# Three Use Cases That Benefit From Machine Control Customizations

USE CASE

KINGSTAR

# Introduction

With the increasing demand for custom products and their life cycle becoming shorter and shorter, controllers and machines must be customizable and scalable. Machines have to be designed for modularity and controllers need to be able to support different versions of a machine, from low end to high end. The real-time control platform which is used to develop the kernel of the controller must provide engineers with the capability to add functions that are specific to their company or market. For the machine builder, the controller itself must provide the capability to add or remove modules, or the selection of different grades of hardware. Lastly, the structure of the machine has to be designed in a way to be able to be easily integrated in the factory network and its operator must be able to load and create different tasks to be executed. The most important thing is customization and KINGSTAR is a machine automation software platform, that has been designed specifically with customization in mind. In this tech brief, we will list the most relevant customization types, describe the KINGSTAR platform and its unique features and then study three different use cases that show how KINGSTAR fits into the customization requirements.

# Customization types

## Support multiple hardware versions

Machine builders often provide different versions of their machine. It could be either a low end to high end version, or a small work piece to large work piece version. It depends on the customers' requirements. Since a machine is produced and maintained for at least 10 years, their components, such as drives and I/O boards, are likely to evolve. Having a different controller for each version limits the number of possible versions and is also very costly to maintain. For that reason, controllers are expected to support all the current and even future versions through simple software updates.

Different versions of robot arms

## Support modules

As mentioned, to be able to work in a wider range of situations, machines are built as modular. Controllers are expected to support the optional modules. It is possible to do even better, as controllers can be designed to detect and identify modules in a way that they can be added or removed at any time. It is even possible to add modules to a running machine and have the controller adapt automatically.

## Kernel tailoring

At the controller level, many companies have their own intellectual property and market specific control functions. As an example, for a robot arm controller developer, the hand teaching would be one of those specific functions. These functionalities are often real-time, which means, the real-time application platform the company uses, needs to be extensible enough to allow the controller developer to extend it with

additional control features. This possibility is often a key differentiator for the controller.

## Application tailoring

Just like the control kernel, applications are also designed as modular and make use of plugins and 3rd party tools, such as vision libraries and calibration devices. Therefore, all settings for the machine need to be exposed to plugins through standard protocols or configuration files.

## Machine integration

Most machines need to be integrated in a production cell or a factory network. In the future, OPC UA will allow this in a standardized way, but at the moment each factory has selected its own networking protocol, which means machines must support a wide range of protocols for machine to machine, and machine to SCADA communications. For production cell integration, it is also often necessary to add extra I/Os to the machine for its synchronization in real-time with other machines.

## Work task

Finally, as products change very fast and multiple products are produced on a single line, it is necessary that the operator is able to select different tasks. This is often done using script files such as G-Code or CAD drawings. As machines are asked to produce "batches of 1", where the task can change for each piece, it becomes necessary to implement a communication with a task database, where each task is automatically downloaded, based on an ID read from the piece to process.

## KINGSTAR automation platform

As mentioned earlier, KINGSTAR is a software platform for machine automation. It targets manufacturers of controllers and machine builders. KINGSTAR is a division of IntervalZero, a company based in the US, with offices worldwide, specialized in real-time and embedded systems for many decades.

