Ada for Automation Demo Application

062 a4a-k3-wui

Stéphane LOS

Version 2022.05, 2022-05-31

Table of Contents

Chapter 1. Description

1.1. Ada for Automation

Ada for Automation (A4A in short) is a framework for designing industrial automation applications using the Ada language.

It makes use of the libmodbus library to allow building a ModbusTCP client or server, or a Modbus RTU master or slave.

It can also use Hilscher communication boards allowing to communicate on field buses like AS-Interface, CANopen, CC-Link, DeviceNet, PROFIBUS, EtherCAT, Ethernet/IP, Modbus TCP, PROFINET, Sercos III, POWERLINK, or VARAN.

With the help of GtkAda, the binding to the Graphic Tool Kit, one can design Graphical User Interfaces.

Thanks to Gnoga, built on top of Simple Components, it is also possible to provide a Web User Interface.

Nice addition is the binding to the Snap7 library which allows to communicate with SIEMENS S7 PLCs using S7 Communication protocol ISO on TCP (RFC1006).

Of course, all the Ada ecosystem is available.

Using Ada bindings, C, C++, Fortran libraries can also be used.

And, since it is Ada, it can be compiled using the same code base to target all major platforms.

1.2. This demo application

This is a demo application featuring:

- a basic command line interface,
- a basic web user interface making use of Gnoga,
- a kernel with a Modbus TCP Server and one Hilscher cifX channel (K3),
- a trivial application that plays with 16 push buttons and 16 LEDs.

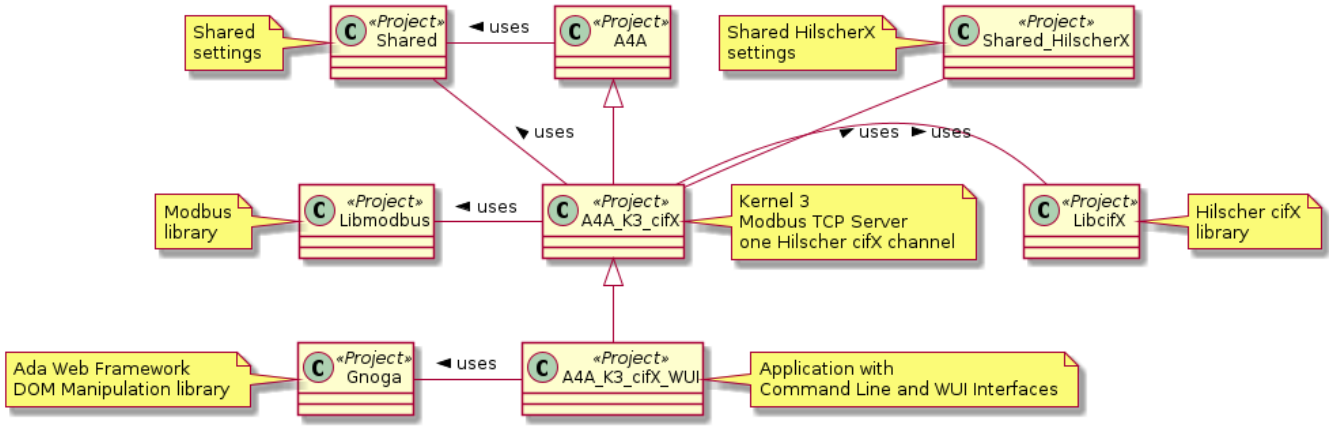
This application is meant to play with **052 a4a_hilscherx_piano**.

There are two variants :

