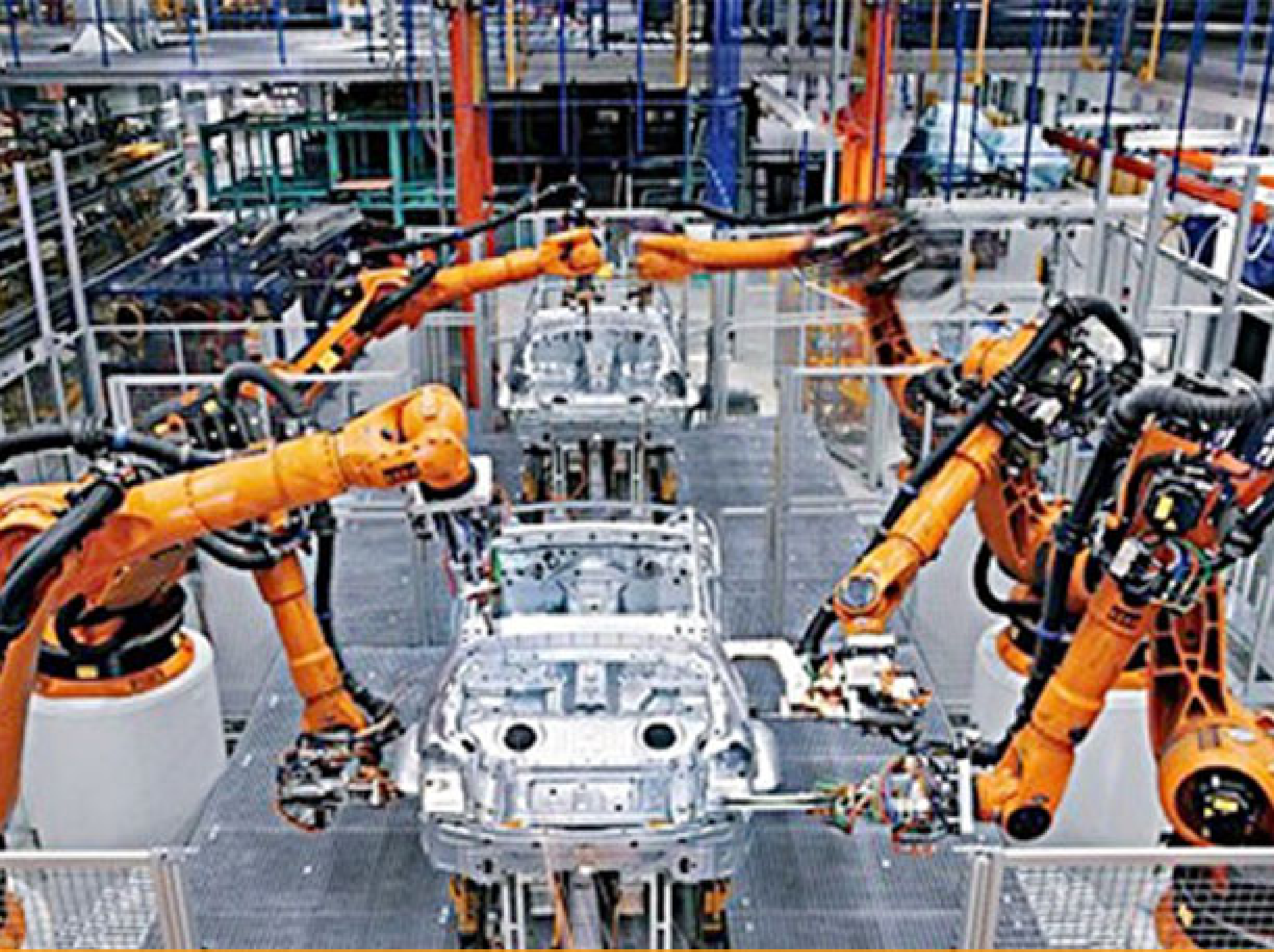


Industrial Automation

Nick Fulton

First Edition, 2012

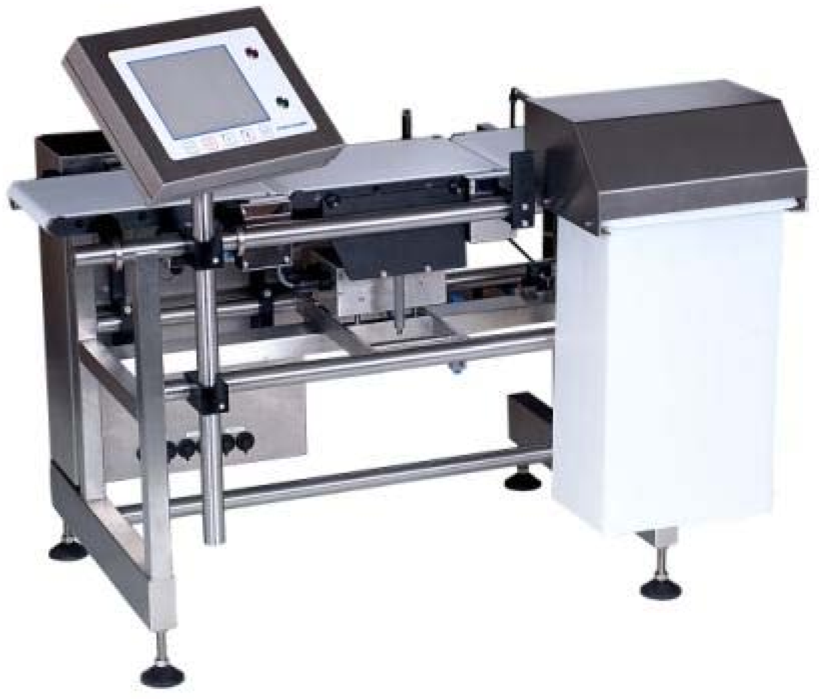
ISBN 978-81-323-3471-2

Published by:
Research World
4735/22 Prakashdeep Bldg,
Ansari Road, Darya Ganj,
Delhi - 110002
Email: info@wtbooks.com

Table of Contents

Chapter-1

Check Weigher



Example checkweigher. Product passes on the conveyor belt where it is weighed

A **checkweigher** is an automatic machine for checking the weight of packaged commodities. It is normally found at the offgoing end of a production process and is used to ensure that the weight of a pack of the commodity is within specified limits. Any packs that are outside the tolerance are taken out of line automatically.

A checkweigher can weigh in excess of 500 items per minute (depending on carton size and accuracy requirements).

Checkweighers often incorporate additional checking devices such as metal detectors and X-ray machines to enable other attributes of the pack to be checked and acted upon accordingly.

A typical machine

A checkweigher incorporates a series of conveyor belts. Checkweighers are known also as belt weighers, in-motion scales, conveyor scales, dynamic scales, and in-line scales. In filler applications, they are known as check scales. Typically, there are three belts or chain beds:

- An infeed belt that may change the speed of the package and to bring it up or down to a speed required for weighing. The infeed is also sometimes used as an indexer, which sets the gap between products to an optimal distance for weighing. It sometimes has special belts or chains to position the product for weighing.

- A weigh belt. This is typically mounted on a weight transducer which can typically be a strain-gauge load cell or a servo-balance (also known as a force-balance), or sometimes known as a split-beam. Some older machines may pause the weigh bed belt before taking the weight measurement. This may limit line speed and throughput.

- A reject belt that provides a method of removing an out-of-tolerance package from the conveyor line. The reject can vary by application. Some require an air-amplifier to blow small products off the belt, but heavier applications require a linear or radial actuator. Some fragile products are rejected by "dropping" the bed so that the product can slide gently into a bin or other conveyor.

For high-speed precision scales, a load cell using electromagnetic force restoration(EMFR) is appropriate. This kind of system charges an inductive coil, effectively floating the weigh bed in an electromagnetic field. When the weight is added, the movement of a ferrous material through that coil causes a loss of ElectroMagnetic Force. A precision circuit charges the coil back to its original charge. The amount added to the coil is precisely measured. The voltage produced is filtered and sampled into digital data. That voltage is then passed through a Digital Signal Processor (DSP) filter and ring-buffer to further reduce ambient and digital noise and delivered to a computerized controller.

It is usual for a built-in computer to take many weight readings from the transducer over the time that the package is on the weigh bed to ensure an accurate weight reading.

Calibration is critical. A lab scale, which usually is in an isolated chamber pressurized with dry nitrogen(pressurized at sea level) can weigh an object within plus or minus 100th of a gram, but ambient air pressure is a factor. This is straightforward when there is no motion, but in motion there is a factor that is not obvious-noise from the motion of a weigh belt, vibration, air-conditioning or refrigeration which can cause drafts. Torque on a load cell causes erratic readings.

A dynamic, in-motion checkweigher takes samples, and analyzes them to form an accurate weight over a given time period. In most cases, there is a trigger from an optical(or ultrasonic) device to signal the passing of a package. Once the trigger fires, there is a delay set to allow the package to move to the "sweet spot" (center) of the weigh bed to sample the weight. The weight is sampled for a given duration. If either of these times are wrong, the weight will be wrong. There seems to be no scientific method to predict these timings. Some systems have a "graphing" feature to do this, but it is generally more of an empirical method that works best.

- A reject conveyor to enable the out-of-tolerance packages to be removed from the normal flow while still moving at the conveyor velocity. The reject mechanism can be one of several types. Among these are a simple pneumatic pusher to push the reject pack sideways from the belt, a diverting arm to sweep the pack sideways and a reject belt that lowers or lifts to divert the pack vertically. A typical checkweigher usually has a bin to collect the out-of-tolerance packs.

Tolerance methods

There are several tolerance methods:

- The traditional "minimum weight" system where weights below a specified weight are rejected. Normally the minimum weight is the weight that is printed on the pack or a weight level that exceeds that to allow for weight losses after production such as evaporation of commodities that have a moisture content. The larger wholesale companies have mandated that any product shipped to them have accurate weight checks such that a customer can be confident that they are getting the amount of product for which they paid. These wholesalers charge large fees for inaccurately filled packages.

- The European Average Weight System which follows three specified rules known as the "Packers Rules".
- Other published standards and regulations such as NIST Handbook 133

Data Collection

There is also a requirement under the European Average Weight System that data collected by checkweighers is archived and is available for inspection. Most modern checkweighers are therefore equipped with communications ports to enable the actual pack weights and derived data to be uploaded to a host computer. This data can also be used for management information enabling processes to be fine-tuned and production performance monitored.

Checkweighers that are equipped with high speed communications such as Ethernet ports are capable of integrating themselves in to groups such that a group of production lines that are producing identical products can be considered as one production line for the purposes of weight control. For example, a line that is running with a low average weight can be complemented by another that is running with a high average weight such that the aggregate of the two lines will still comply with rules.

An alternative is to program the checkweigher to check bands of different weight tolerances. For instance, the total valid weight is 100 grams ±15 grams. This means that the product can weigh 85 g - 115 g. However, it is obvious that if you are producing 10,000 packs a day, and most of your packs are 110 g, you are losing 100 kg of product. If you try to run closer to 85 g, you may have a high rejection rate.

EXAMPLE: A checkweigher is programmed to indicate 5 zones with resolution to 1 g:

1. Under Reject.... the product weighs 84.9 g or less
2. Under OK........ the product weighs 85 g, but less than 95 g
3. Valid........... the product weighs 96 g, but less than 105 g
4. Over OK......... the product weighs 105 g, and less than 114 g
5. Over Reject..... the product weighs over the 115 g limit

With a check weigher programmed as a zone checkweigher, the data collection over the networks, as well as local statistics, can indicate the need to check the settings on the upstream equipment to better control flow into the packaging. In some cases the dynamic scale sends a signal to a filler, for instance, in real-time, controlling the actual flow into a barrel, can, bag, etc. In many cases a checkweigher has a light-tree with different lights to indicate the variation of the zone weight of each product.

Application considerations

Speed and accuracy that can be achieved by a checkweigher is influenced by the following:

- Pack length
- Pack weight
- Line speed required
- Pack content (solid or liquid)
- Motor technology
- Stabilization time of the weight transducer
- Airflow causing readings in error
- Vibrations from machinery causing unnecessary rejects
- Sensitivity to temperature, as the load cells *can* be temperature sensitive
- Quality control tape
- Band with the principles of quality control weight
- Weighing tape quality control principles
- Automatic weighing tape

- Weight control tape
- Food weight control tape
- Drug for weight control tape
- Detergent industry for the quality control tape
- Weight control tape for chemical industry
- Weight control tape for the pasta industry
- Flour weight control tape for the industry

Applications

In-motion scales are dynamic machines that can be designed to perform thousands of tasks. Some are used as simple caseweighers at the end of the conveyor line to ensure the overall finished package product is within its target weight.

An in motion conveyor checkweigher can be used to detect missing pieces of a kit, such as a cell phone package that is missing the manual, or other collateral. Checkweighers are typically used on the incoming conveyor chain, and the output pre-packaging conveyor chain in a poultry processing plant. The bird is weighed when it comes onto the conveyor, then after processing and washing at the end, the network computer can determine whether or not the bird absorbed too much water, which as it is further processed, will be drained, making the bird under its target weight.

A high speed conveyor scale can be used to change the pacing, or pitch of the products on the line by speeding, or slowing the product speed to change the distance between packs before reaching a different speed going into a conveyor machine that is boxing multiple packs into a box.

A checkweigher can be used to count packs, and the aggregate (total) weight of the boxes going onto a pallet for shipment, including the ability to read each package's weight and cubic dimensions. The controller computer can print a shipping label and a bar-code label to identify the weight, the cubic dimensions, ship-to address, and other data for machine ID through the shipment of the product. A receiving checkweigher for the shipment can read the label with a bar code scanner, and determine if the shipment is as it was before the transportation carrier received it from the shipper's loading dock, and determine if a box is missing, or something was pilfered or broken in transit.

Checkweighers are also used for Quality management. For instance, raw material for machining a bearing is weighed prior to beginning the process, and after the process, the quality inspector expects that a certain amount of metal was removed in the finishing process. The finished bearings are checkweighed, and bearings over- or underweight are rejected for physical inspection. This is a benefit to the inspector, since he can have a high confidence that the ones not rejected are within machining tolerance.A common usage is for throttling plastic extruders such that a bottle used to package detergent meets that requirements of the finished packager.

Quality management can use a checkweigher for Nondestructive testing to verify finished goods using common Evaluation methods to detect pieces missing from a "finished" product, such as grease from a bearing, or a missing roller within the housing.

Checkweighers can be built with metal detectors, x-ray machines, open-flap detection, bar-code scanners, holographic scanners, temperature sensors, vision inspectors, timing screws to set the timing and spacing between product, indexing gates and concentrator ducts to line up the product into a designated area on the conveyor. An industrial motion checkweigher can sort products from a fraction of a gram to many, many kilograms. In English units, is this from less than 100th of an ounce to as much as 500 lbs or more. Specialized checkweighers can weigh commercial aircraft, and even find their center-of-gravity.

Checkweighers can be very high speed, processing products weighing fractions of a gram at over 100m/m (meters per minute, such as pharmaceuticals, and 200 lb bags of produce at over 100fpm(feet per minute). They can be designed in many shapes and sizes, hung from ceilings, raised on mezzanines, operated in ovens or in refrigerators. Their conveying medium can be industrial belting, low-static belting, chains similar to bicycle chains(but much smaller), or interlocked chain belts of any width. They can have chain belts made of special materials, different polymers, metals, etc.

Checkweighers are used in cleanrooms, dry atmosphere environments, wet environments, produce barns, food processing, drug processing, etc. Checkweighers are specified by the kind of environment, and the kind of cleaning will be used. Typically, a checkweigher for produce is made of mild steel, and one that will be cleaned with harsh chemicals, such as bleach, will be made with all stainless steel parts, even the Load cells. These machines are labeled "full washdown", and must have every part and component specified to survive the washdown environment.

Checkweighers are operated in some applications for extremely long periods of time-24/7 year round. Generally, conveyor lines are not stopped unless there is maintenance required, or there is an emergency stop, called an E-stop. Checkweighers operating in high density conveyor lines may have numerous special equipments in their design to ensure that if an E-stop occurs, all power going to all motors is removed until the E-stop is cleared and reset.

Chapter-2

CANopen

CANopen is a communication protocol and device profile specification for embedded systems used in automation. In terms of the OSI model, CANopen implements the layers above and including the network layer. The CANopen standard consists of an addressing scheme, several small communication protocols and an application layer defined by a device profile. The communication protocols have support for network management, device monitoring and communication between nodes, including a simple transport layer for message segmentation/desegmentation. The lower level protocol implementing the data link and physical layers is usually Controller Area Network (CAN), although devices using some other means of communication (such as Ethernet Powerlink, EtherCAT) can also implement the CANopen device profile.

The basic CANopen device and communication profiles are given in the CiA 301 specification released by CAN in Automation. Profiles for more specialized devices are built on top of this basic profile, and are specified in numerous other standards released by CAN in Automation, such as CiA 401 for I/O-modules and CiA 402 for motion control.

Device model

Every CANopen device has to implement certain standard features in its controlling software.

- A **Communication unit** implements the protocols for messaging with the other nodes in the network
- Starting and resetting the device is controlled via a state machine. It must contain the states Initialization, Pre-operational, Operational and Stopped. The transitions between states are made by issuing a network management (NMT) communication object to the device.
- The **object dictionary** is an array of variables with a 16-bit index. Additionally, each variable can have an 8-bit subindex. The variables can be used to configure the device and reflect its environment, i.e. contain measurement data.

- The **application** part of the device actually performs the desired function of the device, after the state machine is set to the operational state. The application is configured by variables in the object dictionary and the data is sent and received through the communication layer.

Object dictionary

CANopen devices must have an object dictionary, which is used for configuration and non-realtime communication with the device. An entry in the object dictionary is defined by:

- **Index**, the 16-bit address of the object in the dictionary
- **Object name**, a symbolic type of the object in the entry, such as an array, record, or simple variable
- **Name**, a string describing the entry
- **Type**, gives the datatype of the variable
- **Attribute**, which gives information on the access rights for this entry, this can be read/write, read-only, write-only or read only constant
- The **Mandatory/Optional** field defines whether a device conforming to the device specification has to implement this object or not

The basic datatypes for object dictionary values such as Booleans, integers and floats are defined in the standard, as well as composite datatypes such as arrays, records and strings. The composite datatypes can be subindexed with an 8-bit index. The value in subindex 0 of an array or record indicates the number of elements in the data structure, and is of type UNSIGNED8.

For example, the device communication parameters, standardized in the basic device profile CiA 301 are mapped in the index range 0x1000 - 0x1FFF ("communication profile area"). The first few entries in this area are as follows:

Index	Object name	Name	Type	Attribute	M/O
0x1000	VAR	device type	UNSIGNED32	ro	M
0x1001	VAR	error register	UNSIGNED8	ro	M
...					
0x1008	VAR	manufacturer device name	Vis-String	const	O
...					

