What's the best choice for building a modular machine?

WHITE PAPER

KINGSTAR

Introduction

With the increasing demand for custom products and their life cycle becoming shorter and shorter, controllers and machines must be modular. 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 to select different grades of hardware. System Integrators in the manufacturing space would benefit from modularity in the exact same way as the machine builder. 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. KINGSTAR is a machine automation software platform, that has been designed specifically with modularity in mind. In this tech brief, we will list the most relevant modularity types, the best practices to build modular machines and then describe the KINGSTAR platform and how its unique features make the difference.

Types of modularity

To best describe the different types of modularity in industrial machines, the following examples show the most common types of modularity we have encountered at our customer sites. The more recent and complex machines use many, if not all of these modularity types in order to be suitable for many different use cases.

Pluggable units

Pluggable units are specific types of machines, that do not work on their own. In the past, these machines would have their own controller and be synchronized to another machine through I/Os. For example, such modules could load a piece, flip a PCB board or tag a piece. They were able to be moved from one machine to another depending on the production. With newer technology, they can now be directly controlled by the main machine as a slave, using a fieldbus.

Optional hardware

Most machines today have optional features, such as engraving, performing tests or special operations. Most of these features require extra hardware to work, so machines are built as a group of independent stations, each station performing a specific task. The machine controller must be able to properly identify and control each station.

Versions of hardware

Many companies have different versions of their machine for different payload sizes and precision levels. These different versions will use different hardware models for drives and actuators, but almost identical control logic. Therefore, a single controller type must be able to support all these different hardware versions.

Optional software

Today's Machines often have optional "software only" components such as task editors, CAD/CAM software in CNCs, or analytics to optimize production speed and predict maintenance needs.

Machine interface

As machines rarely work alone, they need to integrate with other machines, and with the factory SCADA and MES. The fieldbus and data format used to exchange information with other machines and with the factory is decided by the end customer (and not by the machine builder). This means, that within the machine, the interface used by the customer has to be an independent module with different options to be compatible with all customer's factories.

Task interface

Since a machine will most likely produce different products along its lifecycle of deployment, it needs to support different tasks, which are defined by the operator. Each industry and factory may have their own tools and formats to define these tasks, which means the task interpreter also needs to be an independent module with multiple versions, for the machine to be usable in different environments.

Architecture of modular machine

In this section we will describe some of the best practices in terms of hardware and software architecture, that allows industrial machines to be fully modular.

Slave controller or Hardware cell

The difference between a slave controller and a hardware cell is, that within the slave controller a standalone controller is included.

As mentioned before, to allow customers to select the features they need, industrial machines should be designed with separate stations that perform individual processes. Each of these stations are hardware cells which are built to present a simple interface to the main machine, usually an IN and OUT connector for the fieldbus and the power. This allows the machine to be assembled by linking the different cells together.

Some of these cells may or may not be used, depending of the task to perform, which means they can be re-used on different machines. As mentioned previously, they are called "pluggable units". If these units can only be used by specific machines of the vendor, or have very simple logic, they do not need to include a controller, as the controller of the machine they are already plugged in, will be used to control them.

Certain other cells are more complex and can be used in a variety of machines, for instance in a robot loader, which in this case should include a slave controller. A slave controller includes the logic to control its cell but does not run any task or user program, instead it receives the task instructions from the main machine.

Hardware abstraction layer

The control logic must be able to run on different versions of the hardware. This allows the same logic to be used on different payload machines and also keeps the logic independent of hardware evolutions. Since a machine will be built for many years, some change in the hardware will most likely occur. Keeping the logic independent allows corrections and updates to be applied on machines that were deployed with different hardware.

This is now increasingly critical as these updates will most likely include security fixes to the machine and to the factory interface. Having to create a separate update for each hardware version would lead to unnecessary efforts and older machines would most likely not be able to receive the security updates without the abstraction layer.

This also allows more procurement security for machine builders, as they can swap hardware brands in case of potential supply issues. Recent shortages in the semiconductor market make this an important requirement for machine builders.

Modular software

Just like the hardware, the software features must be developed as modules, the software platform must be able to interconnect the internal fieldbus and the different application layers. Within the platform, the hardware abstraction layer controls the internal fieldbus to expose the available hardware cells to the applications.