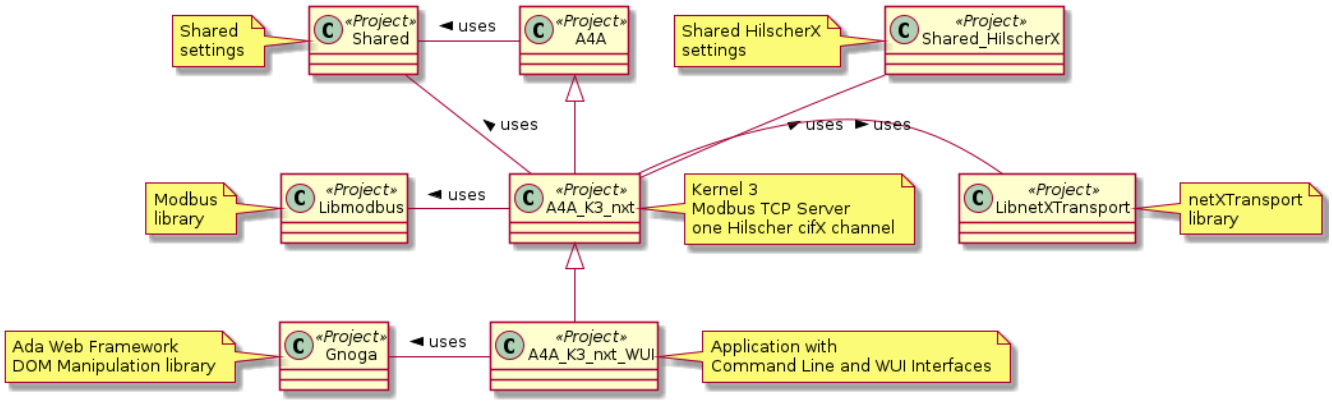
- one using the standard Hilscher cifX API for use with most Hilscher products with ISA, PCI, PCI Express, DPM, SPI interfaces,
- one using the standard Hilscher netXTransport API which has the same cifX API but via a TCP/IP connection for use with Hilscher netHOST.

Chapter 2. Projects diagram

The following picture shows the diagram of the project using cifX API :



The following picture shows the diagram of the project using netXTransport API :



Chapter 3. License

Those files are included in the **Ada for Automation** root folder :

COPYING3
The GPL License you should read carefully. GNU GENERAL PUBLIC LICENSE Version 3, 29 June 2007

COPYING.RUNTIME
GCC RUNTIME LIBRARY EXCEPTION Version 3.1, 31 March 2009

Hence, each source file contains the following header :

```
------------------------------------------------------------------------------
--                            Ada for Automation                            --
--                                                                          --
--                   Copyright (C) 2012-2023, Stephane LOS                  --
--                                                                          --
-- This library is free software;  you can redistribute it and/or modify it --
-- under terms of the  GNU General Public License  as published by the Free --
-- Software  Foundation;  either version 3,  or (at your  option) any later --
-- version. This library is distributed in the hope that it will be useful, --
-- but WITHOUT ANY WARRANTY;  without even the implied warranty of MERCHAN- --
-- TABILITY or FITNESS FOR A PARTICULAR PURPOSE.                            --
--                                                                          --
-- As a special exception under Section 7 of GPL version 3, you are granted --
-- additional permissions described in the GCC Runtime Library Exception,   --
-- version 3.1, as published by the Free Software Foundation.               --
--                                                                          --
-- You should have received a copy of the GNU General Public License and    --
-- a copy of the GCC Runtime Library Exception along with this program;     --
-- see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see    --
-- <http://www.gnu.org/licenses/>.                                          --
--                                                                          --
------------------------------------------------------------------------------
```

Chapter 4. Building

The provided makefiles uses GPRbuild and provides six targets:

- all : builds the executable,
- app_doc : creates the documentation of the source code,
- clean : cleans the space.

Additionally one can generate some documentation using Asciidoctor with :

- read_me_html : generates the README in HTML format,
- read_me_pdf : generates the README in PDF format,
- read_me : generates the README in both formats.

Chapter 5. Running

This application is meant to play with **052 a4a_hilscherx_piano**.

Of course, it can also play with a physical device of your own.

Use option -f makefile_nxt to use netXTransport instead ot cifX.

In a console:

Build the application:

```
make
```

Optionally create the documentation:

```
make app_doc
```

Run the application:

```
make run
```

Use Ctrl+C to exit.

Optionally clean all:

```
make clean
```

Chapter 6. Directories

bin
Where you will find the executable.

doc
The place where GNATdoc would create the documentation.

obj
Build artifacts go here.

src
Application source files.

Chapter 7. Application

This is a basic **Ada for Automation** application which uses one Hilscher cifX channel configured as a fieldbus master and plays with 16 push buttons and 16 LEDs.