Given suitable tools, the object dictionary of a device can be configured by editing an electronic data sheet (EDS) file and uploading the variable values to the device. The format of the EDS-file according to CiA306 is INI file, there is an upcoming XML-style format, that is described in CiA311.

Communication

Communication objects

CANbus, the physical layer of CANopen, can only transmit short packages consisting of an 11-bit id, remote transmission request (RTR) bit and 0 to 8 bytes of data. The CANopen standard divides the 11-bit CAN frame id into a 4-bit function code and 7-bit CANopen node id. This limits the number of devices in a CANopen network to 127. An extension to the CANbus standard (CAN 2.0 B) allows extended frame ids of 29 bits, but in practice CANopen networks big enough to need the extended id range are rarely seen.

In CANopen the 11-bit id of a CAN-frame is known as communication object identifier, or COB-ID. In case of a transmission collision, the bus arbitration used in the CANbus allows the frame with the smallest id to be transmitted first and without a delay. Since in CANopen frames the first 4 bits of the frame id are reserved to the function code, giving a low code number for time critical functions ensures the lowest possible delay.

Contents of a standard CANopen frame:

	Function code	**Node ID**	**RTR**	**Data length**	**Data**
Length	4 bits	7 bits	1 bit	4 bits	0-8 bytes

The standard reserves certain COB-IDs to network management and SDO transfers. Some function codes and COB-IDs have to be mapped to standard functionality after device initialization, but can be configured for other uses later.

Communication models

Different kinds of communication models are used in the messaging between CANopen nodes.

In a **master/slave** relationship, one CANopen node is designated as the master, which sends or requests data from the slaves. The NMT protocol is an example of a master/slave communication model.

A **client/server** relationship is implemented in the SDO protocol, where the SDO client sends data (the object dictionary index and subindex) to an SDO server, which replies with one or more SDO packages containing the requested data (the contents of the object dictionary at the given index).

A **producer/consumer** model is used in the Heartbeat and Node Guarding protocols. In the *push-model* of producer/consumer, the producer sends data to the consumer without a specific request, whereas in the *pull model*, the consumer has to request the data from the producer.

Protocols

Network management (NMT) protocols

The NMT protocols are used to issue state machine change commands (ie. to start and stop the devices), detect remote device bootups and error conditions.

The **Module control protocol** is used by the NMT master to change the state of the devices. The CAN-frame COB-ID of this protocol is always 0, meaning that it has a function code 0 and node id 0, which means that every node in the network will process this message. The actual node id, to which the command is meant to, is given in the data part of the message. This can also be 0, meaning that all the devices in the bus should go to the indicated state.

The **Heartbeat protocol** is used to monitor the nodes in the network and verify that they are alive. A heartbeat producer (usually a slave device) periodically sends a message with binary function code of 1110 and its node id (COB ID = 0x700 + node id). The data part of the frame contains a byte indicating the node status. The heartbeat consumer reads these messages. If the messages fail to arrive within a certain time limit (defined in the object dictionary of the devices) the consumer can take action to, for example, reset the device or indicate an error. Frame format is :

COBID	Data Byte 0
0x700 + NodeID	State

CANopen devices are required to make the transition from the state Initializing to Pre-operational automatically during bootup. When this transition is made, a single heartbeat message is sent to the bus. This is the **bootup protocol**.

A response/reply-style (pull model) protocol for slave monitoring called Node guarding protocol exists.

Service Data Object (SDO) protocol

The SDO protocol is used to set and read values from the object dictionary of a remote device. The device whose object dictionary is accessed is the SDO server and the device accessing the remote device is the SDO client. The communication is always initiated by the SDO client. In CANopen terminology, communication is viewed from the SDO server, so that a read from an object dictionary results in an SDO upload and a write to dictionary is an SDO download.

As the object dictionary values can be larger than the 8 byte limit of a CAN frame, the SDO protocol implements segmentation and desegmentation of longer messages. Actually, there are two of these protocols: SDO download/upload and SDO Block download/upload. The SDO block transfer is a newer addition to standard, which allows large amounts of data to be transferred with slightly less protocol overhead.

The COB IDs of the respective SDO transfer messages from client to server and server to client can be set in the object dictionary. Up to 127 SDO servers can be set up in the object dictionary addresses 0x1200 - 0x127F. Similarly, the SDO client connections of the device can be configured with variables at 0x1280 - 0x12FF. However the *pre-defined connection set* defines an SDO channel which can be used even just after bootup (in the Pre-operational state) to configure the device. The COB IDs of this channel are 0x600 + node id for receiving and 0x580 + node id for transmitting.

To initiate a download, the SDO client sends the following data in a CAN message with the 'receive' COB ID of the SDO channel:

3 bits	**1 bit**	**2 bits**	**1 bit**	**1 bit**	**2 bytes**	**1 byte**	**4 bytes**
ccs=1	reserved(=0)	n	e	s	index	subindex	data

- **ccs** is the client command specifier of the SDO transfer, this is 0 for SDO segment download, 1 for initiating download, 2 for initiating upload, 3 for SDO segment upload and 4 for aborting an SDO transfer
- **n** is the number of bytes in the data part of the message which do not contain data, only valid if e and s are set
- **e**, if set, indicates an expedited transfer , i.e. all data exchanged are contained within the message. If this bit is cleared then the message is a segmented transfer where the data does not fit into one message and multiple messages are used.
- **s**, if set, indicates that the data size is specified in n (if e is set) or in the data part of the message
- **index** is the object dictionary index of the data to be accessed
- **subindex** is the subindex of the object dictionary variable
- **data** contains the data to be uploaded in the case of an expedited transfer (e is set), or the size of the data to be uploaded (s is set, e is not set)

Process Data Object (PDO) protocol

Process Data Object protocol is used to process real time data among various nodes. You can transfer up to 8 bytes (64bits) data per one PDO either from or to the device. One PDO can contain multiple object dictionary entries and the objects within one PDO is configurable using the mapping and parameter object dictionary entries.

There are two kinds of PDOs: transmit and receive PDOs (TPDO and RPDO). The former is for data coming from the device and the latter is for data going to the device, ie. with RPDO you can send data to the device and with TPDO you can read data from the device. In the pre-defined connection set there are identifiers for four (4) TPDOs and four (4) RPDOs available. With configuration 512 PDOs are possible.

PDOs can be sent synchronously or asynchronously. Synchronous PDOs are sent after the SYNC message whereas asynchronous messages are sent after internal or external trigger. For example, you can make a request to a device to transmit TPDO that contains data you need by sending empty TPDO with RTR flag (if the device is configured to accept TPDO requests).

With RPDOs you can, for example, start two devices simultaneously. You only need to map the same RPDO into two or more different device and make sure those RPDOs are mapped with the same COB ID.

Synchronization Object (SYNC) protocol

The Sync-Producer provides the synchronization-signal for the Sync-Consumer. When the Sync-Consumer receive the signal they start carrying out their synchronous tasks.

In general the fixing of the transmission time of synchronous PDO messages coupled with the periodicity of transmission of the Sync Object guarantees that sensor devices may arrange to sample process variables and that actuator devices may apply their actuation in a coordinated fashion.

The identifier of the Sync Object is available at index 1005h.

Time Stamp Object (TIME) protocol

Usually the Time-Stamp object represents an absolute time in ms after midnight and the number of days since January 1, 1984. This is a bit sequence of length 48 (6 byte).

Some time critical applications especially in large networks with reduced transmission rates require very accurate synchronization; it may be necessary to synchronize the local clocks with an accuracy in the order of microseconds. This is achieved by using the optional high resolution synchronization protocol which employs a special form of time stamp message to adjust the inevitable drift of the local clocks.

The high-resolution time-stamp is encoded as unsigned32 with a resolution of 1 microsecond which means that the time counter restarts every 72 minutes. It is configured by mapping the high resolution time-stamp (object 1013h) into a PDO.

Emergency Object (EMCY) protocol

Emergency messages are triggered by the occurrence of a device internal fatal error situation and are transmitted from the concerned application device to the other devices with high priority. This makes them suitable for interrupt type error alerts. An Emergency Telegram may be sent only once per ‘error event’, i.e. the emergency messages must not be repeated. As long as no new errors occur on a device no further emergency message must be sent. By means of CANopen Communication Profile defined emergency error codes, the error register and device specific additional information are specified in the device profiles.

Initialization

Sample trace of communications between a master and 2 pressure transducer slaves configured for id 1 and node id 2.

CAN ID	DATA LENGTH	DATA	Description
0x0	2	1 0	Master puts bus into operational mode
0x80	0		Master sends a SYNC message, which triggers devices to send data
0x181	4	CD 82 01 00	Node at ID 1 (CID-0x180), reading pressure of 0x0182CD(99021) pascals
0x182	4	E5 83 01 00	Node at ID 2 (CID-0x181), reading pressure of 0x0183E5(99301) pascals

Electronic Device Description

The EDS file is a file format, defined in CiA306, that describes the communication behaviour and the object dictionary entries of a device. This allows tools such as service tools, configuration tools, development tools, and others to handle the devices properly.

Those EDS files are mandatory for passing the CiA CANopen Conformance Test. A free EDS checker is CANchkEDS.

Since end of 2007 a new XML based format called XDD is defined in CiA311. XDD is conformant to ISO Standard 15745. For both formats a free editor is available, called CANeds.

Glossary of CANopen Terms

PDO Process Data Object - Inputs and outputs. Values of type RPM, V, Hz, mAmps etc.
SDO Service Data Object - Configuration settings, possibly NODE ID, baud rate, offset, gain etc.
COB-ID - CAN Object Identifiers
CAN ID - CAN Identifier. This is the 11 bit CAN message identifier which is at the beginning of every CAN message on the bus.
EDS - Electronic data sheet. This is an INI style resp. XML style formatted file.
DCF - Device configuration file. This is modified EDS with settings for node ID and baud rate.

Chapter-3

SCADA

SCADA stands for *supervisory control and data acquisition*. It generally refers to industrial control systems: computer systems that monitor and control industrial, infrastructure, or facility-based processes, as described below:

- Industrial processes include those of manufacturing, production, power generation, fabrication, and refining, and may run in continuous, batch, repetitive, or discrete modes.
- Infrastructure processes may be public or private, and include water treatment and distribution, wastewater collection and treatment, oil and gas pipelines, electrical power transmission and distribution, Wind farms, civil defense siren systems, and large communication systems.
- Facility processes occur both in public facilities and private ones, including buildings, airports, ships, and space stations. They monitor and control HVAC, access, and energy consumption.

Common system components

A SCADA System usually consists of the following subsystems:

- A Human-Machine Interface or HMI is the apparatus which presents process data to a human operator, and through this, the human operator monitors and controls the process.
- A supervisory (computer) system, gathering (acquiring) data on the process and sending commands (control) to the process.
- Remote Terminal Units (RTUs) connecting to sensors in the process, converting sensor signals to digital data and sending digital data to the supervisory system.
- Programmable Logic Controller (PLCs) used as field devices because they are more economical, versatile, flexible, and configurable than special-purpose RTUs.
- Communication infrastructure connecting the supervisory system to the Remote Terminal Units.

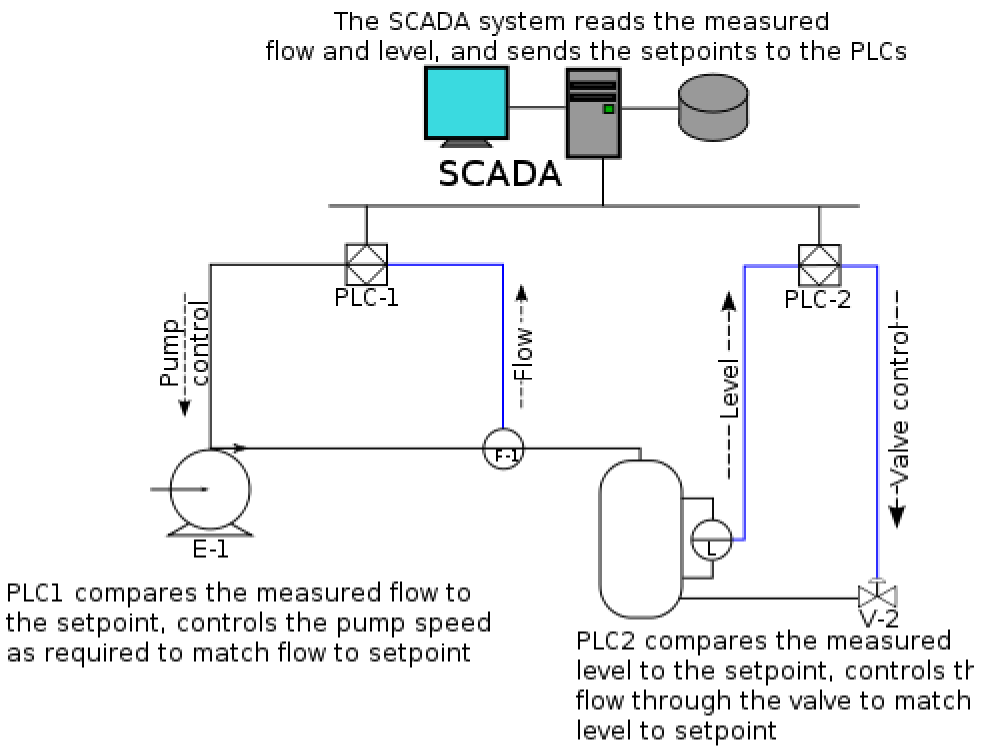
Supervision vs. control

There is, in several industries, considerable confusion over the differences between SCADA systems and distributed control systems (DCS). Generally speaking, a SCADA system always refers to a system that *coordinates*, but does not *control* processes in real time. The discussion on real-time control is muddied somewhat by newer telecommunications technology, enabling reliable, low latency, high speed communications over wide areas. Most differences between SCADA and DCS are culturally determined and can usually be ignored. As communication infrastructures with higher capacity become available, the difference between SCADA and DCS will fade.

Summary: 1. DCS is process oriented, while SCADA is data acquisition oriented. 2. DCS is process state driven, while SCADA is event driven. 3. DCS is commonly used to handle operations on a single locale, while SCADA is preferred for applications that are spread over a wide geographic location. 4. DCS operator stations are always connected to its I/O, while SCADA is expected to operate despite failure of field communications.

Systems concepts

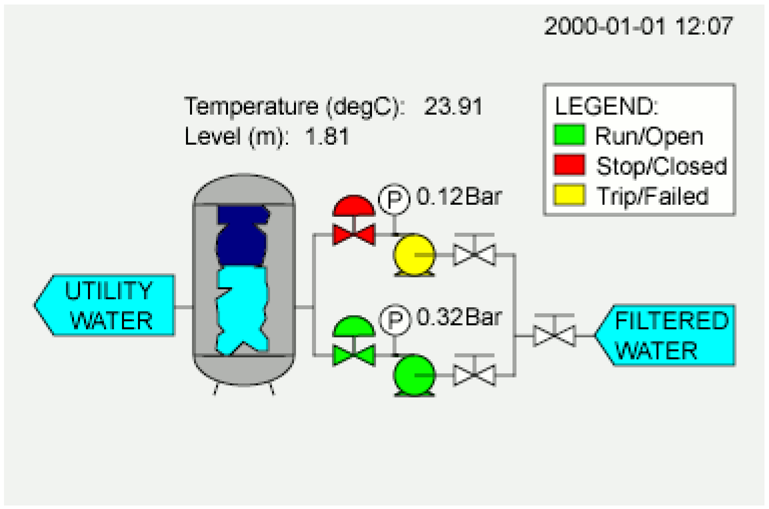
The term SCADA usually refers to centralized systems which monitor and control entire sites, or complexes of systems spread out over large areas (anything between an industrial plant and a country). Most control actions are performed automatically by Remote Terminal Units ("RTUs") or by programmable logic controllers ("PLCs"). Host control functions are usually restricted to basic overriding or *supervisory* level intervention. For example, a PLC may control the flow of cooling water through part of an industrial process, but the SCADA system may allow operators to change the set points for the flow, and enable alarm conditions, such as loss of flow and high temperature, to be displayed and recorded. The feedback control loop passes through the RTU or PLC, while the SCADA system monitors the overall performance of the loop.


The SCADA system reads the measured flow and level, and sends the setpoints to the PLCs
SCADA
PLC-1
PLC-2
Pump control
Flow
Level
Valve control
F-1
L
E-1
V-2
PLC1 compares the measured flow to the setpoint, controls the pump speed as required to match flow to setpoint
PLC2 compares the measured level to the setpoint, controls th flow through the valve to match level to setpoint

Data acquisition begins at the RTU or PLC level and includes meter readings and equipment status reports that are communicated to SCADA as required. Data is then compiled and formatted in such a way that a control room operator using the HMI can make supervisory decisions to adjust or override normal RTU (PLC) controls. Data may also be fed to a Historian, often built on a commodity Database Management System, to allow trending and other analytical auditing.

SCADA systems typically implement a distributed database, commonly referred to as a *tag database*, which contains data elements called *tags* or *points*. A point represents a single input or output value monitored or controlled by the system. Points can be either "hard" or "soft". A hard point represents an actual input or output within the system, while a soft point results from logic and math operations applied to other points. (Most implementations conceptually remove the distinction by making every property a "soft" point expression, which may, in the simplest case, equal a single hard point.) Points are normally stored as value-timestamp pairs: a value, and the timestamp when it was recorded or calculated. A series of value-timestamp pairs gives the history of that point. It's also common to store additional metadata with tags, such as the path to a field device or PLC register, design time comments, and alarm information.

Human Machine Interface



Typical Basic SCADA

A Human-Machine Interface or HMI is the apparatus which presents process data to a human operator, and through which the human operator controls the process.

An HMI is usually linked to the SCADA system's databases and software programs, to provide trending, diagnostic data, and management information such as scheduled maintenance procedures, logistic information, detailed schematics for a particular sensor or machine, and expert-system troubleshooting guides.

The HMI system usually presents the information to the operating personnel graphically, in the form of a mimic diagram. This means that the operator can see a schematic representation of the plant being controlled. For example, a picture of a pump connected to a pipe can show the operator that the pump is running and how much fluid it is pumping through the pipe at the moment. The operator can then switch the pump off. The HMI software will show the flow rate of the fluid in the pipe decrease in real time. Mimic diagrams may consist of line graphics and schematic symbols to represent process elements, or may consist of digital photographs of the process equipment overlain with animated symbols.

The HMI package for the SCADA system typically includes a drawing program that the operators or system maintenance personnel use to change the way these points are represented in the interface. These representations can be as simple as an on-screen traffic light, which represents the state of an actual traffic light in the field, or as complex as a multi-projector display representing the position of all of the elevators in a skyscraper or all of the trains on a railway.

