Integrating tools to aid the automation of PLC development within the TwinCAT environment

European XFEL

N. Mashayekh, B. Baranasic, M. Bueno, T. Freyermuth, P. Gessler, S. T. Huynh, N. Jardón Bueno, J. Tolkiehn, L. Zanellatto
European XFEL GmbH, Holzkoppel 4, 22869 Schenefeld, Germany

Introduction

Whilst there were tools [1] previously developed in order to aid Programmable Logic Controller (PLC) project generation at the European XFEL, overtime, these tools became harder to manage. Many of the tools were developed in an array of programming languages, and require the PLC developers to become adept in multiple Integrated Development Environment (IDE) and languages. In turn, in order to enhance or add to an existing tool or function, edits would have had to be made across the multiple applications to ensure consistency. This approach can work if there is a well integrated team, however, also caused bottle necks and had a high dependency on all of the tools being kept up-to-date.

A new approach was envisioned where all of the function previously being performed either manually, or via some means of automation, was collated together into a Single Point of Contact (SPoC). Within the backdrop of TwinCAT environment, it was a logical step to build upon this platform by integrating this new functionality and interface into TwinCAT itself via the means of Visual Studio extensions.

Developing Extensions

There are multiple alternatives to customise and enhance the functionality of the Visual Studio to their specific needs. Amongst all of the available customisations, the tool window and toolbar button are two which are noteworthy.

- **Toolbar button:** toolbar buttons (Fig.1,2) are UI elements placed on toolbars within Visual Studio. These buttons trigger specific actions or commands when clicked, therefore they are suited for simpler actions who do not need complex user interface.
- **Tool window:** a tool window (Fig. 3) is a dockable window in Visual Studio that can host various controls and components, offering a customized workspace within the Visual Studio. To create a tool window, typically a WPF user interface is defined which implements the necessary logic to handle user interactions.

Fig.1: TwinCAT3 default toolbar buttons

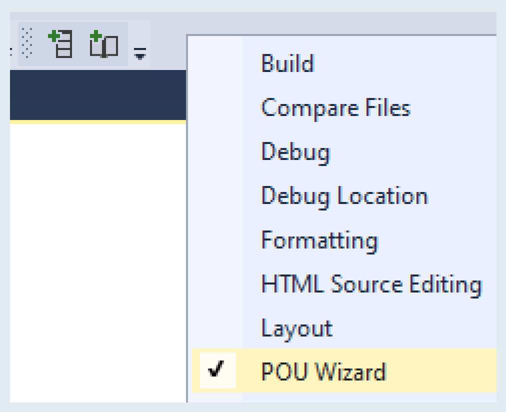
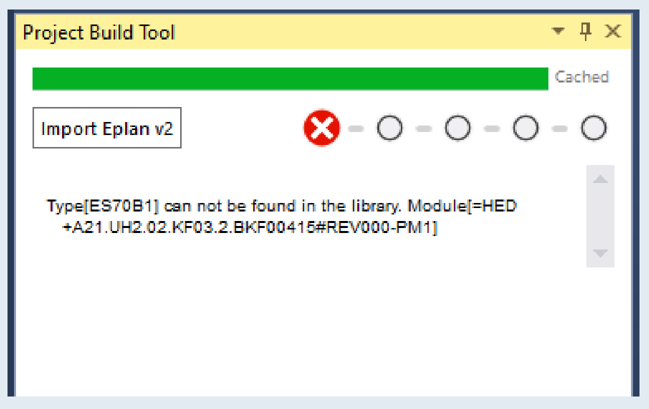


Fig.2: Custom toolbar button



Type[ES70B1] can not be found in the library. Module[=HED +A21.UH2.02.KF03.2.BKF00415#REV000-PM1]

Fig.3: Custom tool window

GENERATION OF PLC CODE USING EXTENSIONS

There are two fundamental approaches to generating PLC code via the Beckhoff Automation Interface.

- Develope or generate new Program Organization Units (POUs) to provide new functionality. (Fig. 4)
- Utilise and modify existing POUs. (Fig. 5)

