DCAF Generic Motion Plugin Project

Requirements Specification

v. 0.1

August 25th, 2017

NI

Introduction

* Purpose

This document is to describe the architecture of the DCAF Softmotion Module. This document does not explain the DCAF architecture – please view the DCAF documentation.

* Scope

The DCAF Softmotion module is aimed towards simplifying the experience of creating embedded motion systems. Through DCAF and this module, the user would not need to write motion specific code but instead send commands through tags to the motion module.

* Overview
* Reference Material

This document references the DS 402 Motion over EtherCAT standard.

* Definitions and Acronyms

DCAF – Distributed Control and Automation Framework

Sequencer – code module that choses what steps the system needs to take. For example, the Sequencer checks a safety input then moves a motor to close a valve.

Axis – the software description of the motion hardware and its control system. The system commands a Moves on the Axis, which then communicates with the hardware as necessary.

Coordinate System – a logical grouping of axis. Generally they describe a physical component (such as a gantry) or a full system (all of the axis in a machine). They often require synchronization.

* System Overview

The DCAF Softmotion Module is added to the configuration editor as any other module. It comes with a series of inputs and outputs for commands, parameters, results and status updates. This module will implement the basic kinds of moves and use the terminology of the DS 402 standard. This is because the DS 402 standard defines a tag based command scheme which fits the DCAF use-case.

Each instance of the module commands one axis or coordinate system.

* System Architecture
* Architectural Design

This project contains 3 main pieces.

The first is the implementation of the moves and of the concept of an axis. This class is made to be overridden if you need to implement a different, non-softmotion axis.

The second is the motion module itself. This module implements a state machine similar to the DS-402 one.

The third is a set of tools and APIs to make working DCAF Tags easier. Because the State Machine uses a DS 402 styled Controlword and Statusword, the project contains tools to go from a status to bits. Similarly, an API is created to facilitate passing parameters to the module and reading the results.



* Design Rationale

The project leans heavily on the DS 402 standard because it is a tag-based motion scheme. This means that a new communication scheme does not need to be created, and users familiar with the DS 402 standard should more easily understand this scheme. However, instead of creating a large amount of tags for the PDOs/SDOs, an array as Parameters and one as Results will be used. The Sequencer will then write and read those tags to switch the target mode or get motion I/O info, among other events.

* Data Design
* Data Description

The Module needs access to 5 main pieces of data. First, it needs to have knowledge of what **Axis** it is communicating with. This is set up in the DCAF Configuration Editor.

Second and third, a Controlword and Statusword are used to send commands to the module and receive status information back. These are done as per the DS402 standard.

Fourth, a parameter array contains information on the command to execute when the Controlword contains a command to begin a move, or to read the parameters. The first element in the array is the Mode (Equivalent to PDO XXXX), the rest of the elements are interpreted as per this first element.

Fifth, a results array contains information on the system. The first value is used to decide how to interpret the rest of the values. On every iteration where the Controlword is not asking the motion module to give back a specific result, it will return the status of the Motion I/O and the position, velocity and acceleration of the axis.

For the fourth and fifth element, the arrays will concatenate the information on each axis.

* Component Design

*In this section, we take a closer look at what each component does in a more systematic way. If you gave a functional description in section 3.2, provide a summary of your algorithm for each  function listed in 3.2 in procedural description language (PDL) or pseudocode. If you gave an  OO description, summarize each object member function for all the objects listed in 3.2 in PDL or pseudocode. Describe any local data when necessary.*

* Human Interface Design
* Overview of the User Interface
* Screenshots
* Requirements Matrix