An important part of most SCADA implementations is alarm handling. The system monitors whether certain alarm conditions are satisfied, to determine when an alarm event has occurred. Once an alarm event has been detected, one or more actions are taken (such as the activation of one or more alarm indicators, and perhaps the generation of email or text messages so that management or remote SCADA operators are informed). In many cases, a SCADA operator may have to acknowledge the alarm event; this may deactivate some alarm indicators, whereas other indicators remain active until the alarm conditions are cleared. Alarm conditions can be explicit - for example, an alarm point is a digital status point that has either the value NORMAL or ALARM that is calculated by a formula based on the values in other analogue and digital points - or implicit: the SCADA system might automatically monitor whether the value in an analogue point lies outside high and low limit values associated with that point. Examples of alarm indicators include a siren, a pop-up box on a screen, or a coloured or flashing area on a screen (that might act in a similar way to the "fuel tank empty" light in a car); in each case, the role of the alarm indicator is to draw the operator's attention to the part of the system 'in alarm' so that appropriate action can be taken. In designing SCADA systems, care is needed in coping with a cascade of alarm events occurring in a short time, otherwise the underlying cause (which might not be the earliest event detected) may get lost in the noise. Unfortunately, when used as a noun, the word 'alarm' is used rather loosely in the industry; thus, depending on context it might mean an alarm point, an alarm indicator, or an alarm event.

Hardware solutions

SCADA solutions often have Distributed Control System (DCS) components. Use of "smart" RTUs or PLCs, which are capable of autonomously executing simple logic processes without involving the master computer, is increasing. A standardized control programming language, IEC 61131-3 (a suite of 5 programming languages including Function Block, Ladder, Structured Text, Sequence Function Charts and Instruction List), is frequently used to create programs which run on these RTUs and PLCs. Unlike a procedural language such as the C programming language or FORTRAN, IEC 61131-3 has minimal training requirements by virtue of resembling historic physical control arrays. This allows SCADA system engineers to perform both the design and implementation of a program to be executed on an RTU or PLC. A Programmable automation controller (PAC) is a compact controller that combines the features and capabilities of a PC-based control system with that of a typical PLC. PACs are deployed in SCADA systems to provide RTU and PLC functions. In many electrical substation SCADA applications, "distributed RTUs" use information processors or station computers to communicate with digital protective relays, PACs, and other devices for I/O, and communicate with the SCADA master in lieu of a traditional RTU.

Since about 1998, virtually all major PLC manufacturers have offered integrated HMI/SCADA systems, many of them using open and non-proprietary communications protocols. Numerous specialized third-party HMI/SCADA packages, offering built-in compatibility with most major PLCs, have also entered the market, allowing mechanical engineers, electrical engineers and technicians to configure HMIs themselves, without the need for a custom-made program written by a software developer.

Remote Terminal Unit (RTU)

The RTU connects to physical equipment. Typically, an RTU converts the electrical signals from the equipment to digital values such as the open/closed status from a switch or a valve, or measurements such as pressure, flow, voltage or current. By converting and sending these electrical signals out to equipment the RTU can control equipment, such as opening or closing a switch or a valve, or setting the speed of a pump.

Supervisory Station

The term "Supervisory Station" refers to the servers and software responsible for communicating with the field equipment (RTUs, PLCs, etc), and then to the HMI software running on workstations in the control room, or elsewhere. In smaller SCADA systems, the master station may be composed of a single PC. In larger SCADA systems, the master station may include multiple servers, distributed software applications, and disaster recovery sites. To increase the integrity of the system the multiple servers will often be configured in a dual-redundant or hot-standby formation providing continuous control and monitoring in the event of a server failure.

Operational philosophy

For some installations, the costs that would result from the control system failing are extremely high. Possibly even lives could be lost. Hardware for some SCADA systems is ruggedized to withstand temperature, vibration, and voltage extremes, but in most critical installations reliability is enhanced by having redundant hardware and communications channels, up to the point of having multiple fully equipped control centres. A failing part can be quickly identified and its functionality automatically taken over by backup hardware. A failed part can often be replaced without interrupting the process. The reliability of such systems can be calculated statistically and is stated as the mean time to failure, which is a variant of mean time between failures. The calculated mean time to failure of such high reliability systems can be on the order of centuries.

Communication infrastructure and methods

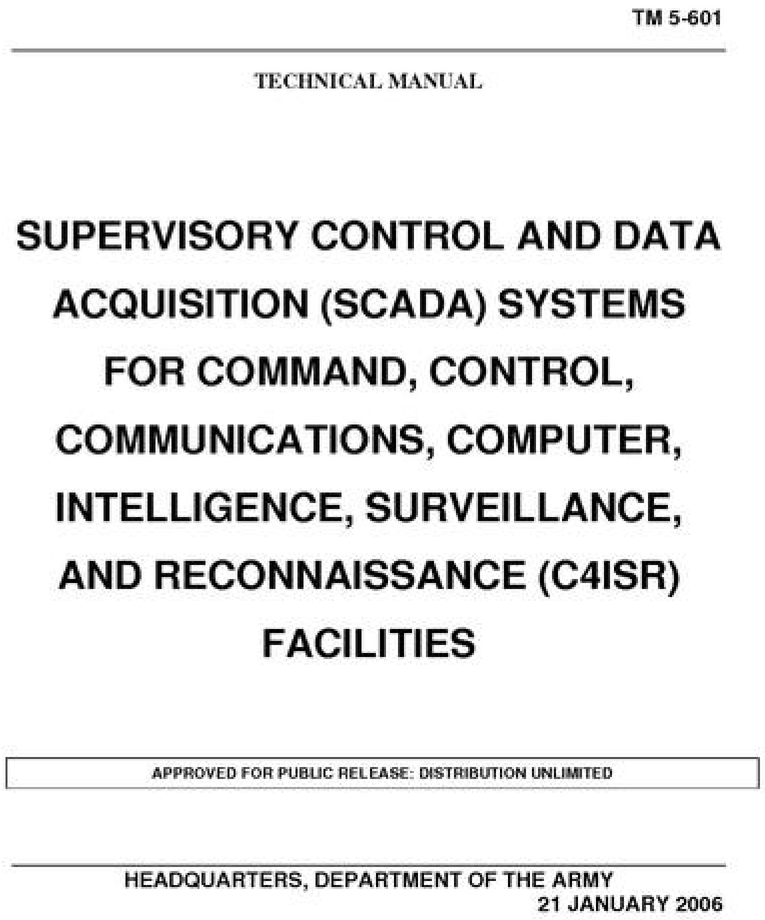
SCADA systems have traditionally used combinations of radio and direct serial or modem connections to meet communication requirements, although Ethernet and IP over SONET / SDH is also frequently used at large sites such as railways and power stations. The remote management or monitoring function of a SCADA system is often referred to as telemetry.

This has also come under threat with some customers wanting SCADA data to travel over their pre-established corporate networks or to share the network with other applications. The legacy of the early low-bandwidth protocols remains, though. SCADA protocols are designed to be very compact and many are designed to send information to the master station only when the master station polls the RTU. Typical legacy SCADA protocols include Modbus RTU, RP-570, Profibus and Conitel. These communication protocols are all SCADA-vendor specific but are widely adopted and used. Standard protocols are IEC 60870-5-101 or 104, IEC 61850 and DNP3. These communication protocols are standardized and recognized by all major SCADA vendors. Many of these protocols now contain extensions to operate over TCP/IP. It is good security engineering practice to avoid connecting SCADA systems to the Internet so the attack surface is reduced.

RTUs and other automatic controller devices were being developed before the advent of industry wide standards for interoperability. The result is that developers and their management created a multitude of control protocols. Among the larger vendors, there was also the incentive to create their own protocol to "lock in" their customer base. A list of automation protocols is being compiled here.

Recently, OLE for Process Control (OPC) has become a widely accepted solution for intercommunicating different hardware and software, allowing communication even between devices originally not intended to be part of an industrial network.

SCADA architectures



TM 5-601

TECHNICAL MANUAL

SUPERVISORY CONTROL AND DATA ACQUISITION (SCADA) SYSTEMS FOR COMMAND, CONTROL, COMMUNICATIONS, COMPUTER, INTELLIGENCE, SURVEILLANCE, AND RECONNAISSANCE (C4ISR) FACILITIES

APPROVED FOR PUBLIC RELEASE: DISTRIBUTION UNLIMITED

HEADQUARTERS, DEPARTMENT OF THE ARMY
21 JANUARY 2006

The United States Army's Training Manual 5-601 covers "SCADA Systems for C4ISR Facilities".

SCADA systems have evolved through 3 generations as follows:

First generation: "Monolithic"

In the first generation, computing was done by mainframe computers. Networks did not exist at the time SCADA was developed. Thus SCADA systems were independent systems with no connectivity to other systems. Wide Area Networks were later designed by RTU vendors to communicate with the RTU. The communication protocols used were often proprietary at that time. The first-generation SCADA system was redundant since a back-up mainframe system was connected at the bus level and was used in the event of failure of the primary mainframe system.

Second generation: "Distributed"

The processing was distributed across multiple stations which were connected through a LAN and they shared information in real time. Each station was responsible for a particular task thus making the size and cost of each station less than the one used in First Generation. The network protocols used were still mostly proprietary, which led to significant security problems for any SCADA system that received attention from a hacker. Since the protocols were proprietary, very few people beyond the developers and hackers knew enough to determine how secure a SCADA installation was. Since both parties had invested interests in keeping security issues quiet, the security of a SCADA installation was often badly overestimated, if it was considered at all.

Third generation: "Networked"

These are the current generation SCADA systems which use open system architecture rather than a vendor-controlled proprietary environment. The SCADA system utilizes open standards and protocols, thus distributing functionality across a WAN rather than a LAN. It is easier to connect third party peripheral devices like printers, disk drives, and tape drives due to the use of open architecture. WAN protocols such as Internet Protocol (IP) are used for communication between the master station and communications equipment. Due to the usage of standard protocols and the fact that many networked SCADA systems are accessible from the Internet, the systems are potentially vulnerable to remote cyber-attacks. On the other hand, the usage of standard protocols and security techniques means that standard security improvements are applicable to the SCADA systems, assuming they receive timely maintenance and updates.

Trends in SCADA

North American Electric Reliability Corporation has specified that electrical system data should be time-tagged to the nearest millisecond. Electrical system SCADA systems provide this Sequence of events recorder function, using Radio clocks to synchronize the RTU or distributed RTU clocks.

SCADA systems are coming in line with standard networking technologies. Ethernet and TCP/IP based protocols are replacing the older proprietary standards. Although certain characteristics of frame-based network communication technology (determinism, synchronization, protocol selection, environment suitability) have restricted the adoption of Ethernet in a few specialized applications, the vast majority of markets have accepted Ethernet networks for HMI/SCADA.

A few vendors have begun offering application specific SCADA systems hosted on remote platforms over the Internet. This removes the need to install and commission systems at the end-user's facility and takes advantage of security features already available in Internet technology, VPNs and SSL. Some concerns include security, Internet connection reliability, and latency.

SCADA systems are becoming increasingly ubiquitous. Thin clients, web portals, and web based products are gaining popularity with most major vendors. The increased convenience of end users viewing their processes remotely introduces security considerations. While these considerations are already considered solved in other sectors of Internet services, not all entities responsible for deploying SCADA systems have understood the changes in accessibility and threat scope implicit in connecting a system to the Internet.

Security issues

The move from proprietary technologies to more standardized and open solutions together with the increased number of connections between SCADA systems and office networks and the Internet has made them more vulnerable to attacks. Consequently, the security of SCADA-based systems has come into question as they are increasingly seen as extremely vulnerable to cyberwarfare/cyberterrorism attacks.

In particular, security researchers are concerned about:

- the lack of concern about security and authentication in the design, deployment and operation of existing SCADA networks
- the belief that SCADA systems have the benefit of security through obscurity through the use of specialized protocols and proprietary interfaces
- the belief that SCADA networks are secure because they are physically secured
- the belief that SCADA networks are secure because they are disconnected from the Internet

SCADA systems are used to control and monitor physical processes, examples of which are transmission of electricity, transportation of gas and oil in pipelines, water distribution, traffic lights, and other systems used as the basis of modern society. The security of these SCADA systems is important because compromise or destruction of these systems would impact multiple areas of society far removed from the original compromise. For example, a blackout caused by a compromised electrical SCADA system would cause financial losses to all the customers that received electricity from that source. How security will affect legacy SCADA and new deployments remains to be seen.

There are two distinct threats to a modern SCADA system. First is the threat of unauthorized access to the control software, whether it be human access or changes induced intentionally or accidentally by virus infections and other software threats residing on the control host machine. Second is the threat of packet access to the network segments hosting SCADA devices. In many cases, there is rudimentary or no security on the actual packet control protocol, so anyone who can send packets to the SCADA device can control it. In many cases SCADA users assume that a VPN is sufficient protection and are unaware that physical access to SCADA-related network jacks and switches provides the ability to totally bypass all security on the control software and fully control those SCADA networks. These kinds of physical access attacks bypass firewall and VPN security and are best addressed by endpoint-to-endpoint authentication and authorization such as are commonly provided in the non-SCADA world by in-device SSL or other cryptographic techniques.

The reliable function of SCADA systems in our modern infrastructure may be crucial to public health and safety. As such, attacks on these systems may directly or indirectly threaten public health and safety. Such an attack has already occurred, carried out on Maroochy Shire Council's sewage control system in Queensland, Australia. Shortly after a contractor installed a SCADA system there in January 2000 system components began to function erratically. Pumps did not run when needed and alarms were not reported. More critically, sewage flooded a nearby park and contaminated an open surface-water drainage ditch and flowed 500 meters to a tidal canal. The SCADA system was directing sewage valves to open when the design protocol should have kept them closed. Initially this was believed to be a system bug. Monitoring of the system logs revealed the malfunctions were the result of cyber attacks. Investigators reported 46 separate instances of malicious outside interference before the culprit was identified. The attacks were made by a disgruntled employee of the company that had installed the SCADA system. The employee was hoping to be hired full time to help solve the problem.

Many vendors of SCADA and control products have begun to address the risks posed by unauthorized access by developing lines of specialized industrial firewall and VPN solutions for TCP/IP-based SCADA networks as well as external SCADA monitoring and recording equipment. Additionally, application whitelisting solutions are being implemented because of their ability to prevent malware and unauthorized application changes without the performance impacts of traditional antivirus scans. Also, the ISA Security Compliance Institute (ISCI) is emerging to formalize SCADA security testing starting as soon as 2009. ISCI is conceptually similar to private testing and certification that has been performed by vendors since 2007. Eventually, standards being defined by ISA99 WG4 will supersede the initial industry consortia efforts, but probably not before 2011.

The increased interest in SCADA vulnerabilities has resulted in vulnerability researchers discovering vulnerabilities in commercial SCADA software and more general offensive SCADA techniques presented to the general security community. In electric and gas utility SCADA systems, the vulnerability of the large installed base of wired and wireless serial communications links is addressed in some cases by applying bump-in-the-wire devices that employ authentication and Advanced Encryption Standard encryption rather than replacing all existing nodes.

In June 2010, VirusBlokAda reported the first detection of malware that attacks SCADA systems (Siemens' WinCC/PCS7 systems) running on Windows operating systems. The malware is called Stuxnet and uses four zero-day attacks to install a rootkit which in turn logs in to the SCADA's database and steals design and control files. The malware is also capable of changing the control system and hiding those changes. The malware was found by an anti-virus security company on 14 systems with the majority in Iran.

Chapter-4

Automation

KUKA Industrial Robots being used at a bakery for food production

Automation is the use of control systems and information technologies to reduce the need for human work in the production of goods and services. In the scope of industrialization, automation is a step beyond mechanization. Whereas mechanization provided human operators with machinery to assist them with the muscular requirements of work, automation greatly decreases the need for human sensory and mental requirements as well. Automation plays an increasingly important role in the world economy and in daily experience.

Automation has had a notable impact in a wide range of industries beyond manufacturing (where it began). Once-ubiquitous telephone operators have been replaced largely by automated telephone switchboards and answering machines. Medical processes such as primary screening in electrocardiography or radiography and laboratory analysis of human genes, sera, cells, and tissues are carried out at much greater speed and accuracy by automated systems. Automated teller machines have reduced the need for bank visits to obtain cash and carry out transactions. In general, automation has been responsible for the shift in the world economy from industrial jobs to service jobs in the 20th and 21st centuries.

Advantages and disadvantages

The main **advantages** of automation are:

- **Replacing human operators in tasks that involve hard physical or monotonous work.**
- **Replacing humans in tasks done in dangerous environments (i.e. fire, space, volcanoes, nuclear facilities, underwater, etc.)**
- **Performing tasks that are beyond human capabilities of size, weight, speed, endurance, etc.**
- **Economy improvement. Automation may improve in economy of enterprises, society or most of humanity. For example, when an enterprise invests in automation, technology recovers its investment; or when a state or country increases its income due to automation like Germany or Japan in the 20th Century.**

The main **disadvantages** of automation are:

- **Technical Limitation.** Current technology is unable to automate all the desired tasks.
- **Security Threats/ Vulnerability** : An automated system may have limited level of intelligence, hence it is most likely susceptible to commit error.
- **Unpredictable development costs**. The research and development cost of automating a process may exceed the cost saved by the automation itself.
- **High initial cost.** The automation of a new product or plant requires a huge initial investment in comparison with the unit cost of the product, although the cost of automation is spread in many product batches.

Relationship to unemployment

Multivariate effect

Most people consider it common sense that automation has the potential to foster unemployment, because it obviates human work by transferring tasks to machines. However, the translation of that potential into observed effect has largely not happened in the two centuries during which it has been continually predicted. After many decades of automation development and dissemination, the net macroeconomic effect has been generally positive—automation has been part of a general trend of economic growth worldwide; standards of living have risen in many places; and automation has never yet been shown to have induced any widespread structural unemployment. The main explanation for this is that, so far, job losses in any one particular economic niche have always been more than offset by job gains in other niches. As the lowered unit cost of goods and services (which the automation made possible) gave consumers more purchasing power to devote to other goods and services, new jobs sprang up in the production of those goods and services. Thus each time that automation has freed up human resources, those resources have been redeployed by market forces (although it did not always happen without turbulence in the lives of individual workers).

One of the earliest promises of automation was to allow more free time, without any threat of income reduction. This effect has been seen in many individual facets of life (for example, the automatic washing machine has made laundry less time-consuming; engine control units have reduced the amount of automotive downtime; the automatic dishwasher has made dishwashing less time-consuming), but the net outcome of modern life in developed economies remains a state of hurry and busyness, mostly because rising living standards have brought rising expectations in direct relation. (Each time-saving improvement has made room for a new aspiration to take its place.)