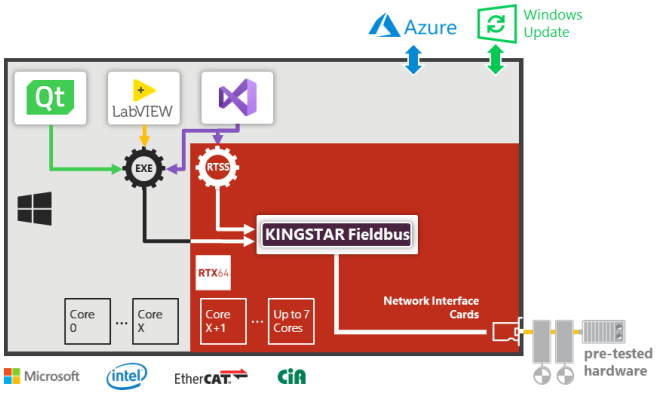
Historically IntervalZero has been a company focused on Windows based machine controllers.
For over twenty years IntervalZero developed and maintained the RTX product line, providing a real-time extension for Windows. RTX is used with control and communication boards to build machine
controllers. As the power of processors increase and new Ethernet-based fieldbus protocols emerge, customers continue to ask for software protocols and software-based control logic. Therefore, IntervalZero developed the KINGSTAR automation software platform for building smart machine controllers.

The KINGSTAR offer consists of five components:

- **KINGSTAR Fieldbus** (real-time EtherCAT® Master)
- **KINGSTAR Motion** (motion control)
- **KINGSTAR PLC** (software Programmable Logic Controller)
- **KINGSTAR Vision** (real-time GigE Vision solution)
- **KINGSTAR IoT** (IoT-enabled platform)

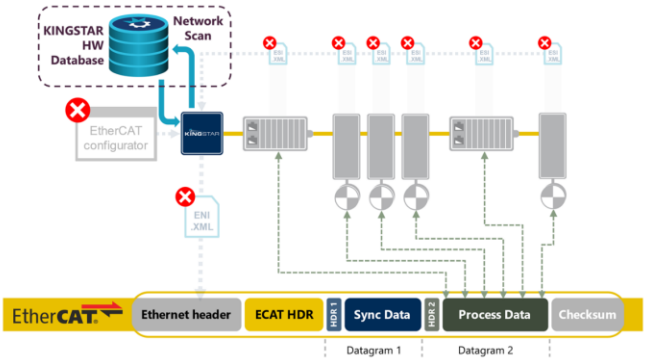
KINGSTAR runs on the 64-bit version of the real-time extension, RTX64, which is the core component of the KINGSTAR platform and transforms Windows into a real-time operating system.

Running with Windows 10 64-bit RTX64 allows real-time applications to be developed with C/C++ in Visual Studio. It can be used on a wide range of general-purpose computers, and is deployed in many different industries, such as automation and robotics, but also medical, defense and simulators.

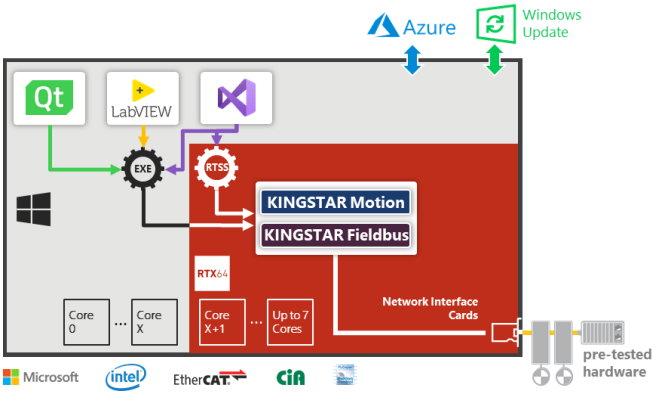



KINGSTAR Fieldbus architecture

KINGSTAR fieldbus implements a Plug & Play EtherCAT® stack on RTX64. As mentioned in our whitepaper which compares the five most important fieldbuses on the market, IntervalZero believes EtherCAT® is the best protocol for machine automation and the KINGSTAR product is based on it. To provide more flexibility to applications, IntervalZero took advantage of the EtherCAT® bus scan capabilities, to build an automatic configuration feature, which allows the same application to run with different hardware configurations. The main benefit of this automatic configuration is, that it supports all the major servo-drive and I/O hardware brands and users can add support for new EtherCAT-based hardware without updating KINGSTAR. In addition, the fieldbus layer provides direct access to variables, as if they were local, completely hiding the fieldbus from applications.



