Bernd Seeberger

infoteam Software GmbH

CANopen based Function Blocks for Motion Control

Abstract: Motion Control applications need fast and reliable communications to guarantee synchronised operation. CANopen provides the standardised protocol for CAN based controllers to build up heterogeneous networks of PLC's and motion controllers.

The distribution of function blocks within a multi-vendor network of co-operating controllers requires a standardisation of programming languages. IEC 1131-3 established a mature standard in the market since some years.

This paper reports work in PLCopen defining a set of standard function blocks and the corresponding software tools for portable motion control applications. Programs written for smart CAN devices use either graphical or textual languages. Ladder or Function block Diagrams Structured Text or Instruction List

This report demonstrates that PLCopen Portability Level specifications for IL enable even the portability of motion control function blocks. It is the prerequsite for reusing software in control automation.

1. Foreword:

The Taskforce Motioncontrol is currently working on a draft to define functionblocks for motion control. At a first look, it seems that IEC languages may not solve motioncontrol needs, since motioncontrol is mainly sequential, whereas the IEC languages are preferable used for cyclic programs. Nevertheless, you will see that IEC 1131-3 languages provide a rewarding tool for programming motion controllers. Still a bit of rethinking is necessary if you want to use a cyclic approach for motion control tasks but this doesn't mean any restrictions to the programmer, on the contrary expands the possible solutions for a motion control task and makes specific tasks even simpler.

2. Introduction:

The goal of the Motioncontrol Taskforce:

Description of a set of function block's interfaces and behaviour in a formal way for handling motion control functions for users of programming systems.

There are several conditions which must be considered when defining a set of Functionblocks, which should be used in an IEC 1131-3 environment. The development process must enclose the programming and the configuration of motion control FB's. There should also be ways to initialise the Functionblocks, as well as the possibility to access the Functionblocks during run time, e.g. Onlinedebugging.

The hirachy is carried out as follows:

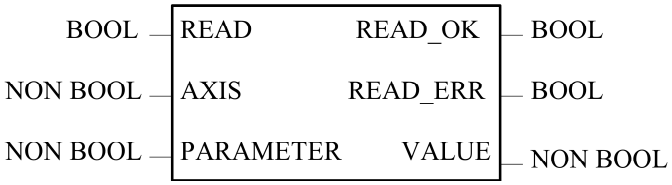
->Program
    ->Drive interface
        ->Position, velocity or current loop control

The Program represents the highest level and is written in an IEC 1131-3 compliant language. The drive interface contains the I/O mapping of drive parameters and hardwired axles. The current loop control remains inside the controller next to the drive and it's specification is part of the CAN Proposal 402.

3. The definitionprocess

The Definition of Functionblocks is carried out in two steps.
First, define a basic set of Function Blocks. Out of the basic function blocks, develop higher level Functionblocks. The range of Functionblocks varies from initialisation, Conversion, single-axis, multiaxis control... to error handling and safety like quick stop e.t.c..

There must also be a way to retrieve data from an axis, like the current position. During the meetings of the Task Force Motioncontrol, it was concluded to define special Functionblocks for that purpose. The following functionblock is an example from Intramat, which can be used to retrieve any data from an axis.



This function block reads data from the axis. PARAMETER defines the type of data requested. The way of addressing and the definition of Values are proprietary (if no standardization is made).

Having the Functionblocks run in several modes is also an important point. For example, you could have a security mode during which all the Functionblocks behave different. No conclusion was done on this point so far.

4. *How to use Motioncontrol Functionblocks in an IEC program.*

How to represent and access the MC Data?

First of all, a Motioncontrol Functionblock is a type. It could be either a struct or a Functionblock. In both cases Motioncontrol specific values could be assigned to the struct or the Functionblock, which enables the IEC program to access certain data.

First apporach: Define MC Data by a structure.

MC1 is a type of an axis (Motioncontroller) which is defined as a struct. The elements Xpos, Ypos... designate specific axis data, which is now accessible.

```
TYPE
   MC1 : STRUCT
            Xpos   : INT;
            Ypos   : INT;
            Torque : INT;
            .
            .
            .
   END_STRUCT;
END_TYPE
```

Second approach: Define MC Data with an FB.

The second approach shows the definition of an axis type as a function block. The advantage, this Functionblock could consist of several basic functions already. On the other hand, some axles would require a lot of parameters e.g. up to 150, which would lead to huge FB's.

```
FUNCTION_BLOCK MC1
        VAR_IN_OUT
                Xpos    : INT;
                Ypos    : INT;
                Torque  : BOOL;
                .
                .
                .
        END_VAR
END_FUNCTION_BLOCK
```

Instantiate the Axis on Programlevel.