Automation also does not imply unemployment when it makes possible tasks that were unimaginable without it (such as exploring Mars with the Sojourner rover). Likewise with fields where the economy is already fully adapted to an automated technology, and the jobs were lost long enough ago that the displacement was long since absorbed by the workforce (as with the continually advancing automation of the telephone switchboard, which eliminated most telephone operator jobs and kept many more from ever existing in the first place).

Today automation is quite advanced (relative to just a few lifetimes ago), and it continues to advance with an accelerating pace throughout the world. Although it has been encroaching on ever more skilled jobs, the general well-being and quality of life of most people in the world (where political factors have not muddied the picture) have improved. Clearly a multivariate effect has been at work (something much more than just the obvious idea that automation has the *potential* to cause unemployment). In fact, the idea that automation posed an *imminent* threat to employment, first articulated in 1811 by a group of textile workers known as Luddites, has proven to be so fallacious over the ensuing two centuries that economists call the imminent-threat idea the *Luddite fallacy*.

There is some concern today that the economy's ability to continue absorbing ever-increasing automation without experiencing significant structural unemployment may be heading toward an upper limit—that is, that we are approaching a point where the Luddite premise will no longer be entirely fallacious, because the relationship of humans to machines that made it fallacious is changing. In this view, the empirical strength of the eternal-fallaciousness idea is only a reflection of the parameter values of the environment thus far. In other words, the idea is undoubtedly an excellent explanation of the past, but whether it can accurately predict the future is an independent problem. Like an investment prospectus, proponents of this view caution that "past performance is no guarantee of future results."

Timeline of concerns about automation's relationship to unemployment

Early in the Industrial Revolution

Historical concerns about the effects of automation date back to the very beginning of the Industrial Revolution, when a social movement of English textile machine operators in the early 19th century known as the Luddites protested against Jacquard's automated weaving looms. The Luddites destroyed a number of these machines, which they felt threatened their jobs.

Later in the Industrial Revolution

The development of the American system of manufacturing disgusted many skilled machinists at a time when the very definition of being a machinist included a core element of skilled toolmaking and fitting on a craft basis. Innovations of this system included increasing reliance on jigs and gauges and on machine tools that built more of a process into the tool's movements (such as turret lathes and screw machines). These innovations continually turned skilled work into semi-skilled or unskilled, contributing to vast migrations of laborers across borders and oceans. However, despite this transformation, there were always other economic niches for skilled workers to go to, given enough searching. Recessions interfered with employment, but no foundational aspects of structural unemployment were caused by automation itself.

During the Machine Age

As in the preceding century, the period of 1880 to 1940 saw no underlying automation-induced structural lack of new economic opportunities for skilled workers to go to, given enough searching, although the Great Depression caused a tremendous disruption to employment. The foundational *potential* for full employment had not been lost, as would later be shown by the post–World War II economic expansion and other economic miracles.

During the 1950s through 1990s

The postwar development of new automation technologies using electronics, servomechanisms, and digital computers stoked a new wave of fears similar to the old Luddite ones. Among the working class and labor unions, there was stiff resistance to loss of employment through automation, including contract clauses won in hard-fought contract negotiations that mandated alternate employment for any workers whose positions were eliminated by automation. These clauses seemed a great victory for union workers at large corporations in developed nations, but because they had no effect at smaller, nonunionized companies or in developing nations, those corporations faced withering competition that shrank their market shares until their workers' gains eventually undermined their own success. However, the salvation for employment rates damaged in the industrial sector (secondary sector of the economy) came from the service sector (tertiary sector), which absorbed all of the workers that automation displaced elsewhere. For example, many manufacturing jobs left the United States during the 1990s but were offset by a one-time massive increase in IT jobs at the same time. And in some cases the freeing up of the labor force allowed more people to enter higher skilled managerial jobs and technically specialized jobs, which are typically higher paying. Therefore, fears of unemployment due to automation were generally dismissed as just another instance of the Luddite premise, which had proven fallacious time and again over many decades. Given this obvious empirical contradiction of the premise, people who nevertheless returned to it were usually viewed by the mainstream as cranks misled by quixotic leftist political bias. For example, works by scholars including David F. Noble and Jeremy Rifkin were often respected but discounted. At worst, they were mocked with the disparaging label "neo-Luddite". Noble even wrote a later book titled *Progress Without People: In Defence of Luddism* to try to further explain why the Luddite premise should not be laughed out of academia.

Post-market musings

Rifkin's *End of Work*, published in 1995 and written by a non-engineer, predicted automation-induced unemployment despite having a rather hazy idea of how IT would evolve over the next decade. (The book mentioned the Internet once in passing and the World Wide Web not at all. Its IT focus was mostly on robotics.) Also hazy was Rifkin's explanation of any solution to the problem. The book's subtitle called the solution a "post-market economy", but its concluding chapters did not clearly lay out how such an economy could be engineered, leaving readers to conclude that a non-market solution involving a planned economy was implied between the lines.

In terms of political economy implications, there was no clear differentiation at the time between the ideas of authors like Noble or Rifkin (on the one hand) and traditional leftist agitation (on the other hand). To the extent that readers could ask "What point is this guy getting to?" and answer the question with "socialism" or "a welfare state", they dismissed these authors.

During the 2000s and 2010s

Since the 1990s, the possibility has been raised again in even an apolitical, technocratic way that the Luddite premise (that automation creates unemployment) was only fallacious in the absence of highly advanced and ubiquitous automation, which until recently was mostly out of reach technologically. This would explain why it has always been fallacious until now, but also why it might not always remain so. For example, Marshall Brain, Martin Ford, and others have suggested that exponentially accelerating information technology (IT) may ultimately result in widespread structural unemployment, because an implicit assumption underlying the "eternally fallacious" idea (that lots of regular humans will always find ways to do service work that machines can't do) will itself be fallacious as IT advances. They suggest that, unlike in the 20th century, when the tertiary sector absorbed all of the workers that the automation of the secondary sector expelled, the tertiary sector now also faces depopulation via automation; its employment will shrink, not grow, and this time there is no other sector to backstop the process by absorbing the displaced workers. The high unemployment rates of the late-2000s recession have brought the idea of structural unemployment back into mainstream attention, as observations are made about positions that require extensive specialized skill and experience standing long vacant even while general unemployment rates above 9% (and horror stories of fruitless job searches) would seem to suggest that such vacancies ought to be scarcer. The idea that automation has finally advanced to the point that the Luddite premise is no longer entirely fallacious is one of the components of some theoretical explanations for the string of jobless recoveries in developed economies in recent decades. Expectations that the (already eroding) fallaciousness will fall off sharply in coming decades underlie the fear of structural shift.

Writers such as Rifkin, Brain, and Ford often suggest that the structure of the economy will have to shift to a basic income because its present structural foundation (trading labor for income) will no longer be an available option on a full employment basis. It would perhaps be available to only 90% of workers in the next decade, perhaps 75% of workers a decade after that, and so on. Often included in the basic income idea is an element of civic obligation, such that able people must somehow contribute civically in order to receive the basic income. The labor-market economy (trading labor for income) already achieves that outcome today (because working for income generally produces civic value in various ways, directly and indirectly), but the argument is that advanced automation will decouple the linkage that makes that possible. Thus the same result (trading civic value for income) would have to be driven by different forces—either non-market ones, or via a new kind of market. The non-market idea seems infeasible given the generally abysmal performance record of planned economies. But the idea of engineered new markets leaves room for the disciplining and motivating powers that make capitalist markets capable of positively shaping human behavior where government alone is usually unable.

New-market engineering

Brain and Ford's books, in stark contrast to Rifkin's, came later and were written by engineers with extensive under-the-hood knowledge of modern production methods, computer hardware and software, and the Internet. They explicitly reject non-market solutions as unworkable and instead suggest new kinds of markets. Rather than being "post-market" proponents, such authors could be called "new-market" proponents. They vigorously distance themselves from socialism or welfare states—generally seeking to keep a market economy with private enterprise, which they believe cannot be preserved *unless* its foundation is modified from its current structure. Thus, quite contrary to being anti-market agents (as critics might suppose them to be), they believe themselves to be *salvaging* markets from destruction. They envision creating consumer purchasing power by some other mechanism than the traditional labor market as we have known it so far, in order that free markets may continue to provide the invisible hand component of production-possibilities decisions. In other words, they believe that market forces are necessary to generate allocative efficiency, and they believe that without a structural modification that (at least partially) decouples purchasing power (and consumer confidence) from employment determined by the traditional labor market, there will be a systemic market failure, which they seek to avoid.

Just as new-market advocates are pro-market and pro-private-property, they are also very much non-Luddite (in fact, exactly opposite of Luddite) in the respect that they *like* technology—they don't *hate* it. They want it to continue advancing as robustly as ever. They simply feel that income and purchasing power must be decoupled from human participation in production. (The decoupling does not have to happen all at once; it could start small and gradually increase.) If that happens, then they essentially do not have any problem with technology or automation, per se. In contrast to old-style welfare, they do not feel that income should be unconditional, or equal, or "free" (given out "for nothing"). They believe that people should have to work for it (in a new sense of the word "work"), in the respect that they are given incentives to do positive things, like take classes, read books, conserve (or remediate) environmental resources, and so on. People would be paid to do civically valuable things, and if they chose not to do those things, they would not be paid. In this way, new-market advocates align themselves with human nature, which generally requires selfish motivations and incentives to shape behavior, and with the market's invisible hand, which is needed to make the right production-possibilities decisions (because the idea that individual human managers, or groups of them, are capable of making those decisions correctly with zero invisible-hand assistance has been empirically discredited).

Discussions of new-market ideas usually lead to the topic of post-scarcity economic paradigms. But new-market engineers argue that without clear-eyed, realist engineering of the intermediate steps, there is an abyss of dysfunction and hardship between today's economy of scarcity and the starry-eyed goal of any fully developed economy of abundance.

Wage-recapture market variant

Ford's main new-market mechanism would be to create a tax that recaptures most (not all) of the value that firms and their customers gain from eliminating wages, then use the tax revenue to pay people for doing civically valuable actions—that is, pursuing activities, such as higher education or environmental preservation, that have positive externalities. The main reason for paying these "wages" need not be their altruistic or environmentalist components; the main reason is simply to prevent the market economy from collapsing due to noncirculation of value (that is, the lack of adequate trade which would occur if lack of consumer purchasing power and confidence left no way for an adequate mass market to exist). Ford points out that the tax could not take *all* of the gains away from the corporations and their customers, because this would destroy the natural incentive to innovate that a market economy needs to be sustainable. The value would be split between the innovators, their customers, and the rest of the population, because leaving out any of that trio would wreck the sustainability of the model. (The leaving out of the third leg is what is causing today's economic pathologies and promising tomorrow's, in the view of new-market engineers.) Ford's idea is an earnest market effort because it preserves the invisible hand as the maker of production-possibilities decisions for goods and services. However, it does rely on human planning (via a technocratic government agency in each country) to make the production-possibilities decisions for civic actions. The latter is viewed as unfortunate but necessary due to the lack of an alternative.

Mirror-image market variant

Another idea for a new-market mainspring which solves the aforementioned "lack of an alternative" problem is a "mirror image" idea, which has an even more private-sector approach in which the invisible hand helps make even the civic-actions production-possibilities decisions. In this model, the government does not collect a wage-recapture tax at all. Instead of enforcing tax payment, it only enforces payment of a new-style "wage" directly from corporations to consumers that looks to us today like something we might label "mandatory philanthropy", but which would actually be a true market wage of a new type. In today's old market, money flows from consumers, through (partially automated) companies, past the eyes of the government enforcement sentry (but not through its hands) as wages, into the hands of workers (who are also the consumers, thus completing the cycle of value recirculation). In the new market (mirror-image variant), money would flow from consumers, through (highly automated) companies, past the eyes of the government enforcement sentry (but not through its hands) as [new-style] "wages", and into the hands of [new-style] "wage" earners, who are paid the "wage" for civically valuable actions. (They are also the consumers, thus completing the cycle of value recirculation).

In this model, the decisions about what the civic actions are can be made by the invisible hand, because each "mandated philanthropist" gets a large degree of authority in what actions their "philanthropy" (which is actually [functionally] a new-style "payroll") will or won't pay for. Many such paymasters functioning simultaneously could constitute the "buyers" in a market for civically valuable actions (with mass-market "workers" as the "sellers"). There would still be *some* regulation involved, because, for example, it would be illegal to base the "payroll" decisions on race, color, religion, creed, gender, sexual orientation, ethnicity, disability, marital or veteran status, and so forth. To decide which "workers" were on a given "payroll", there might be a clearinghouse to randomly match the two, rotating assignments every several years. Or perhaps the businesses that run the "payroll" could even "hire" the "workers" themselves, in which case workers would compete for "jobs" by showing off how "productive" they could be in doing the civic actions (another level of invisible hand yet again). The "mirror image" name comes from the idea that this variant of new market is a very free market where the invisible hand remains just as powerful as it was in the 1945-2008 economy, but with many mirror-image aspects (which are visualized above by the amount of quotation marks that are necessary to signify mirror-image senses for words that were always [up till now] widely known only in their non-mirror-image senses).

The axis of reflection in the mirroring seems to be, at root, a "polarity shift" from where human individuals can add value only by *doing production* (from within production systems) to where they can also add value by *avoiding hurting production systems* (from outside). The hurt-avoidance comes from such civic actions as providing goods-and-services demand via consumption (which the system requires in order to stay running) instead of failing to consume (because of lack of income); ensuring the sustainable supply of energy and environmental resources to the production systems (by avoiding *over*consuming those); and ensuring the supply of people educated enough to provide the few humans that the production systems will need in the future, by pursuing education and cognitively enriching pastimes. The humans that the systems need will be few, but those few will need to be highly intelligent, talented, and educated, constituting a human resource that might be endangered if the general population does not act as a "farm team system" for it by valuing education and self-education as a civic action. An analogy is provided by sports' relationship to general life. Few humans are talented and practiced enough to play professional sports, but the professional teams rely on a system that filters such scarce people out of the general population via little league/pee wee programs, high school play, college play, farm-team play, etc. People in the general population are not considered inferior human beings (versus the pro players) because of their lack of pro talent. They are valued as the fans and ticket-buyers that make the pro system economically viable. And a small fraction of them grow up to become pros themselves.

In today's old market, governments enforce the payment of wages by having outlawed their nonpayment (i.e., slavery); by levying tariffs on cheap competition from countries that kept their nonpayment (slavery) (that outlawing has now been global for many decades); and by attempting to minimize their underpayment (i.e., wage slavery, a sharply cheapened value of work [with elites and their customers keeping the money]). In the new market (mirror-image variant), governments enforce the payment of "wages" by outlawing their nonpayment (i.e., evading the "payroll"); by levying tariffs on cheap competition from countries that kept their nonpayment (non-participating countries); and by attempting to minimize their underpayment (i.e., "wage" slavery, a sharply cheapened value of civic actions [with elites and their customers keeping the money]).

One of the inherent challenges of the mirror-image variant is that various forms of dressing up corporations' financial self-interest in a specious cloak of civic virtue would inevitably arise. This would be a "washing" form of marketing and operations that included greenwashing and analogous washing in other domains of life (e.g., education, infrastructure). It seems unlikely that this can be entirely negated; instead, it would have to be perennially pruned by social censure and regulatory oversight. However, no other system is without its chronic weaknesses, either. For example, the classical variants of capitalism (implemented thus far) have scored poorly on various tests, such as environmental sustainability and (potentially) the employability of the average human (as that was traditionally defined) as automation grows pervasive. Twentieth-century variants of communism fared even worse in environmental sustainability, and also failed economically in average standard of living and politically in individual freedom. The wage-recapture new-market variant, with its technocratic decisions on how to spend the revenue, holds promise to minimize the corporate "washing" problem, yet it also holds risks of failing on allocative efficiency and market-driven innovation, which the mirror-image variant mitigates. As elsewhere in reality, each choice has pros and cons, rather than any choice being perfect. The "washing" problem may be the mirror-image analog of classical capitalism's tendency to exaggerate needs (for example, a maker of antibacterial soaps encouraging the populace to fear microbes to an irrational degree).

Both are forms of conflict of interest that cause "chronic irritation" to a socioeconomic system but need not be "fatal" to it if given adequate "medical management". The washing problem may be less systemically injurious than the allocative inefficiency problem, just as the "exaggerated needs" problem of classical capitalism was less systemically injurious than the allocative inefficiency problem of central economic planning.

In choosing the decider of production-possibilities decisions (whether of goods, services, or civic actions), the invisible hand is generally preferred to committees of humans because it has proven to be superior at the decision making (except for regulatory issues such as race-color-religion-etc and the "washing" discussed above). In the future it will also be necessary to ask what role artificial intelligence might possibly have in making those decisions, and whether humans would allow it. Perhaps artificial intelligence, like human intelligence, will share the role with the invisible hand but be barred from usurping the entirety of it.

Implementations

Regarding the chances of any new-market ideas being implemented, there are both significant barriers and significant drivers, with a net potential of perhaps "even chances". Ford discusses many of these barriers and drivers. The barrier side includes (a) natural cultural conservatism that powerfully resists systemic changes; (b) the powerful influence of laissez-faire ideals, which would resist any engineered systemic change to markets (especially *anything* involving a tax); (c) the fact that early implementation by individual countries faces an immediate threat from the export and offshoring competition of countries that *haven't* yet implemented; and (relatedly) (d) the all-or-nothing problem, which may occur if a new system would work well but only if the switch from old to new was an off-on switching rather than a gradual evolution. However, on the driver side there are powerful forces that may answer all of the barriers. Foremost would be a dawning realization by economic elites that they have a choice between a new market with prosperity, or the old market spiraling into near-total failure. Globalization so far has not threatened the wallets of economic elites (only those of average workers), and has in fact enriched the elites thus far; but the changing parameter values of the economic system as automation advances would alter that runtime environment and transform it into a new one, where even the elites' wealth would be threatened by a market failure that killed their businesses and reduced asset values throughout the economy. Realizing these options, elites might actually switch from opposing new markets to actively supporting their implementation (including addressing the competition between countries whose policies differed). The all-or-nothing problem does not have to occur if an implementation is engineered such that extremely profitable, extremely automated industries began piloting new markets while other industries continued with an old-market status quo for quite some time. In this model, the early adopters voluntarily become leaders, and the pilot projects would act simply as economic stimulus on the broader economy (although a type of stimulus much more effective than old-style stimulus, whose efficacy seems to be eroding because it relies on the Luddite premise being a total fallacy as opposed to shifting by degrees out of total fallaciousness). The overall transition in this model could actually be quite painless, as a generally prosperous economy changed gradually over decades from mostly-old-with-some-new to mostly-new-with-some-old. The selfish motivation of the early-adopter leaders would be the aforementioned choice faced by economic elites. They would choose to stimulate the broader economy because that result would ensure their own continuing strong sales and growth by preserving a runtime environment of general prosperity for them to operate within, without which depression or malaise would occur.