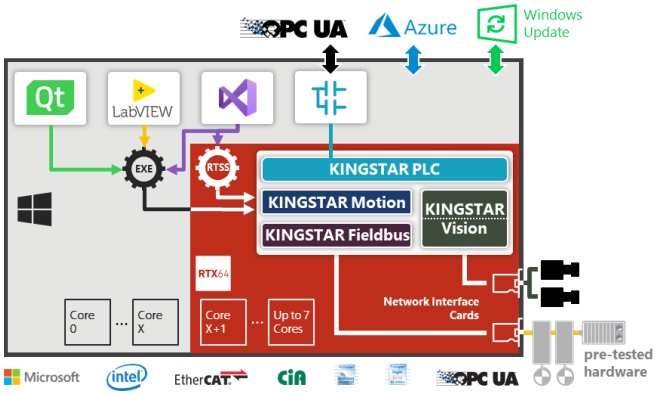

KINGSTAR Fieldbus Automatic configuration feature




KINGSTAR Motion architecture

To further round out the platform for smart machine control, KINGSTAR also provides a software motion component. KINGSTAR Motion complies with the PLCopen Motion Control standard specifications for point to point, synchronized, group motion, blending and kinematics. With modern processors and the optimized motion equations in KINGSTAR, it is possible to control a large number of axes at fast cycle times. For example, applications can use 20 axes with 125us cycle

time or 60 axes with 500us cycle time. Each axis can use a different brand of hardware and have its own control mode. Communication with the drives are based on cyclic synchronous modes, the interpolation is done in the controller, but the PID can be either in the controller or in the drive. The motion algorithms allow modifications of the motion profile while the axis is moving. The synchronization supports electronic camming, gearing and group motion with linear, circular and helical moves. These KINGSTAR Motion features are very flexible as a CAM or gear master axis can have multiple slaves, and itself be a virtual axis or even the slave of another axis. These motion features are available to both real-time and Windows applications.




KINGSTAR PLC with KINGSTAR Vision architecture

The third component is KINGSTAR PLC which provides a fully-featured and integrated software PLC based on an open and accessible RTOS – RTX64 from IntervalZero. KINGSTAR PLC also includes add-on or third-party components for motion control and machine vision that are managed by a rich user interface for C++ programmers and non-developers alike.

KINGSTAR Vision is a real-time GigE Vision® stack, which allows customers to develop vision-directed motion control using OpenCV (open-source libraries) on a Windows PC. KINGSTAR Vision is a comprehensive collection of software tools for developing machine vision, image analysis and medical imaging software applications on GigE Vision® and many other camera interfaces. It includes tools for every step in the process, from application feasibility, to prototyping, through to development and ultimately deployment.

Last but not least, KINGSTAR IoT for Windows PC adds IoT functionality to your machine control by capitalizing on the most open machine automation software platform. For more information on this topic, several white papers are available on our web site www.kingstar.com Especially our Achieving Industry 4.0: Four Critical Features for Smart Machine Automation whitepaper presents an in-depth analyses of this topic.

Now that we introduced you to KINGSTAR, our software platform for machine automation, and went over the most important types of customization, let's explore three different use cases, where customization was a requirement, and see how KINGSTAR fulfills them.

# Use case examples

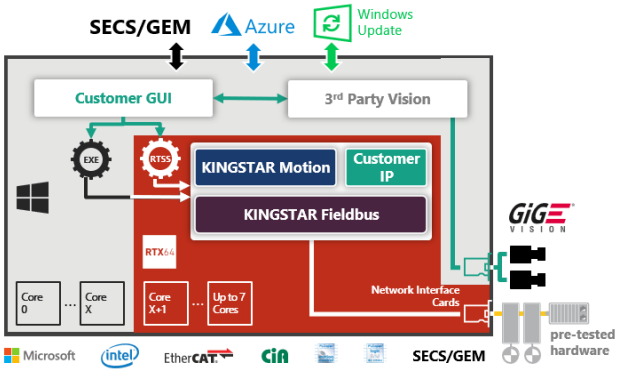
## PCB Processing Machine

The first use case we will present is a PCB (Printed Circuit Board) processing machine. This machine takes a PCB, loads it into a treatment chamber and inspects the result. The machine is designed as modular, as there are multiple possible treatments, such as exposure units for instance. It also has optional components, such as flipping the board to process both sides and inspecting the result.

