



MACRO: A Motion Ring for Distributed Hardware and Centralized Software

It's an increasing dilemma for today's designers of machines employing motion control: as the machines get more complex, the wiring required for traditional highly centralized controllers becomes more and more difficult to deal with, adding to initial cost, multiplying possible failure modes, and even decreasing performance (it can be hard to flex a thick wiring bundle).

Many designers investigate distributed control systems for this reason. The allure of these systems is undeniable: put the intelligence for a particular task out on the machine close to the sensors and actuators. Now the wires from the sensors and actuators can be very short, and any necessary information required to or from the central controller can be combined into a higher-level protocol transmitted over a single cable.

However, most of the work done on the hardware and software has been done for the process control industry and is not optimized for motion control. Process control has neither the high frequency nor the strict determinism required in many motion control applications. Therefore, many of the tools developed for this industry do not support these requirements, either.

As a case in point, look at the "field buses" that are very popular for I/O control these days, with CAN bus and Profibus among the leaders. These are widely used for discrete I/O control, and do a wonderful job of simplifying the system wiring to these I/O points, while maintaining typical PLC scan rates. But it is important to remember that typical PLC scan rates are both slower and less regular than those required for using feedback in motion control.

A common answer to this problem has been to distribute all of the servo-rate tasks – interpolation, loop closure, and possible motor phase commutation – into the distributed node, where they can be done quickly and deterministically. The communication to and from the node is at a higher level, which can often permit lower frequency and determinism.

Many designers wish to make a motion axis "just another node" on their I/O field bus, and this can work well in some applications, particularly if the motion is not coordinated. However, it is important to remember the limitations of such an approach. For an axis drive to be able to handle the servo-rate tasks itself, it must have dozens of set-up parameters: set-points, speeds, accelerations, numerous gain terms, safety limits, etc.

These are typically shoe-horned into a field bus data structure that was never intended for this purpose. Even if most of these values stay constant for an application, setting those items that do not (e.g. end-points and speeds), is typically far more cumbersome than in a centralized controller. Many users find they have simply traded hardware wiring complexity for software and communications complexity.

In applications requiring tight coordination between axes, most I/O fieldbuses do not have the required synchronization mechanisms to assure that coordination. If the motion of multiple axes is simultaneous, but not tightly coordinated, there can still be problems. If one axis faults out, can this information be transferred to the central controller and then to the other axes in time to prevent a problem? (This is not a hypothetical issue. In Silicon Valley recently, a cassette full of silicon wafers was smashed in just such an incident, destroying over a million dollars of product.)

Even with purely sequential moves, this type of distributed approach can be cumbersome. The need to confirm that move “n” has properly completed so that move “n+1” can start safely can take a great deal of time on a distributed system over a typical I/O field bus. On many systems, these delays can add 30% to overall cycle time compared to a highly centralized system.

How, then, do you get the hardware benefits of a distributed system, while maintaining the software benefits and performance of a centralized system? To do this, you do need to distribute your interface electronics, but you also need to keep your control software as centralized as possible, doing almost everything in a single processor, where interactions are fast, deterministic, and easy to code. This requires that the central processor execute high-frequency, hard real-time tasks like the servo-rate tasks, which in turn requires that the communications link be high bandwidth and deterministic.

The underlying technologies to support such a link did not exist at reasonable cost until recently. But during the 1990s, tremendous investment was made in networking technologies, driving their capabilities up and their costs down. One key base technology was 100-megabit-per-second (Mbps) Ethernet, developed for high-end office networking systems. While office networks use a software protocol that is not deterministic, the underlying hardware technologies can be used in deterministic systems.

A second key technology brought to the mainstream was high-bandwidth fiber-optic transmission. Fiber-optic communication eliminates electrical transmission waveform problems and all issues of electromagnetic interference during transmission. It also provides automatic isolation between nodes, eliminating possible ground-loop problems.

In the mid-1990s, Delta Tau Data Systems took a detailed look at both the needs of industrial machine builders and the available technologies from a variety of fields that could be used to meet these needs. Delta Tau developed the MACRO (Motion And Control Ring Optical) communications system as a result. MACRO builds on the underlying Ethernet and fiber-optic hardware technologies with a uniform packet structure, a hard real-time update, and a simple, deterministic master/slave protocol to create a communications link that permits highly distributed hardware with highly centralized software.

Ring Operation

MACRO transmits data at a raw rate of 125 Mbps, with 2 of every 10 bits redundant for low-level error checking, resulting in a true data rate of 100 Mbps. Packets contain 72 bits of transmissible data, with 24 bits of supporting data (a header byte, a ring-address byte, and a checksum byte for a second level of error checking), for a total of 96 bits, permitting packet transmission in a microsecond. One packet can contain all of the data needed for closed-loop control of an axis.

Up to 256 packets can be transmitted in a single ring cycle. Even with a conservative assumption of ½-microsecond between packets, a fully loaded ring can be updated at a 2.5 kHz rate. In more typical implementations, a 16-packet ring can be updated at up to 40 kHz; a 32-packet ring at up to 20 kHz. The timing of the ring transmissions is driven by the same clock signal on the ring’s master controller that drives its own repetitive calculations, keeping the hardware and software fully synchronized. The ring is fully self-synchronized; other devices on the ring are tied to the ring master’s clock over the ring.

Multiple master controllers are permitted on a single ring; they communicate through each other to their own slave devices. All are tied to the clock of the ring master. Packets of command data

simply pass through other masters and slave devices for which they are not addressed. When they reach the device for which they have been addressed, the command data is latched into the slave device, and feedback data from the slave substituted into the data packet on the fly for return to the master over the remainder of the ring. When the master receives this packet, it automatically latches in this data, ending the ring cycle.

Ring Protocols

MACRO can operate in any of several control modes. In all of these modes, both command and feedback data are updated and transferred at ring cycle rates:

- Position mode: Position commands, position feedback (for reporting only)
- Velocity mode: Velocity commands, position feedback
- Torque mode: Torque commands, position feedback
- Sine-wave mode: Phase current commands, position feedback
- Direct PWM mode: Phase voltage commands, position and phase-current feedback

In all of these modes, the controller's servo and commutation routines can operate just as they would in a self-contained controller, reading feedback from the ring just as they would read local feedback (e.g. directly from an encoder counter), and writing command outputs to the ring just as they would to a local device (e.g. a D/A converter). From this point, the MACRO ring hardware takes over automatically. Note that the last two modes are unique to MACRO.

All of the protocols are exceedingly simple, meaning that the software and hardware at the slave nodes can be implemented quickly, reliably, and cost-effectively. Other strategies require every remote slave node to contain powerful and complex capabilities.

In addition, non-servo I/O can be directly mapped and transferred over the ring at the ring update rate, making it as easy for the controller to use I/O over the ring as it is to use I/O on the controller.

Conclusion

MACRO's features make it uniquely capable of combining the best of centralized and distributed control strategies. System wiring cost and complexity can be greatly reduced as in other distributed schemes, but the control strategy can be kept virtually totally centralized, as in a "unitary" controller, maintaining low cost and complexity of the software.