Given the aforementioned choice faced by economic elites, those in the private sector might even choose to pursue the new market without government involvement. But the private sector faces two hurdles that would make it difficult: the natural competition between firms (which is necessary and thus must be protected by competition law), and legal obligation to maximize shareholder value. The traditional definitions of shareholder value evolved in an earlier era whose commercial environment had different parameter values due to lack of advanced automation. Those traditional definitions would bar new-style payrolls. But in the face of market failure without them, perhaps a case would emerge for an updated definition. Competition is the other hurdle. Companies are barred by competition law from agreeing to limit competition, and even if they weren't, individual companies generally cannot make the first move of increasing expense without being killed by competition from rivals who don't. This is why "a level playing field" would have to be created by policy, or to use a different analogy, "a high tide that lifted all boats equally". This is directly analogous to existing minimum wage laws. Individual companies generally could not survive in the market if they volunteered to self-enforce minimum limits on wages (in the absence of any laws requiring them). There *is* breathing room for above-market wages (e.g., to attract superior talent) at some companies in some industries who enjoy a relatively high level of imperfect competition; but most companies in most industries face competition too close to pure to survive the attempt. In this sense, the mandated value recirculation (whatever anyone calls it, from "wage recapture" to "new-style wages") is as unremarkable and non-novel an idea as any legislative or regulatory mechanism in commerce. For goals that make long-term systemic balance possible but cannot be pursued by the self-interest of individual market players, these mechanisms provide a path by forcing all competitors to play the game by the same rules. Existing examples include employment standards (e.g., child labor laws, minimum wage laws), environmental protection, and financial regulation (to prevent bubbles and thus crashes). These exist in perennial tension with the forces of pure capitalism; thus the extremes perform checks and balances on each other. Businesses usually fight for inadequate regulation; government usually fights for excessive regulation; and a sustainable balance results. Over decades, systemic pathologies gradually push the balance point out of the sustainable range; periodic breakdowns then yield correction by counteractive forces (e.g., trust-busting [leftward correction], the Reagan revolution [rightward correction]).

Laissez-faire ideals reigned supreme worldwide for about three decades (roughly centered on the fall of the Soviet Bloc, which vindicated capitalism over central planning in many ways). In this environment, where the lesson commonly extrapolated was that pure capitalism will always be better than any mixed-economy alternatives, the prevailing theory has been that higher corporate taxes can only harm economic prosperity. The reasoning is partly that countries can simply compete to undercut each other's corporate tax rates (which is true), but also, more importantly, that only the invisible hand is capable of recirculating capital back toward the base of the economy in a successful manner (which is not to be dismissed lightly, and may in fact be true). The disparaging label for such ideas is "trickle-down economics", but many intelligent people have earnestly believed in these ideals; and the fact that their discounting has often been facile and done by imperfect opponents has only encouraged believers to stay faithful. Widespread fervor for trickle-down beliefs (in both the public and private sectors) poses a formidable barrier to the wage-recapture-tax new-market variant. But these conventional beliefs rely on the assumption that the Luddite premise is entirely and eternally fallacious. Unfortunately, there has already been a decade of empirical evidence that low taxes, new business investment, and economic growth no longer have a sure-fire correlation to strong, "good-jobs" employment in developed economies. If the Luddite premise has been starting to shift into partial accuracy, then no amount of continued low taxes and deregulation will ever be able to produce enough trickling down to create broad-based prosperity. In that case, mandating the payment of new-style wages could recirculate value back to the base of the mass market. The promise of the mirror-image variant would be that humans need not turn to taxes for the value recovery (at the top) nor to central planning for the distribution details (at the bottom). As long as the "minimum wage" (referring to the new-style wages) and other employment standards are being enforced, then government's role ends there.

Conceivably, both the Luddite premise and the lump of labor premise could change states in continuous-graph fashion, from completely false to partially true, depending on parameter values in the commercial environment, most specifically, the available modes of value recirculation. In this view, for two centuries they were completely false or very nearly so, because the traditional labor market provided sufficient means of value recirculation. As that fails because of advanced automation, they could enter partial influence. But if new forms of broad-based personal income came into being (for example, basic income, guaranteed minimum income, or new-style wages), they could revert back to a state of complete or near-complete falsehood again. The difference would be that enough value was circulating broadly enough through a mass market of consumers and corporations that services which today could not possibly garner middle-class wages and benefits (for example, full-time jobs reading stories to hospitalized children, painting murals, or attending university) would become viable at that wage level.

Other goals of automation (beyond productivity gains and cost reduction)

In manufacturing, the purpose of automation has shifted to issues broader than productivity and costs.

Reliability and precision

The old focus on using automation simply to increase productivity and reduce costs was seen to be short-sighted, because it is also necessary to provide a skilled workforce who can make repairs and manage the machinery. Moreover, the initial costs of automation were high and often could not be recovered by the time entirely new manufacturing processes replaced the old. (Japan's "robot junkyards" were once world famous in the manufacturing industry.)

Automation is now often applied primarily to increase quality in the manufacturing process, where automation can increase quality substantially. For example, automobile and truck pistons used to be installed into engines manually. This is rapidly being transitioned to automated machine installation, because the error rate for manual installment was around 1-1.5%, but has been reduced to 0.00001% with automation.

Health and environment

The costs of automation to the environment are different depending on the technology, product or engine automated. There are automated engines that consume more energy resources from the Earth in comparison with previous engines and those that do the opposite too. Hazardous operations, such as oil refining, the manufacturing of industrial chemicals, and all forms of metal working, were always early contenders for automation.

Convertibility and turnaround time

Another major shift in automation is the increased demand for flexibility and convertibility in manufacturing processes. Manufacturers are increasingly demanding the ability to easily switch from manufacturing Product A to manufacturing Product B without having to completely rebuild the production lines. Flexibility and distributed processes have led to the introduction of Automated Guided Vehicles with Natural Features Navigation.

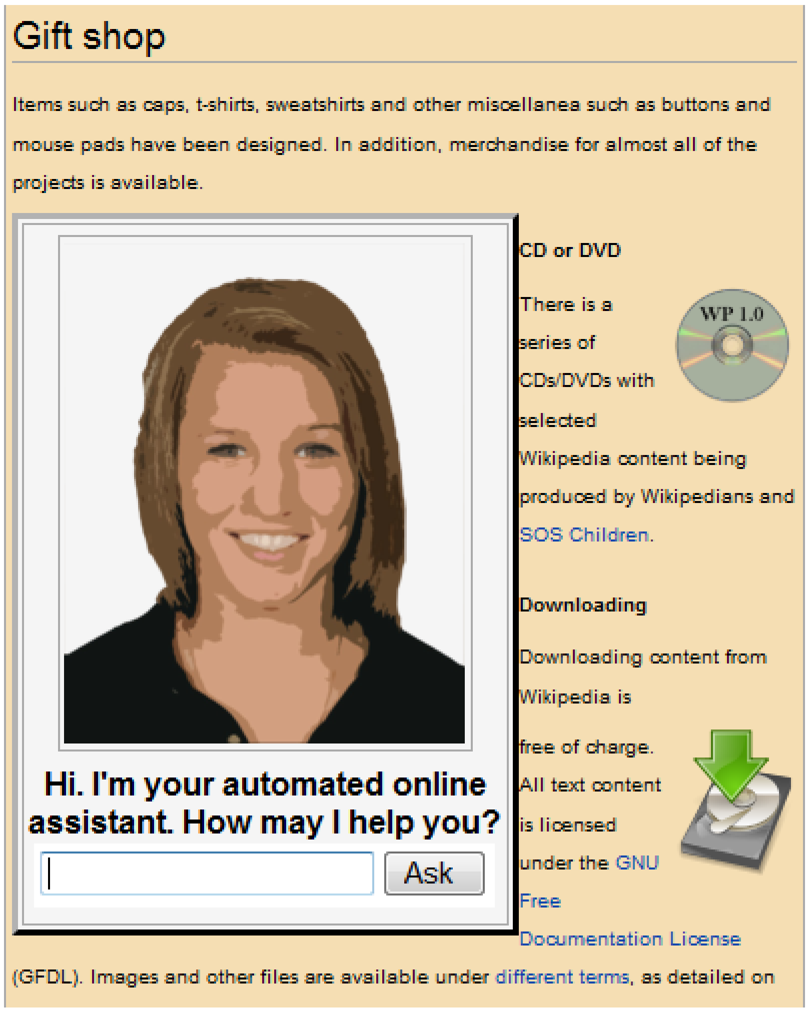
Digital electronics helped too. Former analogue-based instrumentation was replaced by digital equivalents which can be more accurate and flexible, and offer greater scope for more sophisticated configuration, parametrization and operation. This was accompanied by the fieldbus revolution which provided a networked (i.e. a single cable) means of communicating between control systems and field level instrumentation, eliminating hard-wiring.

Discrete manufacturing plants adopted these technologies fast. The more conservative process industries with their longer plant life cycles have been slower to adopt and analogue-based measurement and control still dominates. The growing use of Industrial Ethernet on the factory floor is pushing these trends still further, enabling manufacturing plants to be integrated more tightly within the enterprise, via the internet if necessary. Global competition has also increased demand for Reconfigurable Manufacturing Systems.

Automation tools

Engineers now can have numerical control over automated devices. The result has been a rapidly expanding range of applications and human activities. Computer-aided technologies (or CAx) now serve the basis for mathematical and organizational tools used to create complex systems. Notable examples of CAx include Computer-aided design (CAD software) and Computer-aided manufacturing (CAM software). The improved design, analysis, and manufacture of products enabled by CAx has been beneficial for industry.

Information technology, together with industrial machinery and processes, can assist in the design, implementation, and monitoring of control systems. One example of an industrial control system is a programmable logic controller (PLC). PLCs are specialized hardened computers which are frequently used to synchronize the flow of inputs from (physical) sensors and events with the flow of outputs to actuators and events.



Gift shop

Items such as caps, t-shirts, sweatshirts and other miscellanea such as buttons and mouse pads have been designed. In addition, merchandise for almost all of the projects is available.

Hi. I'm your automated online assistant. How may I help you?

CD or DVD

There is a series of CDs/DVDs with selected Wikipedia content being produced by Wikipedians and SOS Children.

Downloading

Downloading content from Wikipedia is free of charge. All text content is licensed under the GNU Free Documentation License (GFDL). Images and other files are available under different terms, as detailed on

An automated online assistant on a website, with an avatar for enhanced human–computer interaction.

Human-machine interfaces (HMI) or computer human interfaces (CHI), formerly known as *man-machine interfaces*, are usually employed to communicate with PLCs and other computers. Service personnel who monitor and control through HMIs can be called by different names. In industrial process and manufacturing environments, they are called operators or something similar. In boiler houses and central utilities departments they are called stationary engineers.

Different types of automation tools exist:

- ANN - Artificial neural network
- DCS - Distributed Control System
- HMI - Human Machine Interface
- SCADA - Supervisory Control and Data Acquisition
- PLC - Programmable Logic Controller
- PAC - Programmable automation controller
- Instrumentation
- Motion control
- Robotics

Current limits

Many roles for humans in industrial processes presently lie beyond the scope of automation. Human-level pattern recognition, language recognition, and language production ability are well beyond the capabilities of modern mechanical and computer systems. Tasks requiring subjective assessment or synthesis of complex sensory data, such as scents and sounds, as well as high-level tasks such as strategic planning, currently require human expertise. In many cases, the use of humans is more cost-effective than mechanical approaches even where automation of industrial tasks is possible.

Applications of Automation

- **Automated Video surveillance:**

The Defense Advanced Research Projects Agency (DARPA) started the research and development of automated Visual surveillance and Monitoring (VSAM) program 1997-99 and airborne Video Surveillance (AVS) program 1998-2002. Currently there is a major effort underway in the vision community to develop a fully automated tracking surveillance system. Automated video surveillance monitors people and vehicle in real time within a busy environment. Existing automated surveillance systems are based on the environment they are primarily designed to observe, i.e., indoor, outdoor or airborne, the amount of sensors that the automated system can handle and the mobility of sensor, i.e., stationary camera vs. mobile camera. The purpose of a surveillance system is to record properties and trajectories of objects in a given area, generate warnings or notify designated authority in case of occurrence of particular events.

- **Automated Highway Systems:**

As demands for safety and mobility have grown and technological possibilities have multiplied, interest in automation have grown. Seeking to accelerate the development and introduction of fully automated vehicles and highways, The United States Congress authorized more than $650 million over 6 years for intelligent transport systems (ITS) and demonstration projects in the 1991 Intermodal Surface Transportation Efficiency Act (ISTEA). Congress legislated in ISTEA that “The secretary [of transportation] shall develop an automated highway and vehicle prototype from which future fully automated intelligent vehicle-highway systems can be developed. Such development shall include research in human factors to ensure the success of the man-machine relationship. The goal of this program is to have the first fully automated highway roadway or an automated test track in operation by 1997. This system shall accommodate installation of equipment in new and existing motor vehicles." [ISTEA 1991, part B, Section 6054(b)].

Full automation commonly defined as requiring no control or very limited control by the driver; such automation would be accomplished through a combination of sensor, computer, and communications systems in vehicles and along the roadway. Fully automated driving would, in theory, allow closer vehicle spacing and higher speeds, which could enhance traffic capacity in places where additional road building is physically impossible, politically unacceptable, or prohibitively expensive. Automated controls also might enhance road safety by reducing the opportunity for driver error, which causes a large share of motor vehicle crashes. Other potential benefits include improved air quality (as a result of more-efficient traffic flows), increased fuel economy, and spin-off technologies generated during research and development related to automated highway systems.

- **Automated manufacturing:**

Automated manufacturing refers to the application of automation to produce things in the factory way. Most of the advantages of the automation technology has its influence in the manufacture processes.

The main advantage of the automated manufacturing are: higher consistency and quality, reduce the lead times, simplification of production, reduce handling, improve work flow and increase the morale of workers when a good implementation of the automation is made.

- **Home Automation**

Home automation (also called domotics) designates an emerging practice of increased automation of household appliances and features in residential dwellings, particularly through electronic means that allow for things impracticable, overly expensive or simply not possible in recent past decades.

Chapter-5

Distributed Control System

A **distributed control system** (DCS) refers to a control system usually of a manufacturing system, process or any kind of dynamic system, in which the controller elements are not central in location (like the brain) but are distributed throughout the system with each component sub-system controlled by one or more controllers. The entire system of controllers is connected by networks for communication and monitoring.

DCS is a very broad term used in a variety of industries, to monitor and control distributed equipment.

- Electrical power grids and electrical generation plants
- Environmental control systems
- Traffic signals
- radio signals
- Water management systems
- Oil refining plants
- Metallurgical Process Plants
- Chemical plants
- Pharmaceutical manufacturing
- Sensor networks
- Dry cargo and bulk oil carrier ships

Elements

A DCS typically uses custom designed processors as controllers and uses both proprietary interconnections and communications protocol for communication. Input and output modules form component parts of the DCS. The processor receives information from input modules and sends information to output modules. The input modules receive information from input instruments in the process (a.k.a. field) and transmit instructions to the output instruments in the field. Computer buses or electrical buses connect the processor and modules through multiplexer or demultiplexers. Buses also connect the distributed controllers with the central controller and finally to the Human-Machine Interface (HMI) or control consoles.

Elements of a distributed control system may directly connect to physical equipment such as switches, pumps and valves or may work through an intermediate system such as a SCADA system.

Applications

Distributed Control Systems (DCSs) are dedicated systems used to control manufacturing processes that are continuous or batch-oriented, such as oil refining, petrochemicals, central station power generation, pharmaceuticals, food & beverage manufacturing, cement production, steelmaking, and papermaking. DCSs are connected to sensors and actuators and use setpoint control to control the flow of material through the plant. The most common example is a setpoint control loop consisting of a pressure sensor, controller, and control valve. Pressure or flow measurements are transmitted to the controller, usually through the aid of a signal conditioning Input/Output (I/O) device. When the measured variable reaches a certain point, the controller instructs a valve or actuation device to open or close until the fluidic flow process reaches the desired setpoint. Large oil refineries have many thousands of I/O points and employ very large DCSs. Processes are not limited to fluidic flow through pipes, however, and can also include things like paper machines and their associated quality controls, variable speed drives and motor control centers, cement kilns, mining operations, ore processing facilities, and many others.

A typical DCS consists of functionally and/or geographically distributed digital controllers capable of executing from 1 to 256 or more regulatory control loops in one control box. The input/output devices (I/O) can be integral with the controller or located remotely via a field network. Today's controllers have extensive computational capabilities and, in addition to proportional, integral, and derivative (PID) control, can generally perform logic and sequential control.

DCSs may employ one or several workstations and can be configured at the workstation or by an off-line personal computer. Local communication is handled by a control network with transmission over twisted pair, coaxial, or fiber optic cable. A server and/or applications processor may be included in the system for extra computational, data collection, and reporting capability.

History

Early minicomputers were used in the control of industrial processes since the beginning of the 1960s. The IBM 1800, for example, was an early computer that had input/output hardware to gather process signals in a plant for conversion from field contact levels (for digital points) and analog signals to the digital domain.

The first industrial control computer system was built 1959 at the Texaco Port Arthur, Texas, refinery with an RW-300 of the Ramo-Wooldridge Company.

The DCS was introduced in 1975. Both Honeywell and Japanese electrical engineering firm Yokogawa introduced their own independently produced DCSs at roughly the same time, with the TDC 2000 and CENTUM systems, respectively. US-based Bristol also introduced their UCS 3000 universal controller in 1975. In 1980, Bailey (now part of ABB) introduced the NETWORK 90 system. Also in 1980, Fischer & Porter Company (now also part of ABB) introduced DCI-4000 (DCI stands for Distributed Control Instrumentation).

The DCS largely came about due to the increased availability of microcomputers and the proliferation of microprocessors in the world of process control. Computers had already been applied to process automation for some time in the form of both Direct Digital Control (DDC) and Set Point Control. In the early 1970s Taylor Instrument Company, (now part of ABB) developed the 1010 system, Foxboro the FOX1 system and Bailey Controls the 1055 systems. All of these were DDC applications implemented within minicomputers (DEC PDP-11, Varian Data Machines, MODCOMP etc.) and connected to proprietary Input/Output hardware. Sophisticated (for the time) continuous as well as batch control was implemented in this way. A more conservative approach was Set Point Control, where process computers supervised clusters of analog process controllers. A CRT-based workstation provided visibility into the process using text and crude character graphics. Availability of a fully functional graphical user interface was a way away.

Central to the DCS model was the inclusion of control function blocks. Function blocks evolved from early, more primitive DDC concepts of "Table Driven" software. One of the first embodiments of object-oriented software, function blocks were self contained "blocks" of code that emulated analog hardware control components and performed tasks that were essential to process control, such as execution of PID algorithms. Function blocks continue to endure as the predominant method of control for DCS suppliers, and are supported by key technologies such as **Foundation Fieldbus** today.