Different applications are often developed in different programming languages and may require real-time, the platform should therefore run in real-time.
At the same time, it also needs to expose its interface to as many environments as possible.

The platform should also have a loose link to the different applications to keep running normally, even if one application fails. This is critical, since some of the applications may include safety. Allowing one application to impact the whole system would require every application to follow safety development rules.

The software platform should also allow communication between the different applications, as modules often need to share information with each other. Usually a current processing state of each piece is kept that every module will update.

For all modules to work smoothly together, a main application will start, initialize the platform and start the different modules based on configuration files and currently connected hardware. It will control the complete startup and shutdown sequence.

Machine interface

The communication between the machine and the rest of the factory is now a very important feature of any industrial machine. This is true for all machines, but when building a complex system, it can become confusing.

The machine interface must provide remote access to the hardware for maintenance and diagnostics, to the overall machine information for process management and analytics, and to the different features to be able to cooperate with different machines.

To reduce possible attacks against the machine and avoid any accidental interference to the production, access to the hardware should not be direct, access for maintenance and diagnostics should go through the machine controller. This way it can make sure that the machine is in maintenance mode and that the link between the controller and the hardware is not broken. This interface uses a different protocol and software module than the other machine interfaces, because it must be extra secure and allows remote control instead of variable and API access.

The interface for process management and analytics is usually very simple as it periodically collects data. But for the analytics and management to be useful, it must

be possible to act upon the results and therefore modify the priorities, scheduling and calibration of the machine. In addition to periodic access to contextualized data, this interface should have APIs to update the machine.

The complexity of the interface for machine collaboration depends widely on how much the machines will depend on each other, which again depends on the deployment scenario. Instead of having to build a different interface for each project, recent standards propose to merge this interface with the management and analytics interface and give a full description of the machine's features including the status, in which parameters can be modified and the functions that can be called. For this to be safe, different features and parameters may have different access restrictions. Each system in the factory, for example SCADA, MES, analytics and other machines, will use different credentials and have access to the different features they need. The integration no longer requires any modification to the machine interface, only to the definition of the different credentials.

Task file

As machines now change the tasks, they perform very often and may even perform different tasks on different items at the same time. If they are at different stations, tasks cannot require manual definition by an operator. Similar to CNC with G/M code files, the task definition must be stored in a file that can be easily switched, modified and downloaded from servers following the instructions of the factory scheduling system. In some cases, instead of having the scheduler assign tasks to every machine, it assigns tasks to the items using an ID tag and the machines will download their task file automatically after reading the ID from the incoming item.

Technical requirements

To achieve the architecture required by the modular machine, important features are required from the hardware, software and protocols used by the machine.

Fieldbus with hot connect, bus scan and auto-configuration

The first critical element to select is the fieldbus used to connect the controller with the different hardware. As modular machines may often have large options, this fieldbus should allow over a hundred devices to be connected with cycles bellow 1ms as some elements will need high sampling rates.

It should also allow the hardware to be chained and tagged, so hardware cells can be plugged in the back of each other and recognized even when the same hardware is used multiple times. Since many combinations are possible, the fieldbus must allow the connected hardware to be scanned at startup for the controller to adapt automatically to what is connected. As we mentioned, maintenance and diagnostics should be possible remotely through the controller, so the fieldbus should allow for device configuration, diagnostics and update. If the machine has to be maintained or modified while running, for example if parts of the process are very long, the fieldbus should support hot connect, which means removing or adding hardware while running.

As these machines use very different hardware, the protocol must be adopted by many hardware vendors. This also allows machine builders to have multiple hardware options in case of supply issues.

Standard PC

The next important piece is the controller hardware. Modern controllers include both real-time and non-real-time components, meaning the hardware must be a multi-core CPU with network ports. In case of high volumes, it is usually recommended to use a specially designed SoC for the controller. This can sometimes become counter-productive, for example when the controller is used on many different machines. In that case either the SoC needs to support the most complex version of the machine, or many different SoCs are needed, which greatly reduces the volume per SoC and increases the development time for each new machine.

