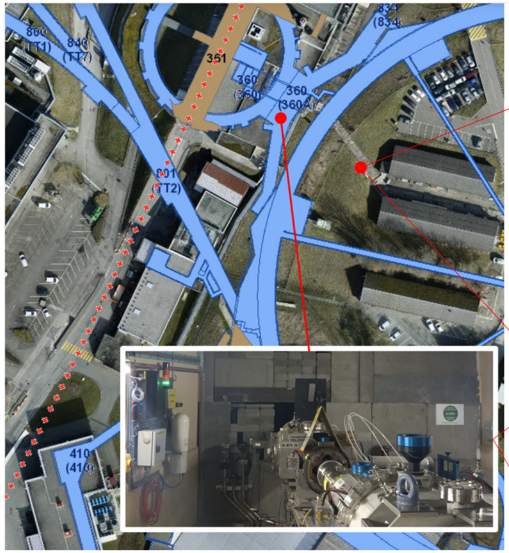
```

    logging(stdout, "Initialise network stack and ROMULUSlib...\n");
    romulus_init("CHROME Device");
    const romulus_param_protection_t* const romulusParametersProtectionMask = _createParameterProtectionMask();
    if(romulusParametersProtectionMask == NULL){
        logging(stderr, "Failed to create parameter protection mask\n");
        return EXIT_FAILURE;
    }

    romulus_socket_t listeningSocket = romulus_net_listen(localParams.RomulusPort);
    if(listeningSocket < 0){
        logging(stderr, "Error while creating listening socket for ROMULUS parameterisation\n");
        return EXIT_FAILURE;
    }
    else{
        logging(stdout, "Created new socket %i to listen for incoming ROMULUS parameterisations\n", listeningSocket);
    }
    SocketCollection socketCollection = networkInitCollection();
```

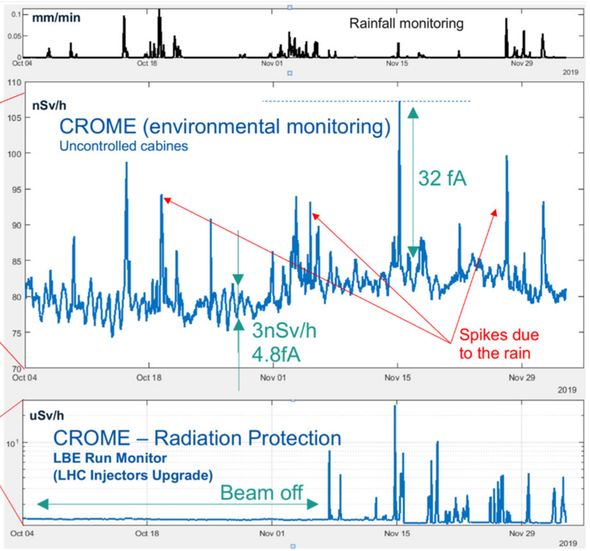
In order to facilitate debugging, ROMULUSlib provides additional utilities.

1. *remote_stream*: prints the TCP/IP frames for Real-Time Data.
2. *struct_info_printer*: prints all Struct Information as defined by the user.
3. *remote_dump*: prints status parameters, configuration parameters and hist data for specified time duration.

Applications: CROME device at CERN

- Currently 150 CROME devices are installed at CERN at various locations.
- All commands/communications sent over CERN TN using ROMULUS protocol implemented in ROMULUSlib.
- Device communication access available to authorized MPs of CERN RP group via REMUS.
- ROMULUSlib_v6.2 in operation.
- **ROMULUSlib_v7** released (in operation from December 2021).
- ROMULUSlib_v1.0 developed and under testing for **CROME CJB**, the intelligent router for experiment interlocks.



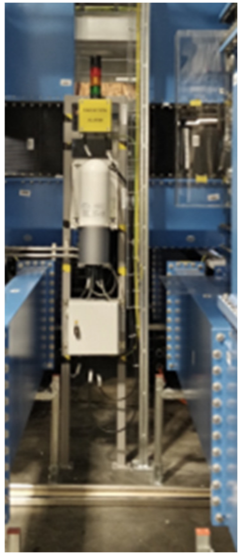


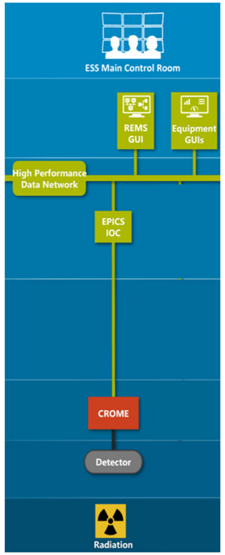
H. Boukabache, "CROME remote management of SoC-based radiation monitors both at CERN and ESS," System-on-Chip 2nd Workshop - CERN, 2021, https://indico.cern.ch/event/996093/

Applications: ROMULUSlib integration with EPICS at ESS

- European Spallation Source (ESS) in Sweden deploys CROME radiation monitors communication using ROMULUS protocol with EPICS framework.

- The ESS' Radiological and Environmental Monitoring System (REMS) uses a two-fold approach:
  a. InfluxDB and Grafana approach for quick online integration
  b. EPICS integration for real time DAQ and control.





J. Hast, "ESS use case of the CROME monitor with EPICS,"2nd System-on-Chip Workshop - CERN, 2021,https://indico.cern.ch/event/996093/
EPICS: Experimental Physics and Industrial Control Systems

System Reliability through Regression Testing: RomLibEmu

- ROMULUS protocol works through request and response message passing.
- So that, malformed packets cannot create unexpected behaviour of CROME devices.
- An independent test framework, RomLibEmu, is developed in Python 3 for application robustness, reliability and safety.

WEBR01

RomLibEmu: Network interface stress tests for the CERN RadiatiOn Monitoring Electronics (CROME)

by Katharina Ceesay-Seitz

October 20, 2021 9:15 pm - 10:30 pm

K. Ceesay-Seitz, M. Leveneur, H. Boukabache, and D. Perrin,“Romlibemu: Network interface stress tests for the cern radi-ation monitoring electronics (crome),” in18th InternationalConference on Accelerator and Large Experimental PhysicsControl Systems (ICALEPCS’21), 2021.

Conclusion

- ROMULUSlib is a standalone TCP/IP networking library developed in C for POSIX compliant OS for multi-architecture support.
- ROMULUS provides complete functions for customization and data packet construction within the TCP/IP frame.
- Communication support via Single frame, multiple frames or streaming data frames.
- ROMULUSlib integrates seamlessly with SCADA supervision systems like REMUS and EPICS to reliably carrying out millions of data packet transactions every hour.

Thank you for your attention.