Midac Systems of Sydney Australia developed an objected-oriented distributed direct digital control system in 1982. The central system ran 11 microprocessors sharing tasks and common memory and connected to a serial communication network of distributed controllers each running two Z80s. The system was installed at the University of Melbourne.

Digital communication between distributed controllers, workstations and other computing elements (peer to peer access) was one of the primary advantages of the DCS. Attention was duly focused on the networks, which provided the all-important lines of communication that, for process applications, had to incorporate specific functions such as determinism and redundancy. As a result, many suppliers embraced the IEEE 802.4 networking standard. This decision set the stage for the wave of migrations necessary when information technology moved into process automation and IEEE 802.3 rather than IEEE 802.4 prevailed as the control LAN.

The Network Centric Era of the 1980s

The DCS brought distributed intelligence to the plant and established the presence of computers and microprocessors in process control, but it still did not provide the reach and openness necessary to unify plant resource requirements. In many cases, the DCS was merely a digital replacement of the same functionality provided by analog controllers and a panelboard display. This was embodied in The Purdue Reference Model (PRM) that was developed to define Manufacturing Operations Management relationships. PRM later formed the basis for ISA95 standards activities today.

In the 1980s, users began to look at DCSs as more than just basic process control. A very early example of a Direct Digital Control DCS was completed by the Australian business Midac in 1981-1982 using R-Tec Australian designed hardware. The system installed at the University of Melbourne used a serial communications network, connecting campus buildings back to a control room "front end". Each remote unit ran 2 Z80 microprocessors whilst the front end ran 11 in a Parallel Processing configuration with paged common memory to share tasks and could run up to 20,000 concurrent controls objects.

It was believed that if openness could be achieved and greater amounts of data could be shared throughout the enterprise that even greater things could be achieved. The first attempts to increase the openness of DCSs resulted in the adoption of the predominant operating system of the day: *UNIX*. UNIX and its companion networking technology TCP-IP were developed by the Department of Defense for openness, which was precisely the issue the process industries were looking to resolve.

As a result suppliers also began to adopt Ethernet-based networks with their own proprietary protocol layers. The full TCP/IP standard was not implemented, but the use of Ethernet made it possible to implement the first instances of object management and global data access technology. The 1980s also witnessed the first PLCs integrated into the DCS infrastructure. Plant-wide historians also emerged to capitalize on the extended reach of automation systems. The first DCS supplier to adopt UNIX and Ethernet networking technologies was Foxboro, who introduced the I/A Series system in 1987.

The Application Centric Era of the 1990s

The drive toward openness in the 1980s gained momentum through the 1990s with the increased adoption of Commercial off-the-shelf (COTS) components and IT standards. Probably the biggest transition undertaken during this time was the move from the UNIX operating system to the Windows environment. While the realm of the real time operating system (RTOS) for control applications remains dominated by real time commercial variants of UNIX or proprietary operating systems, everything above real-time control has made the transition to Windows.

The introduction of Microsoft at the desktop and server layers resulted in the development of technologies such as OLE for Process Control (OPC), which is now a de facto industry connectivity standard. Internet technology also began to make its mark in automation and the DCS world, with most DCS HMI supporting Internet connectivity. The '90s were also known for the "Fieldbus Wars", where rival organizations competed to define what would become the IEC fieldbus standard for digital communication with field instrumentation instead of 4-20 milliamp analog communications. The first fieldbus installations occurred in the 1990s. Towards the end of the decade, the technology began to develop significant momentum, with the market consolidated around Ethernet I/P, Foundation Fieldbus and Profibus PA for process automation applications. Some suppliers built new systems from the ground up to maximize functionality with fieldbus, such as Rockwell PAX System Honeywell with Experion & Plantscape SCADA systems, ABB with System 800xA, Emerson Process Management with the DeltaV control system, Siemens with the Simatic PCS7 and **azbil** from Yamatake with the Harmonas-DEO system.

The impact of COTS, however, was most pronounced at the hardware layer. For years, the primary business of DCS suppliers had been the supply of large amounts of hardware, particularly I/O and controllers. The initial proliferation of DCSs required the installation of prodigious amounts of this hardware, most of it manufactured from the bottom up by DCS suppliers. Standard computer components from manufacturers such as Intel and Motorola, however, made it cost prohibitive for DCS suppliers to continue making their own components, workstations, and networking hardware.

As the suppliers made the transition to COTS components, they also discovered that the hardware market was shrinking fast. COTS not only resulted in lower manufacturing costs for the supplier, but also steadily decreasing prices for the end users, who were also becoming increasingly vocal over what they perceived to be unduly high hardware costs. Some suppliers that were previously stronger in the PLC business, such as Rockwell Automation and Siemens, were able to leverage their expertise in manufacturing control hardware to enter the DCS marketplace with cost effective offerings, while the stability/scalability/reliability and functionality of these emerging systems are still improving. The traditional DCS suppliers introduced new generation DCS System based on the latest Communication and IEC Standards, which resulting in a trend of combining the traditional concepts/functionalities for PLC and DCS into a one for all solution—named "Process Automation System". The gaps among the various systems remain at the areas such as: the database integrity, pre-engineering functionality, system maturity, communication transparency and reliability. While it is expected the cost ratio is relatively the same (the more powerful the systems are, the more expensive they will be), the reality of the automation business is often operating strategically case by case. The current next evolution step is called Collaborative Process Automation Systems.

To compound the issue, suppliers were also realizing that the hardware market was becoming saturated. The lifecycle of hardware components such as I/O and wiring is also typically in the range of 15 to over 20 years, making for a challenging replacement market. Many of the older systems that were installed in the 1970s and 1980s are still in use today, and there is a considerable installed base of systems in the market that are approaching the end of their useful life. Developed industrial economies in North

America, Europe, and Japan already had many thousands of DCSs installed, and with few if any new plants being built, the market for new hardware was shifting rapidly to smaller, albeit faster growing regions such as China, Latin America, and Eastern Europe.

Because of the shrinking hardware business, suppliers began to make the challenging transition from a hardware-based business model to one based on software and value-added services. It is a transition that is still being made today. The applications portfolio offered by suppliers expanded considerably in the '90s to include areas such as production management, model-based control, real-time optimization, Plant Asset Management (PAM), Real Time Performance Management (RPM) tools, alarm management, and many others. To obtain the true value from these applications, however, often requires a considerable service content, which the suppliers also provide.

Chapter-6

Industrial Ethernet

Industrial Ethernet (IE) is the name given to the use of Ethernet networking in an industrial environment, for automation and process control. A number of techniques are used to adapt Ethernet for the needs of industrial processes, which must provide real time behavior. By using standard Ethernet, automation systems from different manufacturers can be interconnected throughout a process plant. Industrial Ethernet takes advantage of the relatively larger marketplace for computer interconnections using Ethernet to reduce cost and improve performance of communications between industrial controllers.

IE components used in plant process areas must be designed to work in harsh environments of temperature extremes, humidity, and vibration that exceed the ranges for information technology equipment intended for installation in controlled environments.

Example

Beer takes several days to weeks to produce and requires the monitoring and management of temperature, pressure, liquid flow, stirring, adding ingredients and much more that is handled by the Production IE network. This is commonly referred to as ICT or Industrial Control Technology that is connected through the IE.

A major beer brewer once had his production network (IE) go down for a period of several hours and during that time they could not be sure that the brew was being kept at the correct temperature and that the correct items had been added on time and given the correct time to mix or cure. The result was that the brew had to be dumped. Not only did this result in a lot of non productive work, the cost of a lost brew of several thousand bottles of beer but the clean up and loss of all the ingredients and the time lost. If this beer had been bottled and it was bad or spoiled it would have resulted in lawsuits and loss of market share another major cost. If this had been a chemical plant the results could have been deadly and an even more expensive cost/loss.

This is a simple but powerful example of the need for IE and that it cannot be managed as the best effort, user focused IT world of today.

Advantages and difficulties

Industrial Ethernet Protocols - Until recently, a PLC (Programmable logic controller) would communicate with a slave machine using one of several possible open or proprietary protocols, such as Modbus, Sinec H1, Profibus, CANopen, DeviceNet or FOUNDATION Fieldbus. However, there is now increasing interest in the use of Ethernet as the link-layer protocol, with one of the above protocols as the application-layer.

Some of the advantages are:

- Increased speed, up from 9.6 kbit/s with RS-232 to 1 Gbit/s with IEEE 802 over Cat5e/Cat6 cables or optical fiber
- Increased overall performance
- Increased distance
- Ability to use standard access points, routers, switches, hubs, cables and optical fiber, which are immensely cheaper than the equivalent serial-port devices
- Ability to have more than two nodes on link, which was possible with RS-485 but not with RS-232
- Peer-to-peer architectures may replace master-slave ones
- Better interoperability

The difficulties of using Industrial Ethernet are:

- Migrating existing systems to a new protocol (however, many adapters are available)
- Real-time uses may suffer for protocols using TCP (but some use UDP and layer 2 protocols for this reason)
- Managing a whole TCP/IP stack is more complex than just receiving serial data
- The minimum Fast Ethernet frame size including inter-frame spacing is about 80 bytes, while typical industrial communication data sizes can be closer to 1-8 bytes. This often results in a data transmission efficiency of less than 5%, negating any advantages of the higher bitrate.
    - On Gigabit Ethernet the minimum frame size is 512Bytes, reducing the typical efficiency to less than 1%.
    - Some of the Industrial Ethernet protocols introduce modifications to the Ethernet protocol to improve efficiency.

Main protocols

Serial	Ethernet	Protocol	Network	Standards
Modbus-RTU	Modbus-TCP	TCP/IP		IEC 61158 and IEC 61784
Profibus	PROFINET IO	Isochronous real time protocol (IRT), Real time protocol (RT), Real time over UDP protocol (RTU)	Switches, router and wireless, from 100 Mbit/s up to 1 Gbit/s	IEC 61158 and IEC 61784
DeviceNet (CIP); ControlNet (CIP)	Ethernet/IP (CIP)	TCP/IP; UDP/IP	Switches, router and wireless, from 100 Mbit/s up to 1 Gbit/s	IEC 61158 and IEC 61784; ODVA EtherNet/IP standard
Foundation Fieldbus H1	Foundation Fieldbus High Speed Ethernet (HSE)			
CANopen	Ethernet Powerlink		Ethernet 100Mbit/s	IEC 61158, EPSG (Ethernet Powerlink Standardization Group)
CANopen	EtherCAT	EtherCAT, EtherCAT/UDP	Ethernet 100Mbit/s	IEC 61158, IEC/PAS 62407, IEC 61784-3, ISO 15745-4
	VARAN **V**ersatile **A**utomation **R**andom **A**ccess **N**etwork	VARAN, TCP/IP, Safety	Ethernet 100Mbit/s	VARAN-BUS USER GROUP - VNO
SERCOS I / II	SERCOS III		Ethernet 100Mbit/s	IEC 61491, merged into IEC 61158

FL-Net (OPCN-2)	UDP/IP	Ethernet 10Mbit/s	by JEMA (Japan Electrical Manufacturers' Association)

(Note the highly ambiguous name given the Ethernet version of DeviceNet. The "IP" in Ethernet/IP stands for Industrial Protocol.)

Chapter-7

EtherCAT

EtherCAT - Ethernet for Control Automation Technology - is an open high performance Ethernet-based fieldbus system. The development goal of EtherCAT was to apply Ethernet to automation applications which require short data update times (also called cycle times) with low communication jitter (for synchronization purposes) and low hardware costs.

Introduction

Typical automation networks are characterized by short data length per node, typically less than the minimum payload of an Ethernet frame. Using one frame per node per cycle therefore leads to low bandwidth utilization and thus to poor overall network performance. EtherCAT therefore takes a different approach, called "processing on the fly".

Functional principle

With EtherCAT, the Ethernet packet or frame is no longer received, then interpreted and copied as process data at every node. The EtherCAT slave devices read the data addressed to them while the telegram passes through the device. Similarly, input data are inserted while the telegram passes through. The frames are only delayed by a fraction of a microsecond in each node, and many nodes - typically the entire network - can be addressed with just one frame.

Protocol

The EtherCAT protocol is optimised for process data and is transported directly within the standard IEEE 802.3 Ethernet frame using Ethertype 0x88a4. It may consist of several sub-datagrams, each serving a particular memory area of the logical process images that can be up to 4 gigabytes in size. The data sequence is independent of the physical order of the nodes in the network; addressing can be in any order. Broadcast, multicast and communication between slaves are possible and must be done by the master device. If IP routing is required, the EtherCAT protocol can be inserted into

UDP/IP datagrams. This also enables any control with Ethernet protocol stack to address EtherCAT systems.

Performance

Short cycle times can be achieved since the host microprocessors in the slave devices are not involved in the processing of the Ethernet packets to transfer the process images. All process data communication is handled in the slave controller hardware. Combined with the functional principle this makes EtherCAT a high performance distributed I/O system: Process data exchange with 1000 distributed digital I/O takes about 30 µs, which is typical for a transfer of 125 byte over 100Mb/s Ethernet. Data for and from 100 servo axis can be updated with up to 10 kHz. Typical network update rates are 1–30 kHz, but EtherCAT can be used with slower cycle times, too, if the DMA load is too high on your PC.

Topology

Using full-duplex Ethernet physical layers, the EtherCAT slave controllers close an open port automatically and return the Ethernet frame if no downstream device is detected. Slave devices may have two or more ports. Due to these features EtherCAT can support almost any physical topology such as line, tree or star. The bus or line structure known from the fieldbusses thus also becomes available for Ethernet. Also possible is the combination of line and branches or stubs: any EtherCAT device with three or more ports can act as junction, no additional switches are required. The classic switch-based Ethernet star topology can be used either with switches configured to forward traffic directly between ports, or with special slave devices: the switches are then located between the network master and the slave devices. The special slave device (remember standard slave devices don't have a MAC address) assembly attached to one switch port together forms an EtherCAT segment, which is either addressed via its MAC address or via port based VLANs. Since 100BASE-TX Ethernet physical layer is used, the distance between any two nodes can be up to 100 m (300 ft). Up to 65535 devices can be connected per segment. If an EtherCAT network is wired in ring configuration (requires two ports on the master device), it can provide cable redundancy.

Synchronization

For synchronization a distributed clock mechanism is applied, which leads to very low jitters of significantly less than 1 µs even if the communication cycle jitters, which is equivalent to the IEEE 1588 Precision Time Protocol standard. Therefore EtherCAT does not require a special hardware in the master device and can be implemented in software on any standard Ethernet MAC, even without dedicated communication coprocessor.

The typical process of establishing a distributed clock is initiated by the master by sending a broadcast to all slaves to a certain address. Upon reception of this message, all slaves will latch the value of their internal clock twice, once when the message is received and once when it returns (remember EtherCAT has a ring topology). The master can then read all latched values and calculate the delay for each slave. This process can be repeated as many times as required to reduce jitter and average out values. Total delays are calculated for each slave depending on their position in the slave-ring and will be uploaded to an offset register. Finally the master issues a broadcast readwrite on the system clock, which will make the first slave the reference clock and forcing all other slaves to set their internal clock appropriately with the now known offset.

To keep the clocks synchronised after initialization, the master or slave must regularly send out the broadcast again to counter any effects of speed difference between the internal clocks of each slave. Each slave should adjust the speed of their internal clock or implement an internal correction mechanism whenever they have to adjust.

The system clock is specified as a 64 bit counter with a base unit of 1 ns starting at January 1, 2000, 0:00.

Device profiles

The device profiles describe the application parameters and the functional behaviour of the devices including the device class-specific state machines. For many device classes, fieldbus technology already offers reliable device profiles, for example for I/O devices, drives or valves. EtherCAT supports both the CANopen device profile family as well as the drive profile known as the Sercos drive profile. Since the application view does not change when migrating from CANopen or Sercos, this assists users and device manufacturers alike.

Functional safety

The protocol enhancement called *Safety over EtherCAT* enables safety-related communication and control communication on the same network. The safety protocol is based on the application layer of EtherCAT, without influencing the lower layers. It is certified according to IEC 61508 and meets the requirements of Safety Integrity Level (SIL) 3. Certified products using the Safety over EtherCAT protocol have been available since 2005.

Gateways

For integration of existing fieldbus components (e.g., CANopen, DeviceNet, Profibus) into EtherCAT networks gateway devices are available. Also other Ethernet protocols can be used in conjunction with EtherCAT: The Ethernet frames are tunneled via the EtherCAT protocol, which is the standard approach for internet applications (e.g. VPN, PPPoE (DSL) etc.). The EtherCAT network is fully transparent for the Ethernet device, and the real-time characteristics are not impaired since the master dictates exactly when the tunneled transfers are to occur and how much capacity of the 100Mb/s media the tunneled protocols can use. All internet technologies can therefore also be used in the EtherCAT environment: integrated web server, e-mail, FTP transfer etc.

Implementation

Master can be implemented in software on any standard Ethernet MAC. Several vendors supply code for different operating systems. There are also several open and shared source implementations. For slave devices special EtherCAT slave controller chips are required in order to perform the "processing on the fly" principle. EtherCAT slave controllers are available as code for different FPGA types and are also available as ASIC implementations.

EtherCAT Technology Group

The EtherCAT Technology Group (ETG) is international user and vendor organization headquartered in Nuremberg (Germany). It was founded in November 2003 and has offices in Tokyo (Japan), Beijing (China), Seoul (Korea), and Austin, Tx (USA). As of June 2010, it has over 1350 member companies from 50 countries. The ETG considers itself to be a forum for end users from different sectors, and for machine manufacturers and suppliers of control technology with the aim of supporting and promoting EtherCAT. The ETG provides information about EtherCAT and its application, organizes technical training classes, has technical and marketing committees, and promotes EtherCAT on trade shows in major industrial markets.

International standardization

The EtherCAT Technology Group is an official liaison partner of the IEC (International Electrotechnical Commission) working groups for digital communication. The EtherCAT specification was published as IEC/PAS 62407 in 2005, which was removed end of 2007 since EtherCAT had been integrated into the international fieldbus standards IEC 61158 and IEC 61784-2 as well as into the drive profile standard IEC 61800-7. These IEC standards have been approved unanimously in September and October 2007 and were published as IS (International Standards) later that year. In IEC 61800-7, EtherCAT is a standardized communication technology for the SERCOS and CANopen drive profiles. EtherCAT is also part of ISO 15745-4, the standard for XML device description. Furthermore, SEMI has added EtherCAT to its standards portfolio (E54.20) and approved the technology for usage in semiconductor and flat panel display manufacturing equipment. In April 2010, Edition 2 of IEC 61784-3 was accepted, which contains the Safety over EtherCAT Protocol. In September 2008, the EtherCAT Installation Profile was submitted to IEC 61784-5.