For modular machines, a standard PC is the most suitable solution as different CPUs and RAM can be used on each version of the machine without additional work or design cost. Standard computers allow standard PCI boards to be used, which is important for factory communication.

Subsystem with multiple interfaces

To build a modular software, it is critical to have a central subsystem to interconnect all the elements. Another option would be to create a pyramid with the different modules, that are able to call each other, but this type of architecture is not as flexible as new modules cannot easily be inserted in the middle of the existing stack.

Usual software design recommendations would be, that the interface only takes care of the interconnection between modules, but this would add complexity to the development, because some of the function (fieldbus and motion) are used by almost all the application modules. Therefore, we recommend developing a subsystem as application platform that would include the fieldbus, input / output access and standard motion in addition to the application interconnection. As some of the application modules will be real-time, this subsystem must also run on the real-time side of the controller, which means it should not include the main application, that would run on the non-real-time side of the controller. The software architecture will have a main application that starts the control subsystem which includes the fieldbus, the fieldbus then starts and scans the bus and the main application launches the additional software modules based on the scanned hardware and configuration files.

As the other software modules can be very different, from a real-time process to a visualization tool or data analytics, they may be developed with many different languages and environments, so the subsystem must have interfaces for both industrial real-time environments such as C, C++ and PLC and IT environments like .NET, Java or Python.

Standard communication & control

When a machine is deployed, it is most often added to an existing production line in a factory, so it must interface with the existing machines and factory automation. This means supporting a wide range of standard and proprietary protocols. Therefore, it must be possible to add an optional protocol board, usually a PCI card, to interface with the factory and other machines. To do this without modifying the controller software the machine interface should be based on an open and widely used standard, this way it will be simple to find components that convert from the standard protocol to the factory protocol.

Additionally, every customer will most likely request slight modifications to the logic or to be able to add their own software logic. In most cases this logic is used to control and synchronize the interface with other machines. To avoid having to send engineers to modify the controller for every project, the controller itself should include a standard control environment to allow the customer to add the logic themselves, this environment is usually a software PLC.

Scripted task

Since machines now perform many different tasks at the same time, these tasks must be defined in files which are easy to modify and update. In some cases, the tasks must be editable while they are processed. Instead of a compiled application, it is more flexible to use a scripting language and to provide an editor directly in the controller.

Remote control & maintenance

Finally, it is increasingly important to provide remote access to the machine to allow control and diagnosis and to prepare for maintenance and define the exact tasks, which will be performed on the machine. It is then possible for local engineers without special training on this specific machine to perform the maintenance.

KINGSTAR

After reviewing the different types of modularity and discussing the best practices and technical requirements for building a modular machine, let's explore the KINGSTAR solution.

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:

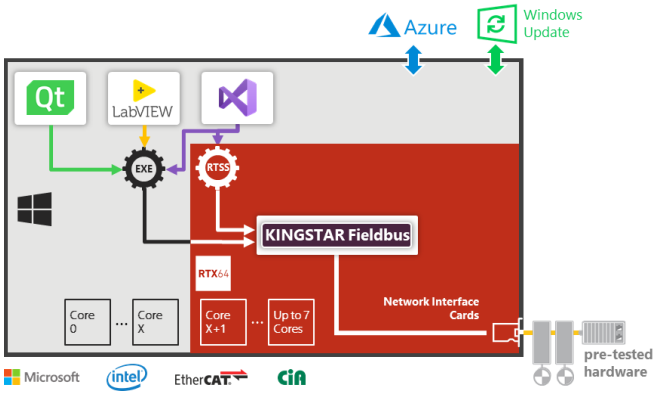
- **KINGSTAR Fieldbus** (real-time EtherCAT® Master)
- **KINGSTAR Motion** (motion control)
- **KINGSTAR PLC** (software Programmable Logic Controller)
- **KINGSTAR Vision** (real-time GigE Vision solution)
- **KINGSTAR IoT** (IoT-enabled platform)

PC-Based real-time control

The first important aspect is the hardware platform, and as explained earlier, a standard PC with a world class operating system brings a lot of benefits.

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.