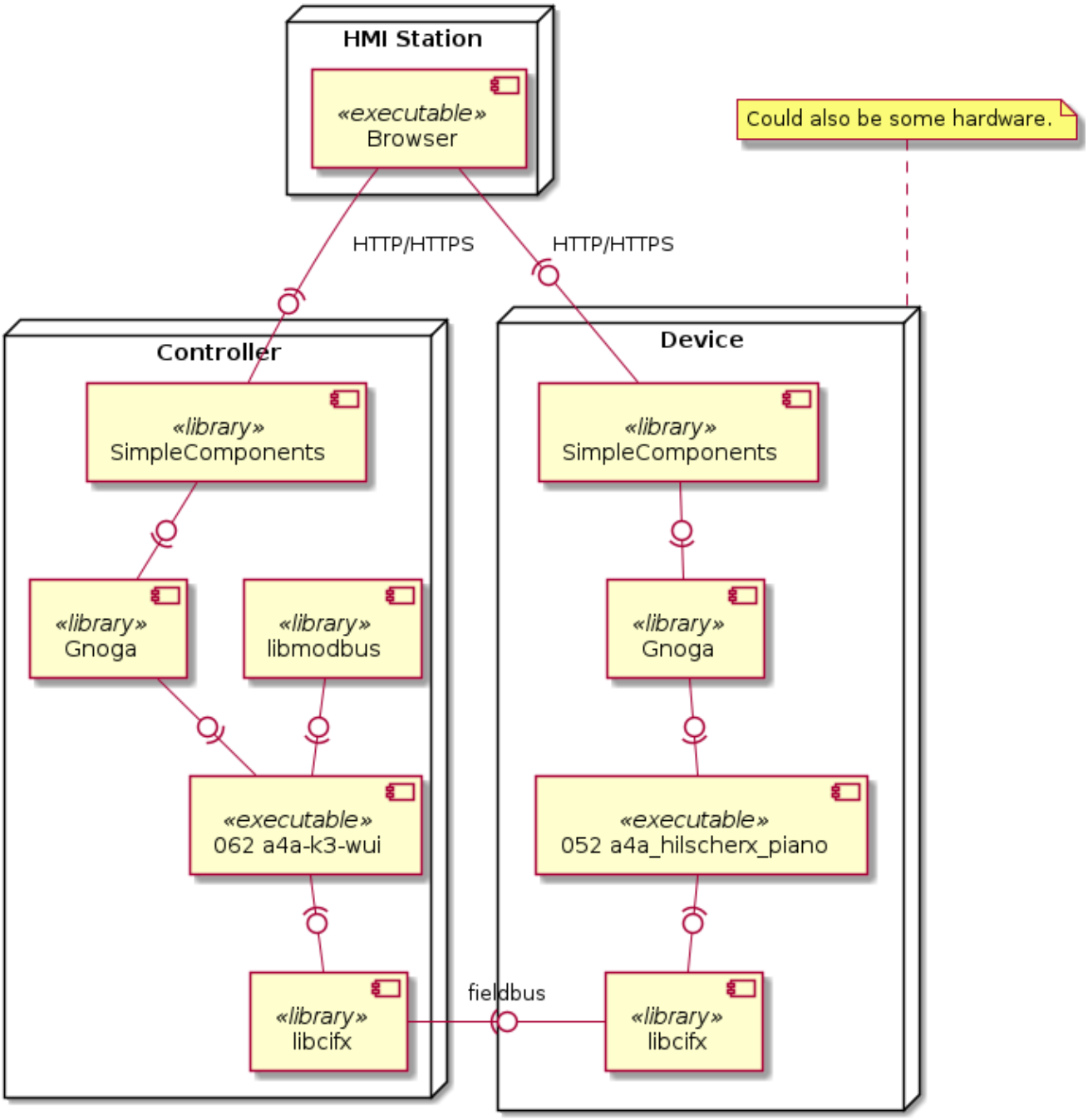
It has a Command Line and Web User Interfaces and communicates with the slave device reading Inputs and writing Outputs using major standard industrial protocols.

The Hilscher cifX channel can be any of a cifX board, comX communication module, a netHOST, or a netX SoC.

The application expects the fieldbus channel to be already configured using SYCON.net.