Companies selected KINGSTAR for this kind of machine, because the platform provided them a flexible and easy integration. First, they selected EtherCAT as the fieldbus for the machine, and by taking advantage of the plug & play capability of our master, their controller was able to detect which modules were connected and start accordingly. There is no need to configure the controller when building a machine, and for the customers it is possible to purchase and add extra modules to machines which are already deployed. Adding a module is as simple as connecting the power and rj45 cables to the main machine.

While loading, flipping and installing, the board uses standard motion. However, the processing uses specialized hardware and therefore requires a special control loop, that is the intellectual property of the machine builder. As KINGSTAR includes the RTX64 RTOS and supports multiple applications, the control engineers are able to develop a library that connects to hardware through EtherCAT, controls it and expose an interface to the machine application.

Having Windows in the controller facilitates the integration of 3rd party tools and software. In this case, it has been a vision library used to inspect the boards. It also allowed them to provide flexibility in the factory using PCI boards with Windows drivers, which support all common protocols.




Integration of Third Party software, Customer IP and Customer GUI to KINGSTAR

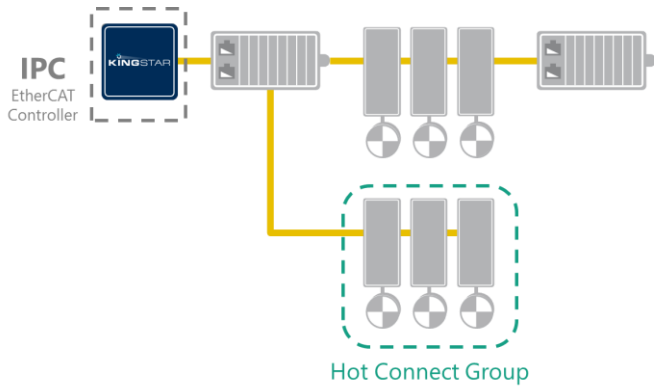
Finally, the machine and application is designed to be able to perform different operations on the boards. All boards are not the same, therefore operations may differ. This means, they must provide a standard task format to the operator to be able to import and select tasks to be executed.

## Test bed

For this second use case, we look at customers in automotive, avionics and other industries who develop test beds. Per default due to their modular design, these machine must be very flexible to meet generic test requirements or turnkey test cell requirement. They must be able to run simple or highly complex tests in standalone or in collaboration with other test beds in a testing center. As such they use a wide range of test sources, many of them being internally designed, and sensors to read the results.

Just like the PCB processing machine, test beds are designed to be modular and include multiple specially designed loops to control the hardware. The main technical particularity of test beds is, that when the devices to be tested change, the hardware used to test them may evolve too. Therefore, modules may be developed after the machine was deployed, and added to it. This means adding hardware and software on a machine that may still be running.

Our customer was able to do this using the hot connect functionality of EtherCAT, in combination with the plug & play capability of KINGSTAR. Hot connect is a feature of EtherCAT, where hardware can be plugged into a running network and started, while the rest of the hardware continues to work. Usually all hardware configuration is stored in a network configuration file, so all the possible modules have to be known before the machine starts. However, EtherCAT supports bus scanning and KINGSTAR starts devices from a database instead of a configuration file. This means, new hardware and configurations can be added on a machine while it is running. As the controller runs on Windows and RTX64, new application files can be added to control this new hardware.