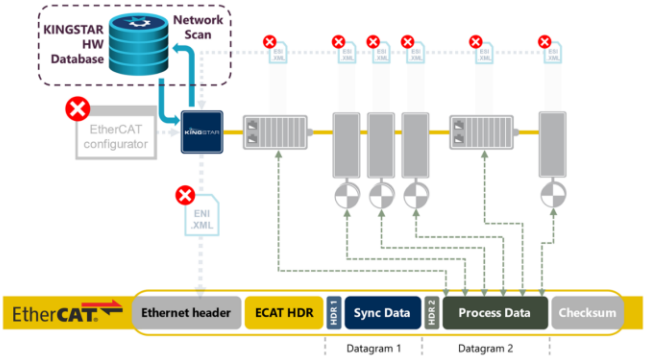
Running with Windows 10 64-bit RTX64 allows real-time applications to be developed with C/C++ in Visual Studio. It can be used on a wide range of general-purpose computers, and is deployed in many different industries, such as automation and robotics, but also medical, defense and simulators.



KINGSTAR Fieldbus architecture

Auto-configured EtherCAT

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 the support of 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.

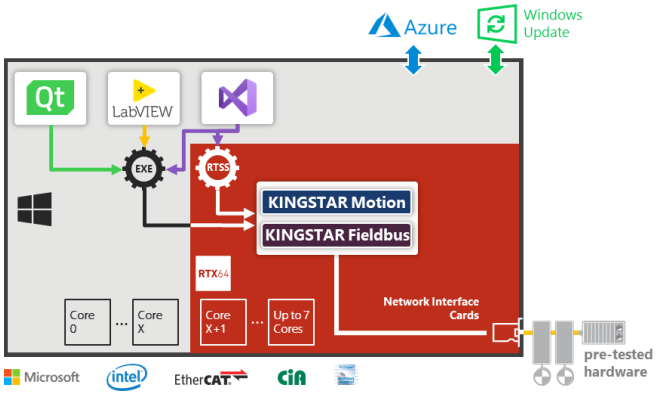


KINGSTAR Fieldbus Automatic configuration feature

Standard-based: EtherCAT, CAN DS402, PLCopen, OPC UA

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 is 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

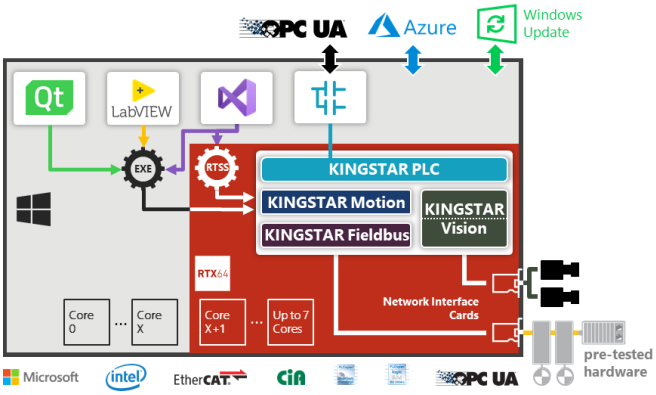
moving. The synchronization supports electronic camming, gearing and group motion with linear, circular and helical moves. These KINGSTAR Motion features are very flexible as a CAM or gear master axis can have multiple slaves, and itself be a virtual axis or even the slave of another axis. These motion features are available to both real-time and Windows applications.



KINGSTAR Motion 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.



KINGSTAR PLC with KINGSTAR Vision architecture

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.

Customizable platform

Besides the modularity, the other top market demand is the ability to customize controllers or machines. We already explored this topic in another white paper and webinar. Some of the technical requirements are very close to what is needed for modularity. A machine automation software platform must be open and

support standards. 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.

Conclusion

KINGSTAR has been designed taking into consideration open standards and market demands. Modularity is probably one of the biggest requirements from both controller makers and machine builders. Other demands, such as customization, maintenance and remote assistance, or security are strong as well and have been already discussed in other whitepapers or webinars as mentioned earlier, or will be the subject of other white papers in the future. 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 have to consider Industry 4.0 in their design, if it is not already planned. Again, supporting standards such as PC based controllers, Windows 10 operating system, OPC UA, Azure, etc… is the best method to prepare your next generation of systems, it could be a controller or a machine, that include IoT services.