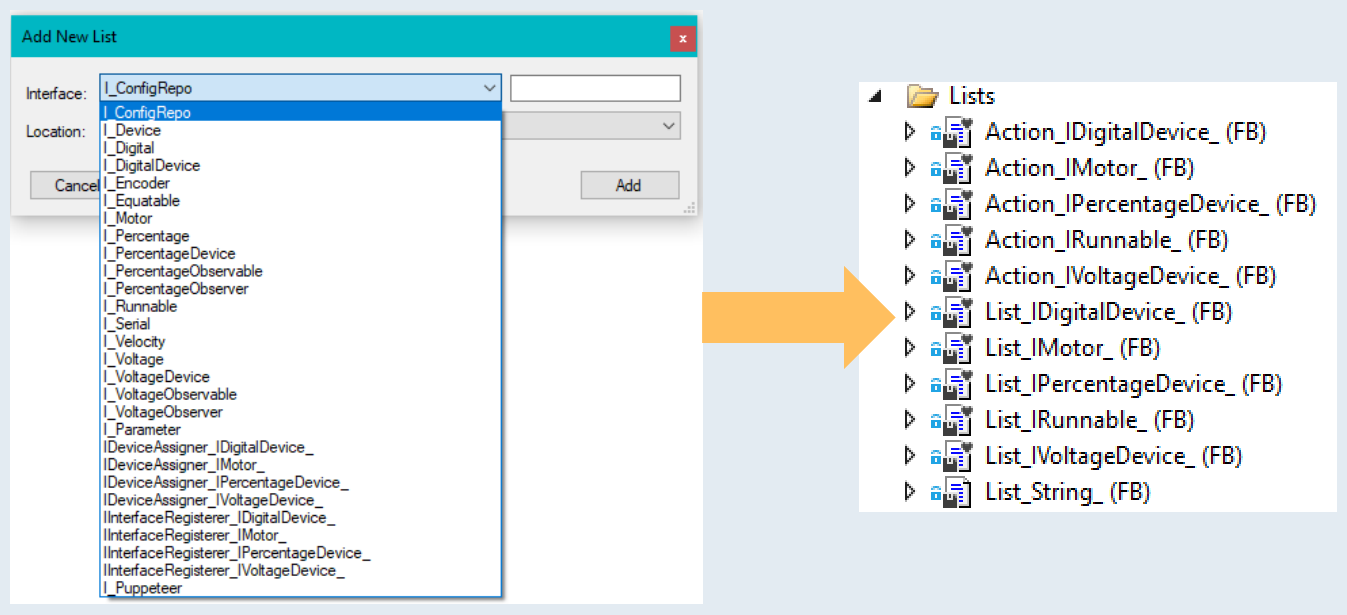


Fig.4: Generating a new List Function Block for a different data type

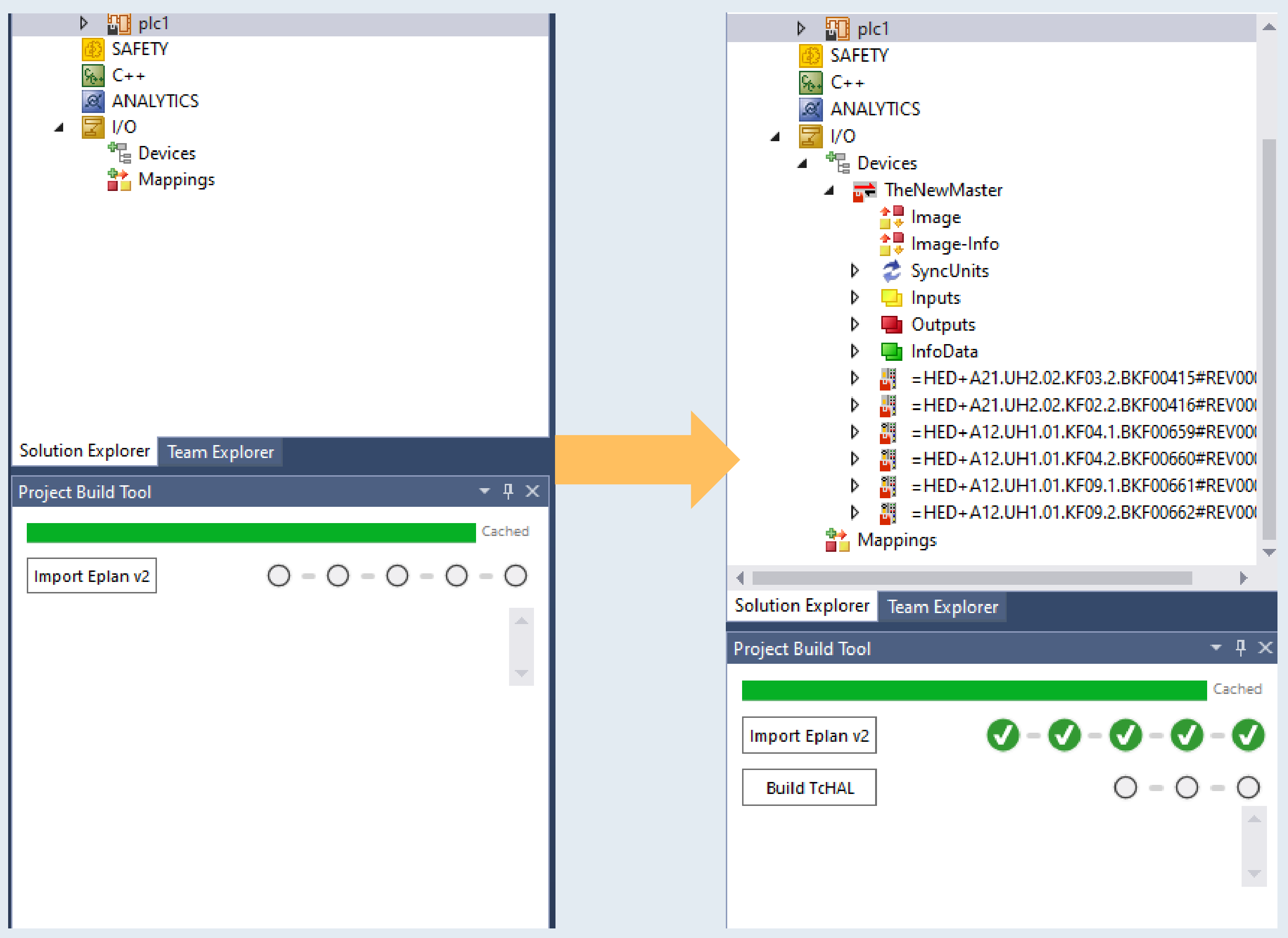


Fig.5: Injecting a new EtherCAT master and devices from EPLAN exports to the Hardware tree

Future Extensions

By integrating the capability to install in-house and required external libraries, our extension aims to create a seamless development experience. Parsing project configuration data ensures that the extension understands the specific requirements of each project, allowing it to intelligently utilize these libraries.

References

[1] S. T. Huynh et al., "AUTOMATIC GENERATION OF PLC PROJECTS USING STANDARDIZED COMPONENTS AND DATA MODELS"ICALEPCS2019, New York, NY, USA, pp 1532-1537.