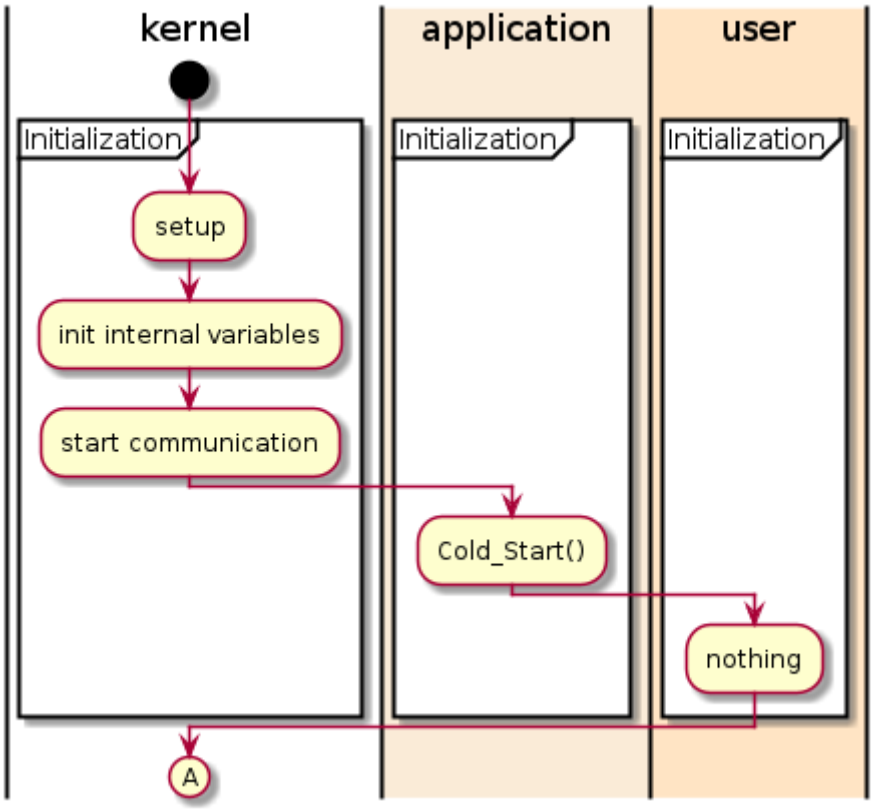
Modbus TCP Server is not used at the moment.