Applications

Typical application fields for EtherCAT are machine controls (e.g. semiconductor tools, metal forming, packaging, injection molding, assembly systems, printing machines, robotics).

Chapter-8

EtherNet/IP and SERCOS Interface

EtherNet/IP

EtherNet/IP (Ethernet Industrial Protocol) is a communications protocol developed by Rockwell Automation, managed by ODVA and designed for use in process control and other industrial automation applications. EtherNet/IP is an application layer protocol and it considers all the devices on the network as a series of "objects". EtherNet/IP is built on the widely used (CIP), which makes seamless access to objects from controlNet and DeviceNet networks.

EtherNet/IP makes use of existing network infrastructure (Ethernet) and the entire EtherNet/IP stack can be implemented in software on a microprocessor. No special hardware such as (Application-specific integrated circuits) or (Field Programmable Gate Arrays) are required for implementing EtherNet/IP.

EtherNet/IP is built on the standard TCP/IP stack, and makes use of all 7 layers of OSI reference model. Since EtherNet/IP makes use of Ethernet for physical layer, it becomes easier to tightly couple Enterprise Servers with the device data (sensors, actuators, drives, valves etc...).

As the technology for improving Ethernet physical layer progresses, EtherNet/IP will benefit indirectly. So, as the Ethernet physical layer technology advances from 10/100 Mbit/s to 1 Gbit/s, EtherNet/IP data transfers also get faster.

EtherNet/IP can be used in automation networks which can tolerate some amount of non-determinism. This is because Ethernet physical media, by definition, is not deterministic.

EtherNet/IP can be easily confused as a combination of Ethernet (the physical layer, link, or medium used in most office and many industrial networking environments) and the Internet Protocol, the world's most ubiquitous (internet) networking protocol and part of the TCP/IP model, comprising a suite of protocols operating at the link, internet (or networking), transport, and application layers.

By comparison, **EtherNet/IP** is an industrial application layer protocol operating over the Ethernet medium and used for communication between industrial control systems and their components, such as a programmable automation controller, programmable logic controller or an I/O system. Furthermore, the "IP" in EtherNet/IP, is not an abbreviation for "Internet Protocol" but instead stands for "Industrial Protocol", referring to Rockwell's adoption of Common Industrial Protocol (Common Industrial Protocol) standards as EtherNet/IP was developed.

Confusing matters further is the fact that when using the OSI Reference Model, EtherNet/IP, at the application layer, operates over both Ethernet (at the physical layer) and IP (at the network layer) and thus complements each of these technologies.

History

EtherNet/IP was developed in the late 1990s by Rockwell Automation as part of Rockwell's industrial Ethernet networking solutions. With little or no input taken from other ODVA members, Rockwell gave EtherNet/IP its moniker and handed it over to ODVA, which now manages the protocol and assures multi-vendor system interoperability by requiring adherence to established standards whenever new products that utilize the protocol are developed.

Today, EtherNet/IP is most commonly used in industrial automation settings (e.g., water processing plants, manufacturing facilities, utilities) in the US and Asia for communication to and from Rockwell Automation's Allen-Bradley-brand control systems. However, other industrial automation and control system vendors, including Phoenix Contact, Opto 22 and WAGO Corporation, have developed programmable automation controllers and I/O capable of communicating via EtherNet/IP.

Schneider-Electric developed an EtherNet/IP cards for PLC Quantum, Premium and M340 on the Unity Platform.

Technical Details

EtherNet/IP classifies Ethernet nodes as predefined device types with specific behaviors. Among other things, this enables:

- Transfer of basic I/O data via UDP-based implicit messaging
- Uploading and downloading of parameters, setpoints, programs and recipes via TCP (i.e., explicit messaging.)
- Polled, cyclic and change-of-state monitoring via UDP, such as RPI and COS in Allen Bradley's ControlLogix control systems.
- One-to-one (unicast), one-to-many (multicast), and one-to-all (broadcast) communication via TCP.

- EtherNet/IP makes use of well known TCP port number 44818 for explicit messaging and UDP port number 2222 for implicit messaging

The EtherNet/IP application layer protocol is based on the Common Industrial Protocol (CIP) standard used in DeviceNet, CompoNet and ControlNet.

As EtherNet/IP is now an open technology, it was suggested to publish the Level 2 source codes via sourceforge.net. However, in lieu of this, freeware source code was available to be downloaded from ODVA's website. At this point in time the ODVA requires that users be registered which means that a vendor ID is required and the code and the standard can no longer be considered free.

SERCOS interface

Introduction

In the field of Industrial Control Systems, the interfacing of various control components must provide a means to coordinate the signals and commands sent between control modules. While tight coordination is desirable for discrete inputs and outputs, it is especially important in motion controls, where directing the movement of individual axis of motion must be precisely coordinated so that the motion of the entire system follows a desired path. Types of equipment requiring such coordination are for example metal cutting machine tools, metal forming equipment, assembly machinery, packaging machinery, robotics, printing machinery and material handling equipment. The **SERCOS** (**SE**rial **R**eal-time **CO**mmunication **S**ystem) interface is a globally standardized open digital interface for the communication between industrial controls, motion devices (drives) and input output devices (I/O). It is classified as standard IEC 61491 and EN 61491. The SERCOS interface is designed to provide hard real-time, high performance communications between industrial motion controls and digital servo drives.

History

Until the early 1980's the majority of servo drive systems used to control motion in industrial machinery were based upon analog electronics. The accepted interface to control such devices was an analog voltage signal, where polarity represented the desired direction of motion, and magnitude represented the desired speed or torque. In the 1980s, drive systems and devices based on digital technology began to emerge. A new method needed to be devised to communicate with, and control such units, as their capabilities could not be exploited with the traditional interface method used with analog drives. The earliest interfaces were either proprietary to one vendor or designed only for a single purpose, making it difficult for users of motion control systems to freely interchange motion control and drives. The membership of the VDW (German Machine Tool Builders' Association) became concerned with the implications of this trend. In response to that, in 1987 the VDW formed a joint working group with the ZVEI (German Electrical and Electronics Industry Association) to develop an open interface specification that was appropriate for digital drive systems. The resulting specification, entitled “SERCOS (**SE**rial **R**eal-time **CO**mmunication **S**ystem) interface, was released and later submitted to the IEC, which in 1995 released it as IEC 61491. After the release of the original standard, original working group member companies including ABB, AEG, AMK, Robert Bosch, Indramat, and Siemens founded the "Interest Group SERCOS" to steward the standard. Over the history of the SERCOS interface, its capabilities have been enhanced to the point where today it is not only used for motion control systems, but also for the control of I/O on machinery, as a single machine network.

Versions

SERCOS-I was released in 1991. The transmission medium used is optical fiber. The data rates supported are 2 and 4 MBit/s, and cyclic update rates as low as 62.5 microseconds. A ring topology is used. SERCOS-I also supports a "Service Channel" which allows asynchronous communication with slaves for less time-critical data.

SERCOS-II was introduced in 1999. It expanded the data rates supported to 2, 4, 8 and 16 MBit/s.

SERCOS-III merges the hard-real-time aspects of the SERCOS interface with the Ethernet standard.

SERCOS interface Features

Important features of the SERCOS interface include:

- Collision-free communication through the use of a time-slot mechanism.
- Highly efficient communication protocol (little overhead).
- Extremely low telegram jitter (specified at less than 1 microsecond, in practice as low as 35 nanoseconds).
- Highly developed standardized profiles agreed upon by multi-vendor technical working groups for dependable interoperability of devices from different manufacturers.
- Ability to control, for example, 70 axes of motion at an update of 250 microseconds for each and every drive (SERCOS-III).

Support

The SERCOS interface is supported globally by SERCOS International e.V. (SI) in Germany. Regional support is provided by SERCOS North America(USA), SERCOS China and SERCOS Japan. These organizations provide a forum for the continued development of the standard, as well as user support.

Conformance Testing and Interoperability

An important aspect of an open, interoperable communications system is rigorous testing of products for adherence to the standard and their ability to operate in networks of products from multiple vendors. SERCOS International e.V. supports a Conformance Laboratory at the University of Stuttgart's Institute for Control Engineering of Machine Tools and Manufacturing Units (ISW). Products successfully passing conformance testing may display a mark indicating they are conformance tested, and are publicized in an index of certified devices.

Chapter-9

PROFINET

PROFINET is the open industrial Ethernet standard of PROFIBUS & PROFINET International (PI) for automation. PROFINET uses TCP/IP and IT standards, and is, in effect, real-time Ethernet.

The PROFINET concept features a modular structure so that users can select the cascading functions themselves. They differ essentially because of the type of data exchange to fulfil the partly very high requirements of speed.

In conjunction with PROFINET, the two perspectives PROFINET CBA and PROFINET IO exist. PROFINET CBA is suitable for the component-based communication via TCP/IP and the real-time communication for real-time requirements in modular systems engineering. Both communication options can be used in parallel.

PROFINET IO was developed for real time (RT) and isochronous real time (IRT) communication with the decentral periphery. The designations RT and IRT merely describe the real-time properties for the communication within PROFINET IO.

PROFINET CBA and PROFINET IO can communicate at the same time on the same bus system. They can be operated separately or combined so that a PROFINET IO subsystem appears as a PROFINET CBA system from a system perspective.

Technology

To achieve these functions, three different protocol levels are defined:

- TCP/IP for PROFINET CBA and the commissioning of a plant with reaction times in the range of 100ms
- RT (Real-Time) protocol for PROFINET CBA and PROFINET IO applications up to 10 ms cycle times
- IRT (Isochronous Real-Time) for PROFINET IO applications in drive systems with cycles times of less than 1ms

The PROFINET protocol can be recorded and displayed using any Ethernet analysis tool. In the current versions, Wireshark/Ethereal are able to decode PROFINET message frames.

PROFINET component model (PROFINET CBA)

A PROFINET CBA system consists of various automation components. One component covers all mechanical, electrical and IT variables. The component can be generated using the standard programming tools. A component is described using a PROFINET Component Description (PCD) file in XML. A planning tool loads these descriptions and enables the logical interconnections between the individual components to be generated for implementing a plant.

This model was largely inspired by the IEC 61499 standard.

The basic idea of CBA is that an entire automation system can in many cases be divided into autonomously operating subsystems, thereby arranging them very clearly. The design and the functions can actually end up in identical or slightly modified form in several systems. These PROFINET components are usually controlled by manageable number of input signals. Within the component, a control program written by the user executes the required function within the component and passes the corresponding output signals to another controller. The engineering that is associated with it is manufacturer-neutral. The communication of a component-based system is only configured, instead of being programmed. The communication with PROFINET CBA (without real time) is suitable for bus cycle times of approx. 50 ... 100 ms. The parallel running RT channel allows for data cycles similar to PROFINET IO (a few ms).

PROFINET and the peripherals (PROFINET IO)

Interfacing the peripherals is implemented by PROFINET IO. It defines the communication with field connected peripheral devices. Its basis is a cascading real-time concept. PROFINET IO defines the entire data exchange between controllers (devices with "master functionality") and the devices (devices with "slave functionality"), as well as parameter setting and diagnosis. PROFINET IO is designed for the fast data exchange between Ethernet-based field devices and follows the provider-consumer model. Field devices in a subordinate PROFIBUS line can be integrated in the PROFINET IO system without any effort and seamlessly via an IO-Proxy (representative of a subordinate bus system). A device developer can implement PROFINET IO with any commercially available Ethernet controller. It is well-suited for the data exchange with bus cycle times of a few ms. The configuration of an IO-System has been kept nearly identical to the "look and feel" of PROFIBUS. PROFINET IO always contains the real-time concept.

A PROFINET IO system consists of the following devices:

- The IO Controller, which controls the automation task.

- The IO Device, which is a field device, monitored and controlled by an IO Controller. An IO Device may consist of several modules and sub-modules.
- The IO Supervisor is software typically based on a PC for setting parameters and diagnosing individual IO Devices.

An Application Relation (AR) is established between an IO Controller and an IO Device. These ARs are used to define Communication Relations (CR) with different characteristics for the transfer of parameters, cyclic exchange of data and handling of alarms.

The characteristics of an IO Device are described by the device manufacturer in a General Station Description (GSD) file. The language used for this purpose is the GSDML (GSD Markup Language) - an XML based language. The GSD file provides the supervision software with a basis for planning the configuration of a PROFINET IO system.

PROFINET and real time

Within PROFINET IO, process data and alarms are always transmitted in real time (RT). Real time in PROFINET is based on the definition of IEEE and IEC, which allow for only a limited time for execution of real-time services within a bus cycle. The RT communication represents the basis for the data exchange for PROFINET IO. Real-time data are treated with a higher priority than TCP(UDP)/IP data. RT provides the basis for the real-time communication in the area of distributed periphery and for the PROFINET component model (PROFINET CBA). This type of data exchange allows bus cycle times in the range of a few hundred microseconds.

PROFINET and isochronous communication

The isochronous data exchange with PROFINET is defined in the isochronous real-time (IRT) concept. PROFINET IO field devices with IRT functionality have switch ports integrated in the field device. They can be based e.g. on the Ethernet controllers ERTEC 400/200. The data exchange cycles are usually in the range of a few hundred microseconds up to a few milliseconds. The difference to real-time communication is essentially the high degree of determinism, so that the start of a bus cycle is maintained with high precision. The start of a bus cycle can deviate up to 1 µs (jitter). IRT is required, for example, for motion control applications (positioning control processes).

Profiles

Profiles are pre-defined configurations of the functions and features available from PROFINET for use in specific devices or applications. They are specified by PI working groups and published by PI. Profiles are important for openness, interoperability and interchangeability, so that the end user can be sure that similar equipments from different vendors perform in a standardised way. User choice encourages competition that drives vendors towards enhanced performance and lower costs.

There are PROFINET profiles for Encoders, for example. Other profiles have been specified for Motion Control (PROFIdrive) and Functional Safety (PROFIsafe). A special profile for Trains also exists.

An important profile is PROFIenergy. This was requested in 2009 by the AIDA group of German automotive Manufacturers (Audi, BMW, Mercedes, Porsche and VW) who wished to have a standardised way of actively managing energy usage in their plants. High energy devices and sub-systems such as robots, lasers and even paint lines are the target for this profile, which will help reduce a plant's energy costs by intelligently switching the devices into 'sleep' modes to take account of production breaks, both foreseen (e.g. weekends and shut-downs) and unforeseen (e.g. breakdowns).

PROFIenergy is applicable across industry and includes monitoring services that can lead to the real time management of energy demand

Additional highlights of the PROFINET concept

Engineering: By supporting the Tool Calling Interface, every manufacturer of field devices can latch onto any TCI-capable software and parameterize and diagnose field devices without having to leave the program.

The proximity recognition and device replacement: All PROFINET field devices determine their neighbours. This allows replacing field devices without additional tools and prior knowledge in case of a fault. By reading this information, the system topology can be graphically represented in a very clear way.

Parameter server: Individually set data can be loaded manufacturer-neutral (e.g. via TCI) and automatically archived in a parameter server. Reloading is also done automatically when replacing a device.

Determinism: PROFINET supports the deterministic data traffic, for example, for high-precision control tasks.

Redundancy: The redundancy concept defined in PROFINET significantly increases system availability.

Benefits of PROFINET

Due to the continuous further development of PROFINET, users are provided with a long-term perspective for the implementation of their automation tasks.

The system operator profits from the simple expandability of the system and high degree of availability from autonomously running subsystems.

Chapter-10

Ethernet Powerlink

Ethernet POWERLINK is a deterministic real-time protocol for standard Ethernet. It is an open protocol managed by the Ethernet POWERLINK Standardization Group (EPSG). It was introduced by Austrian automation company B&R in 2001.

This protocol has nothing to do with power distribution via Ethernet cabling or power over Ethernet (PoE), power line communication or Bang & Olufsens PowerLink cable.

Overview

Ethernet POWERLINK was created taking care of standard-compliance. It expands Ethernet with a mixed Polling- and Timeslicing mechanism. That brings:

- Guaranteed transfer of time-critical data in very short isochronic cycles with configurable response time
- Time-synchronisation of all nodes in the network with very high precision of sub-microseconds
- Transmission of less timecritical data in a reserved asynchronous channel

Modern implementations reach cycle-times of under 200 µs and a time-precision (jitter) of less than 1 µs.

Standardization

POWERLINK is standardized by the opened user- and producer-group EPSG (Ethernet POWERLINK Standardization Group) as a public standard. EPSG was founded in June 2003 as an independent association. Its focus is to leverage the advantages of Ethernet for high performance Real-Time networking systems based on the Ethernet POWERLINK Real-Time protocol, introduced by B&R end of 2001. Various working groups are focusing on different tasks like safety, technology, marketing, certification and end users.

The EPSG is cooperating with leading standardization bodies and associations, like the CAN in Automation (CiA) Group and the IEC.

Physical layer

The original physical layer specified was 100Base-X Fast Ethernet (IEEE 802.3). Since end of 2006 Ethernet Powerlink with Gigabit Ethernet supporting a transmission rate ten times higher (1,000 Mbit/s) is in use. That offers a better performance for the future to extended systems with major production performance, a series of module control systems, numerous drives and completely integrated security equipment. Gigabit Ethernet is on the threshold of general dissemination in IT systems. There is no need for major change in system designs, components or cabling. Networked units that master this high transmission rate and a somewhat better cable (Cat6) must be used. The transition to higher speed Ethernet variants is always possible since Ethernet Powerlink is also attached as a standard Ethernet design and can launch its applications with standard modules such as microcomputers and FPGA modules. Usage of repeating hubs instead of switches within the Real-time domain is recommended to minimise delay and jitter. Ethernet Powerlink uses IAONA's Industrial Ethernet Planning and Installation Guide for clean cabling of industrial networks and both industrial Ethernet connectors 8P8C (RJ45) and M1 are accepted.

Data Link Layer

The Standard Ethernet Data Link Layer of Ethernet Powerlink is extended by an additional bus scheduling mechanism which secures that at a time only one node is accessing the network. The schedule is divided into an isochronous phase and an asynchronous phase. During the isochronous phase, time-critical data is transferred, while the asynchronous phase provides bandwidth for the transmission of non time-critical data. The Managing Node (MN) grants access to the physical medium via dedicated poll request messages. As a result, only one single node (CN) has access to the network at a time, which avoids collisions, usually present on Standard Ethernet. The CSMA/CD mechanism of standard Ethernet, which causes non-deterministic Ethernet behaviour, is deactivated by the collision avoidance mechanism of the Ethernet Powerlink scheduling mechanism.

Basic Cycle

After system start-up is finished, the Real-Time domain is operating under Real-Time conditions. The scheduling of the basic cycle is controlled by the Managing Node (MN). The overall cycle time depends on the amount of isochronous data, asynchronous data and the number of nodes to be polled during each cycle.