An instance of an Axis is declared on Programlevel as follows.

```
PROGRAM
        VAR_GLOBAL
                Axis : MC1 AT%...;
        END_VAR
```

The AT% construct designates an external hardwareaddress. This hardwareaddress represents the actual axis, which is wired to the PLC. Defining the Axis as global makes it visible throughout the whole program.

What happens if you would like to access this axis from inside another FB. Well, you will have to define the Axis as an external variable in the declaration block of the Functionblock you intend to use it.

```
VAR_EXTERNAL
    Axis : MC1;
END_VAR
```

Note: Now, the Axisinstance can be accessed anywhere in the Program, even if called within FB's instanciated in other FB's.

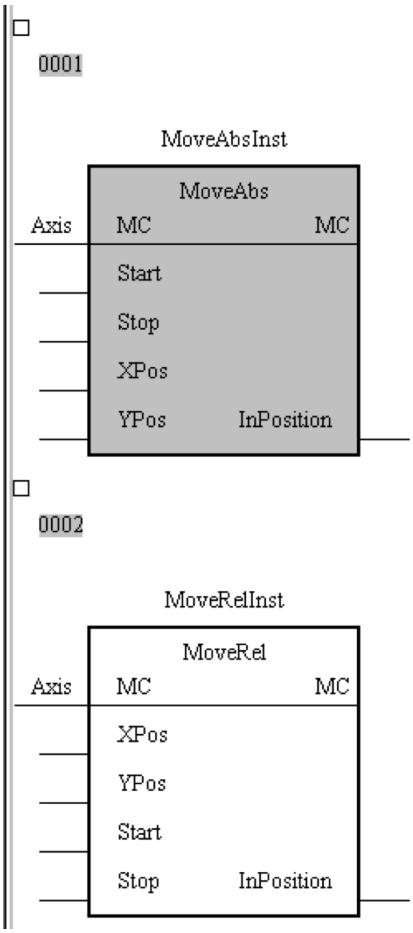
How to apply several methods (FB's) to one axis instance?

One big issue during the Task force Motion control meetings was how to assign several FB's to one axis instance. Lets assume you have an axis which should perform homing and relative movement. In this case you actually deal with two different functions, each defined in a single Functionblock. Both functionblocks have to be applied to the same axis in order to perform the requested movement. Since you want the Functionblocks to be reused and instanciated several times in one program, they must not have any hardwareaddress information. One possible outcome of the meeting which I will present in this paper is to pass the axis instance as an parameter. Like in C++ where you pass the instances of objects, IEC lets you pass the instances of functionblocks and structures, even if they are mapped to a hardware like the axles.

The following example should give you a brief overview on how this mechanism works.
The Motioncontrol functionblock (MoveRel) receives the axisinstance MC of type MC1 as a parameter. Thus, the methods can be applied to a specific axisinstance.

```
FUNCTION_BLOCK    MoveRel
        VAR_IN_OUT                          (* Pass by reference *)
                MC              : MC1;      (* Axis Instance *)
        END_VAR
        VAR_INPUT
                Start           : BOOL;     (* Parameters and methods *)
                Xdist           : INT;
                .
                .
                .
        END_VAR
        VAR_OUTPUT                          (* Acknowledge *)
                InPosition      : BOOL;
        END_VAR
END_FUNCTION_BLOCK
```

The following diagram shows how an IEC 1131-3 program would deal with the passing mechanism. MoveAbsInst and MoveRelInst are instances of axles. Axis is the axisinstance, which is passed to both functionblocks.



Instantiate the axis (Axis), and the MC functionblocks (MoveRelInst, MoveAbsInst).

```
VAR
        Axis        : MC1;
        MoveRelInst : MoveRel ;
        MoveAbsInst : MoveAbs ;
END_VAR
```

The axisinstance (axis) is passed to the MC functionblocks.

4. List of acknowledgements.

1. Minutes of meeting Atlas Copco Controls, infoteam 1997 in Penthaz.

2. 1995, Karl-Heinz John, Michael Tiegelkamp,SPS-Programmierung mit IEC 1131-3, Springer-Verlag Berlin.

3. IEC 1131-3 International Standard 1993-03, IEC.

4. Basic Set of Function Blocks proposal, Atlas Copco Controls, 1997.

5. CiA Draft Standard Proposal 402 Revision 1.0, CANopen Device Profile for Drives and Motion Control, 1997.