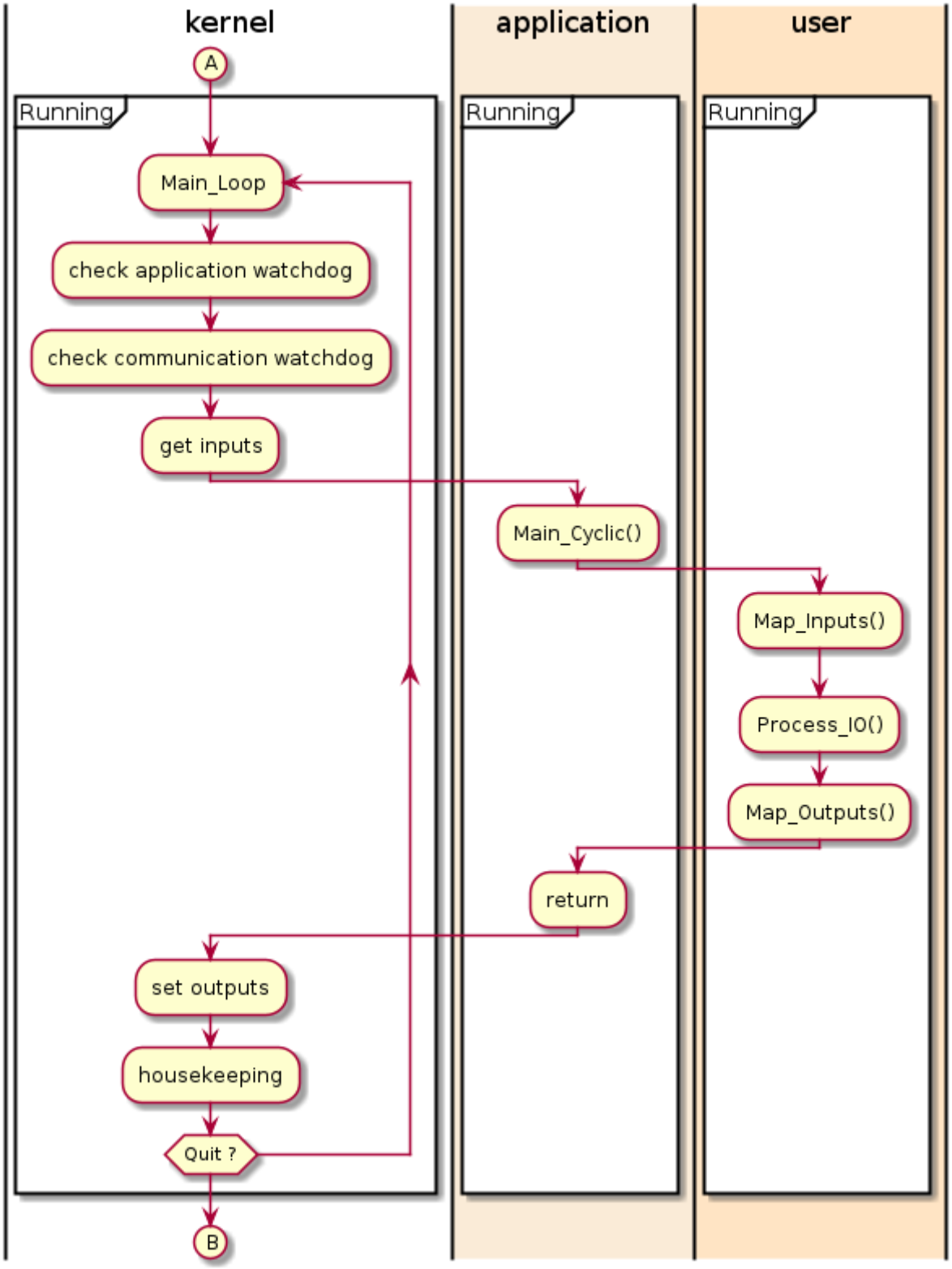
7.1. Deployment diagram

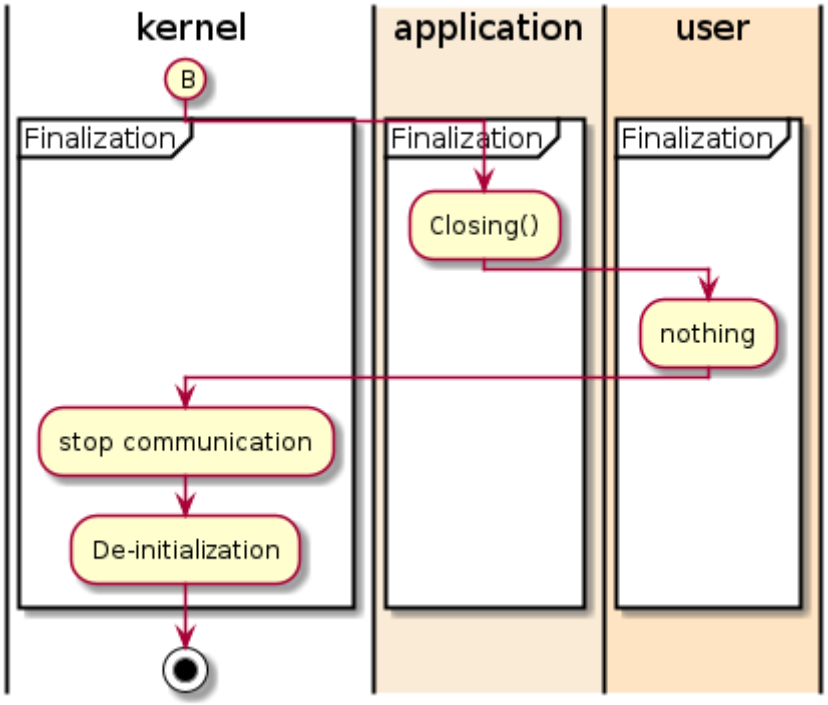

HMI Station
«executable»
Browser
Could also be some hardware.
HTTP/HTTPS
HTTP/HTTPS
Controller
Device
«library»
SimpleComponents
«library»
SimpleComponents
«library»
Gnoga
«library»
libmodbus
«library»
Gnoga
«executable»
062 a4a-k3-wui
«executable»
052 a4a_hilscherx_piano
fieldbus
«library»
libcifx
«library»
libcifx