Hot Connect feature

## Robotic arm

The last use case example we will cover in this tech brief, is a robotic arm. We selected this use case as it showcases most of the customization types. A robotic arm is used for many different tasks from pick & place to welding, so it has many different sizes and precision levels, connects with different tools and integrates with lots of different machines. Customization is key in such environment.

Companies never make a single robot, they support at least multiple loads, often different precision levels, and mechanical architectures such as 4-axes, 6-axes and 7-axes robot arms. As the number of combinations can increase quickly with hardware versions, it is imperative that the controller detects and adapts to the connected hardware automatically.

Robotic arms have to have special force controlled functions, to avoid scratching pieces they pick, and to allow hand teaching programming. This means, the controller developers make their own real-time control loops which are often what differentiates them from the competition.

Robots rarely work alone; they have to integrate in a production unit. This can be done in 3 different ways and the controller must support all of them, as the robot arm may often be moved from a production unit to another.

- The first integration mode is the "master control" mode. In this mode the robot controller will control external hardware, generally I/Os and conveyor belts, to avoid using a second controller and having to write two applications. This means, that the robot controller must include general purpose motion as well as the robot specific motion. This is often done by including a PLC runtime in the controller. It must

also allow the end user to connect extra hardware to the controller. This can be done easily with EtherCAT by exposing an "EtherCAT Out" port to users.

- The second mode is the "peer-to-peer" mode. In this mode the production unit has multiple controllers, each of them has its own application and they synchronize over a communication network or I/Os. For this, the controller should allow extra I/Os to be connected by the user and used in the application program. It should also have protocol board options that users can select from.
- The last mode is the "slave mode". In this mode the robot is controlled remotely by the machine it is working with. This is often the case when the robot loads and unloads parts in a CNC. In this case, both machines have to support a common command interface, which most commonly is a PLC. The master machine can then send commands from its own PLC runtime to the PLC runtime of the robot, using an automation protocol such as OPC UA or EtherCAT. With this mode, no application is written in the robot controller. The APIs of the robot are used in the application of the master machine and sent over the network to the robot controller to be executed.

Multiple companies have selected the KINGSTAR platform to build their robot controller, as it is flexible enough for them to add all the components they need, while providing all the non-robotic features for them. The KINGSTAR runtime has a plug & play EtherCAT fieldbus that handles the different types of hardware used by the robot models, and it also allows end users to add extra hardware without needing to understand EtherCAT. It includes standard motion functions, which can be used to control the external hardware, and the controller programmers can develop their own robotic functions to extend them, so that they will be exposed to the PLC and applications. Using the Windows and RTX64 OSes, they can build their user interface and script processing runtime while also offering the KINGSTAR PLC runtime for users who might prefer this option. Software and hardware boards can also easily be added for other communication protocols, both in the real-time and non-real-time environments. Finally, 3rd party tools such as vision or CNC components are readily available to be added to the platform.

For more information, please watch our KINGSTAR webinar that describes How to build a new generation of 6 axis arm robot controller.

# Conclusion

KINGSTAR has been designed taking into consideration open standards and market demands. Customization is probably one of the biggest requirements from both controller makers and machine builders. Other demands, such as modularity, maintenance and remote assistance, or security are strong as well and will be the subject of other use cases in the future. Machines and controllers are more and more complex, and the development of their features often requires multiple teams with different skills. Developing a specific motion algorithm in the controller kernel requires real-time skills and C/C++ programming when a custom Graphical User Interface for the operator needs HMI skills in a PLC programming environment. In between, other languages and environment might be better options, such as .NET for user interfaces dedicated for system integrators for instance. A machine automation software platform must be open and support standards. Another point that should be mentioned, is the ability of the KINGSTAR platform to embrace the Industry 4.0 era. Most machine builders and therefore controller makers will have to consider Industry 4.0 in their design in the future, if it is not already planned.