7.2. Activity diagram

The Kernel manages the communication channels and provides an interface to those, namely the packages "A4A.Memory.Hilscher_Fieldbus1" and "A4A.Memory.MBTCP_IOSlave".




kernel
application
user
A
Running
Running
Running
Main_Loop
check application watchdog
check communication watchdog
get inputs
Main_Cyclic()
Map_Inputs()
Process_IO()
Map_Outputs()
return
set outputs
housekeeping
Quit ?
B


kernel
application
user
B
Finalization
Finalization
Finalization
Closing()
nothing
stop communication
De-initialization

7.3. Modbus TCP Server Configuration

"../../src/a4a-application-mbtcp_server_config.ads"

```
package A4A.Application.MBTCP_Server_Config is

   ------------------------------------------------------------------
   --  Modbus TCP Server configuration
   ------------------------------------------------------------------

   package Server is new A4A.MBTCP_Server
     (
      Coils_Number           => 65536,
      Input_Bits_Number      => 65536,
      Input_Registers_Number => 65536,
      Registers_Number       => 65536
     );

   Config1 : aliased Server.Server_Configuration :=
     (Server_Enabled           => True,
      Debug_On                 => False,
      Retries                  => 3,
      Server_IP_Address        => To_Bounded_String ("127.0.0.1"),
      Server_TCP_Port          => 1503); --  ①

end A4A.Application.MBTCP_Server_Config;
```

① Modbus TCP Server port : 1503 (default)

7.4. Fieldbus Configuration

"./src/cifx/a4a-application-configuration.ads"

```
package A4A.Application.Configuration is

   ------------------------------------------------------------------
   --  Main Task Configuration
   ------------------------------------------------------------------

   Application_Main_Task_Priority : System.Priority :=
     System.Priority'Last - 10;
   --  System.Default_Priority;

   --  can be cyclic
   --  Application_Main_Task_Type     : constant Main_Task_Type := Cyclic;
   Application_Main_Task_Delay_MS : constant := 50;

   --  or periodic
   Application_Main_Task_Type : constant Main_Task_Type := Periodic;
   Application_Main_Task_Period_MS : constant := 50;

   ------------------------------------------------------------------
   --  Periodic Task 1 Configuration
   ------------------------------------------------------------------

   Application_Periodic_Task_1_Priority : System.Priority :=
     System.Default_Priority;
   Application_Periodic_Task_1_Period_MS : constant := 500;

   ------------------------------------------------------------------
   --  Fieldbus 1 Configuration
   ------------------------------------------------------------------
   --  Alias like "DPM"
   --  or board name like "cifX0" or "TCP0_cifX0"

   Config1 : aliased Fieldbus_Configuration :=
     (Fieldbus_Name      =>
        Fieldbus_Name_Strings.To_Bounded_String ("cifX0"), --  ①
      Fieldbus_Channel   => 0, --  ②
      Board_Name         => (others => Character'First),
      Board_Alias        => (others => Character'First),
      Channel_Handle     => Channel_Handle_Null,
      My_Channel         => null
     );

   ------------------------------------------------------------------
   --  Fieldbuses Configuration
   ------------------------------------------------------------------

   --  Declare all Fieldbuses configurations in the array
```

```
   Fieldbuses_Configuration : Fieldbus_Configuration_Access_Array :=
     (1 => Config1'Access);

end A4A.Application.Configuration;
```

① Channel configuration : board name

② Channel configuration : channel number

7.5. User objects Definition

"../../src/hilscherx/a4a-user_objects.ads"

```
package A4A.User_Objects is

   ----------------------------------------------------------------------
   --  User Objects creation
   ----------------------------------------------------------------------

   First_Cycle : Boolean := True;
   Output_Byte : Byte := 0;  --  ③

   Tempo_TON_1      : TON.Instance;
   --  My Tempo TON 1

   TON_1_Q          : Boolean := False;

   Cmd_Byte      : Byte := 0;  --  ①
   Pattern_Byte  : Byte := 0;  --  ②

end A4A.User_Objects;
```

① 8 Input bits are read that form the Command byte, which is also output in 8 Coils.

② 8 Input bits are read that form the Pattern byte.

③ 8 Coils are written that reflect the Output byte.

7.6. User Functions

"../../src/hilscherx/a4a-user_functions.adb"

```
package body A4A.User_Functions is

   ----------------------------------------------------------------------
   --  User functions
   ----------------------------------------------------------------------

   procedure Map_Inputs is  --  ①
   begin

      Cmd_Byte     := Hilscher_Fieldbus1.Inputs (1);
      Pattern_Byte := Hilscher_Fieldbus1.Inputs (0);

   end Map_Inputs;

   procedure Map_Outputs is --  ②
   begin

      Hilscher_Fieldbus1.Outputs (1) := Cmd_Byte;
      Hilscher_Fieldbus1.Outputs (0) := Output_Byte;

   end Map_Outputs;

   procedure Map_HMI_Inputs is
   begin

      null;

   end Map_HMI_Inputs;

   procedure Map_HMI_Outputs is
   begin

      Bytes_To_Word (LSB_Byte => Hilscher_Fieldbus1.Inputs (1),
                     MSB_Byte => Hilscher_Fieldbus1.Inputs (0),
                     Word_out => MBTCP_IOServer.Input_Registers (0));

      Bytes_To_Word (LSB_Byte => Hilscher_Fieldbus1.Outputs (1),
                     MSB_Byte => Hilscher_Fieldbus1.Outputs (0),
                     Word_out => MBTCP_IOServer.Input_Registers (1));

   end Map_HMI_Outputs;

   procedure Process_IO is --  ③

      Elapsed_TON_1 : Ada.Real_Time.Time_Span;

   begin
```

```
      if First_Cycle then

         Output_Byte := Pattern_Byte;

         First_Cycle := False;

      end if;

      Tempo_TON_1.Cyclic (Start   => not TON_1_Q,
                          Preset  => Ada.Real_Time.Milliseconds (1000),
                          Elapsed => Elapsed_TON_1,
                          Q       => TON_1_Q);

      if TON_1_Q then

         case Cmd_Byte is

         when 0 =>
            Output_Byte := ROR (Value => Output_Byte, Amount => 1);

         when 1 =>
            Output_Byte := ROL (Value => Output_Byte, Amount => 1);

         when others => Output_Byte := Pattern_Byte;

         end case;

      end if;

   end Process_IO;

end A4A.User_Functions;
```

User functions are defined to :

① get the inputs from the channel,

② set channel ouputs.

③ process the data.

7.7. User Application

"../../src/hilscherx/a4a-application.adb"

```
package body A4A.Application is

   procedure Cold_Start is
   begin

      null;

   end Cold_Start;

   procedure Closing is
   begin

      null;

   end Closing;

   procedure Main_Cyclic is
      My_Ident : constant String := "A4A.Application.Main_Cyclic";
   begin

      Map_Inputs;  --  ①

      Map_HMI_Inputs;

      --  Playing with tasks interface
      Main_Outputs.X := Main_Inputs.A;
      Main_Outputs.Y := Main_Inputs.B;
      Main_Outputs.Z := Main_Inputs.C;

      Process_IO;  --  ②

      Map_Outputs;  --  ③

      Map_HMI_Outputs;

   exception

      when Error : others =>
         A4A.Log.Logger.Put (Who  => My_Ident,
                             What => Exception_Information (Error));

         Program_Fault_Flag := True;

   end Main_Cyclic;

   procedure Periodic1_Cyclic is
      My_Ident : constant String := "A4A.Application.Periodic1_Cyclic";
```

```
   begin

      --  Playing with tasks interface
      Periodic1_Outputs.A := not Periodic1_Inputs.X;
      Periodic1_Outputs.B := Periodic1_Inputs.Y + 2;
      Periodic1_Outputs.C := Periodic1_Inputs.Z + 1;

   exception

      when Error : others =>
         A4A.Log.Logger.Put (Who  => My_Ident,
                             What => Exception_Information (Error));

         Program_Fault_Flag := True;

   end Periodic1_Cyclic;

   function Program_Fault return Boolean is
   begin
      return Program_Fault_Flag;
   end Program_Fault;

end A4A.Application;
```

The application cyclically :

① gets the inputs from the channel,

② processes the data,

③ sets channel ouputs.