The basic cycle consists of the following phases:

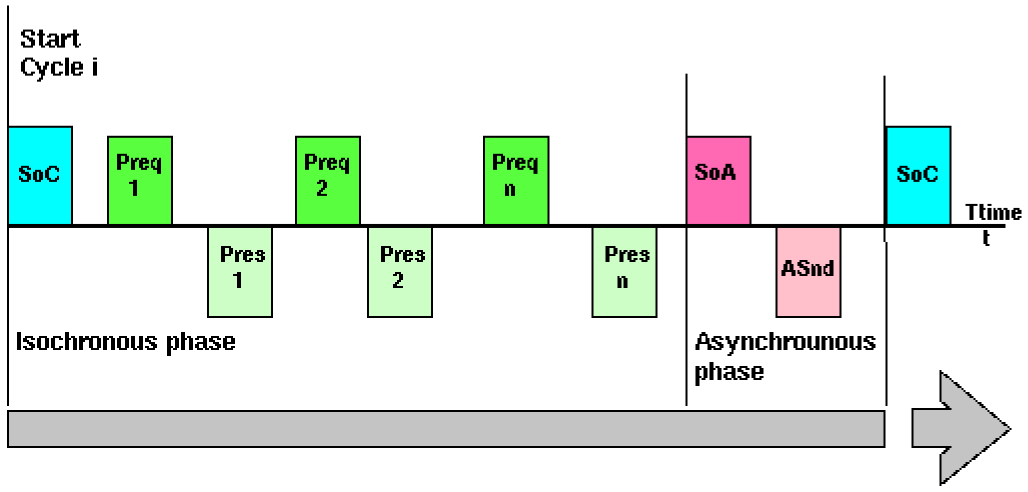
- **Start Phase**: The Managing Node is sending out a synchronization message to all nodes. The frame is called SoC - Start of Cycle.

- **Isochronous Phase**: The Managing Node calls each node to transfer time-critical data for process or motion control by sending the Preq - Poll Request - frame. The addressed node answers with the Pres - Poll Response - frame. Since all other nodes are listening to all data during this phase, the communication system provides a producer-consumer relationship.

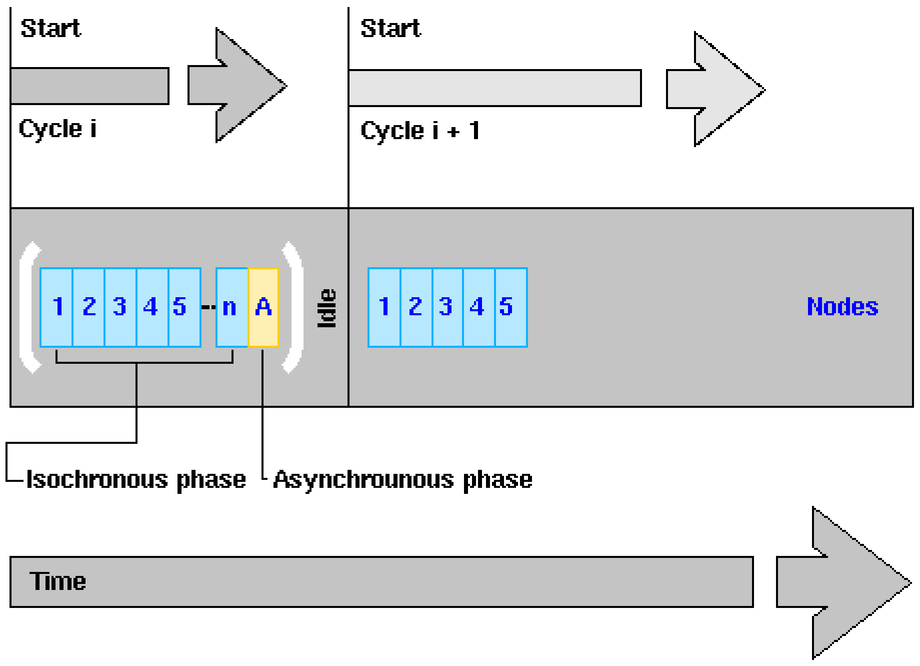
The time frame which includes Preq-n and Pres-n is called time slot for the addressed node.

- **Asynchronous Phase**: The Managing Node grants the right to one particular node for sending ad-hoc data by sending out the SoA - Start of Asynchronous - frame. The addressed node will answer with ASnd. Standard IP-based protocols and addressing can be used during this phase.

The quality of the Real-Time behaviour depends on the precision of the overall basic cycle time. The length of individual phases can vary as long as the total of all phases remain within the basic cycle time boundaries. Adherence to the basic cycle time is monitored by the Managing Node. The duration of the isochronous and the asynchronous phase can be configured.



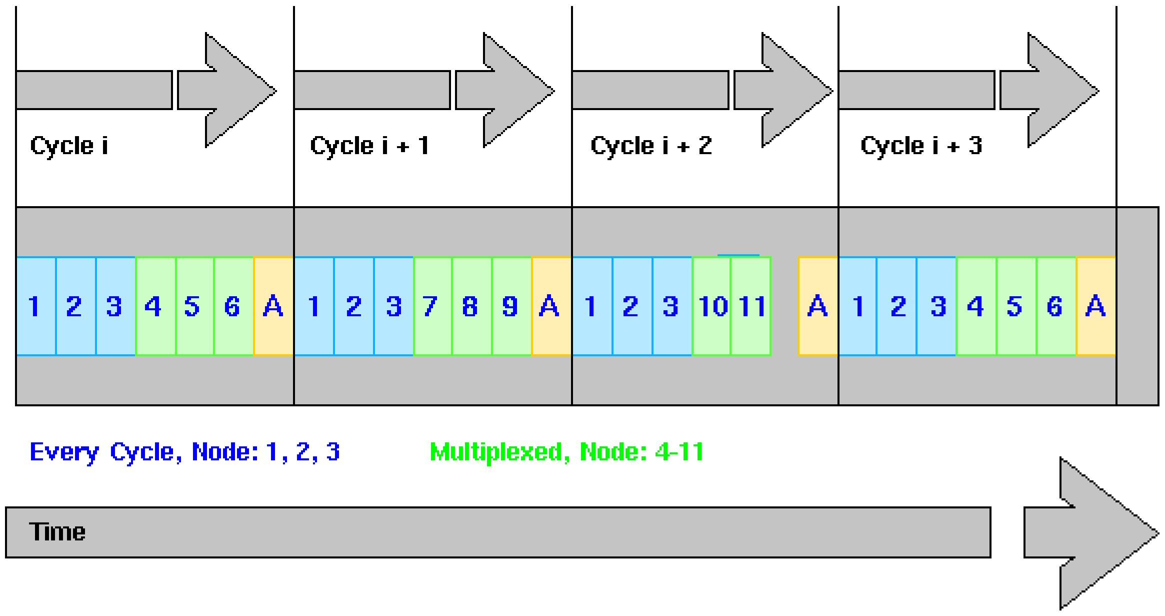
Picture 1: Frames above the time line are sent by the MN, below the time line by different CNs.



Picture 2: Time slots for nodes and the asynchronous time slot

Multiplex for Bandwidth Optimization

In addition to transferring isochronous data during each basic cycle, some nodes are also able to share transfer slots for better bandwidth utilization. For that reason, the isochronous phase can distinguish between transfer slots dedicated to particular nodes, which have to send their data in every basic cycle, and slots shared by nodes to transfer their data one after the other in different cycles. Therefore less important yet still time-critical data can be transferred in longer cycles than the basic cycle. Assigning the slots during each cycle is at the discretion of the Managing Node.


Cycle i
Cycle i + 1
Cycle i + 2
Cycle i + 3
1 2 3 4 5 6 A 1 2 3 7 8 9 A 1 2 3 10 11 A 1 2 3 4 5 6 A
Every Cycle, Node: 1, 2, 3
Multiplexed, Node: 4-11
Time

Picture 3: Time slots in EPL multiplexed mode.

Ethernet Powerlink Safety

Today, machines, plants and safety systems are stuck in a rigid scheme made up of hardware-based safety functions. The consequences of this are cost-intensive cabling and limited diagnostic options. The solution is the integration of safety relevant application data into the standard serial control protocol. Ethernet Powerlink Safety (EPLsafety). EPLsafety allows both publish/subscriber and client/server communication. Safety relevant data is transmitted via an embedded data frame inside of standard communication messages. Measures to avoid any undetected failures due to systematic or stochastic errors are an integral part of the security protocol. EPLsafety is in conformance with IEC 61508. The protocol fulfills the requirements of SIL 3. Error detection techniques have no impact on existing transport layers.

Chapter-11

SERCOS III

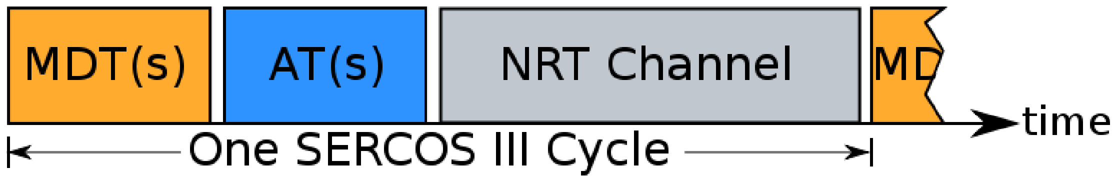
SERCOS III is the third generation of the SERCOS interface, a globally standardized open digital interface for the communication between industrial controls, motion devices, and input/output devices (I/O). SERCOS III merges the hard real-time aspects of the SERCOS interface with Ethernet. It is based upon and conforms to the Ethernet standard (IEEE 802.3 & ISO/IEC 8802-3). Work began on SERCOS III in 2003, with vendors releasing first products supporting it in 2005. In addition to the standard SERCOS features cited under the SERCOS interface general description, SERCOS III also provides:

- Cyclic updates to devices at rates as low as 31.25 µsec
- Support for up to 511 Slave devices on one network
- Redundancy: Bump-less physical layer single-fault recovery
- Detection of a dropped physical connection within 25 µsec (less than one cycle update)
- Hot plugging: insertion & configuration of devices into network while cyclic communication is active

General architecture

In order to achieve the throughput and jitter requirements required in the applications the interface is designed for, SERCOS III operates primarily in a Master/Slave arrangement exchanging cyclic data between nodes. The Master initiates all data transmission during a SERCOS real-time cycle. All data transmissions begin and end at the Master (circular).

SERCOS III cycle

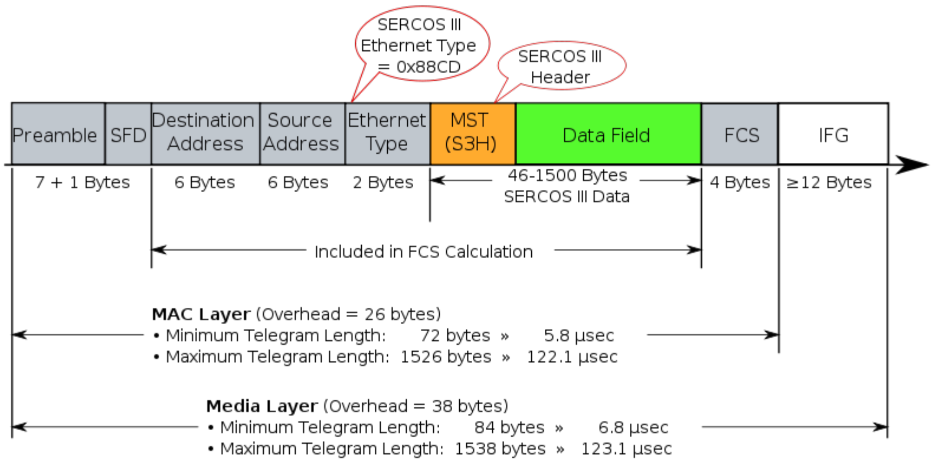


The basic SERCOS III cycle

Communication across a SERCOS III network occurs in strict cyclic intervals. A cycle time is chosen by the user for a given application, ranging from 31.25 µsec. to 65 msecs. Within each cycle, data is exchanged between SERCOS III nodes using two types of telegrams: MDTs and ATs. After all MDTs and ATs are transmitted, SERCOS III nodes allow the remaining time in the cycle to be used as an NRT (Non real time) Channel, which can be used to exchange data using other formats, such as IP.

The network remains available to NRT traffic until the next cycle begins, at which time the SERCOS III nodes close the nodes to NRT traffic again. This is an important distinction. SERCOS is purposely designed to provide open access at all ports for other protocols between cyclic real time messages. No tunneling is required. This provides the advantage that any SERCOS III node is available, whether SERCOS III is in cyclic mode or not, to use other protocols, such as TCP/IP, without any additional hardware to process tunneling. SERCOS nodes are specified to provide a store and forward method of buffering non-SERCOS messages should they be received at a node while cyclic communication is active.

Telegrams


SERCOS III Ethernet Type = 0x88CD
SERCOS III Header
Preamble
SFD
Destination Address
Source Address
Ethernet Type
MST (S3H)
Data Field
FCS
IFG
7 + 1 Bytes
6 Bytes
6 Bytes
2 Bytes
46-1500 Bytes SERCOS III Data
4 Bytes
≥12 Bytes
Included in FCS Calculation
MAC Layer (Overhead = 26 bytes)
• Minimum Telegram Length: 72 bytes » 5.8 µsec
• Maximum Telegram Length: 1526 bytes » 122.1 µsec
Media Layer (Overhead = 38 bytes)
• Minimum Telegram Length: 84 bytes » 6.8 µsec
• Maximum Telegram Length: 1538 bytes » 123.1 µsec

SERCOS III Telegram Structure

Telegram format

All SERCOS III telegrams conform to the IEEE 802.3 & ISO/IEC 8802-3 MAC (Media Access Control) frame format.

Destination address
: The destination address for all SERCOS III telegrams is always 0xFFFF FFFF FFFF (all 1s), which is defined as a broadcast address for Ethernet telegrams. This is because all telegrams are issued by the Master, and are intended for all Slaves on the network.

Source address
: The source address for all SERCOS III telegrams is the MAC address of the Master, as it issues all telegrams.

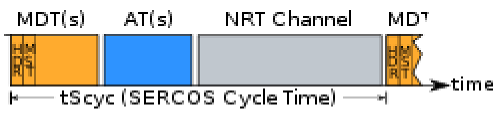
Ethernet type
: A unique EtherType value has been assigned via the IEEE EtherType Field Registration Authority for SERCOS III (0x88CD).

SERCOS III header
: The beginning of the Ethernet-defined data field always begins with a SERCOS III header, which contains control and status information unique to SERCOS.

SERCOS III data field
: The SERCOS III header is followed by the SERCOS III data field, which contains a configurable set of variables defined for each device in the network.

Telegram types

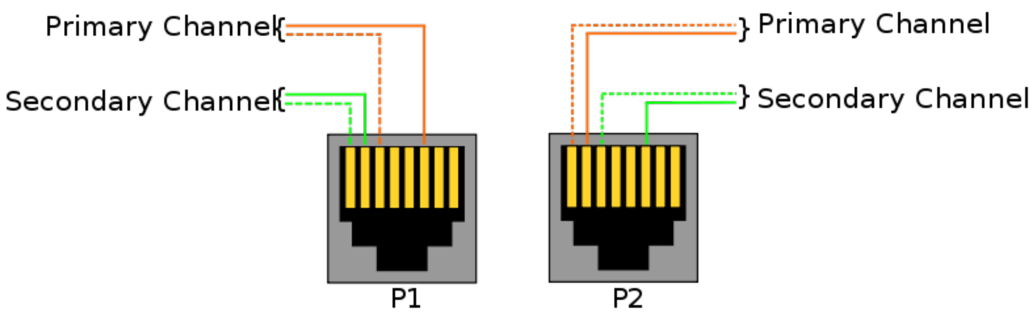
Two main types of telegrams are used within the SERCOS III Cycle. The Master Data Telegram (MDT), and the Acknowledge telegram (AT). Both telegram types are issued by the Master (control). The MDT contains information provided by the Master to Slaves. It is filled by the Master, and read by Slaves. The AT is issued by the Master, but actually populated by each Slave with their appropriate response data (feedback values, input states, etc.). More than one Slave uses the same AT, filling in its pre-determined area in the AT telegram, updating checksums, and then passing the telegram to the next device. This method reduces the impact of the Ethernet frame overhead on the performance of the network without compromising IEEE 802.3 & ISO/IEC 8802-3. The amount of data sent from the Master to Slaves, as well as the sum of the data returned by the Slaves, may exceed the 802.3-specified maximum 1500-byte data field size. To comply with this limit, SERCOS III may use more than one MDT telegram in a cycle, as well as more than one AT telegram (up to 4 in each case).



SERCOS III Synchronization

Synchronization

To achieve true hard real time characteristics, SERCOS III, like SERCOS I & II, uses a form of synchronization that depends upon a synchronization "mark" issued by the Master control at exact equidistant time intervals. All nodes in a SERCOS Network use this telegram to synchronize all activities in the node. To account for variations in network components, delays are measured in the node-to-node transmissions during phase-up (initialization) of a SERCOS network, and those values compensated for during normal operation. Unlike SERCOS I & II, where a separate Master Sync Telegram, or MST is used for this purpose, SERCOS III includes the MST in the first MDT transmitted. No separate telegram is issued. The time between two MSTs is exactly equal to the designated SERCOS Cycle Time, tScyc.



SERCOS III Physical Interface Nomenclature

Physical and data link layers

SERCOS III supports standard IEEE 802.3 & ISO/IEC 8802-3 100Base-TX or 100Base-FX (100 Mb/s baseband) Full Duplex physical layer (PHY) entities. 802.3-compliant Media-Access Controller (MAC) sub-layers are used. Autonegotiation must be enabled on each PHY, but only 100Mbit full duplex is supported. Auto (MAU [Media Attachment Unit]-Embedded) Crossover is specified between the two Physical Medium Attachment (PMA) units present with a duplex port. These two units are referred to as the Primary Channel and Secondary Channel in the SERCOS III specification. Dual interfaces are required (two duplex interfaces per device). Within the SERCOS III specification the dual interfaces are referred to as P1 and P2 (Ports 1 and 2).

SERCOS III stack

All of the functionality required to configure a SERCOS III interface is contained in a stack that is available in both "hard" and "soft" versions. The hard version is widely used for embedded applications (such as drives, I/O modules and micro-controller based motion control), where:

  - It is important that the overhead of managing the SERCOS III nodes not be placed upon the device processor.
  - Nanosecond jitter is required.

The hardware stack is available in a number of different forms. These currently include:

-
  - A bit stream for Xilinx FPGAs
  - A bit stream for Altera FPGAs
  - A Net list for Xilinx FPGAs
  - A Net list for Altera FPGAs
  - The "netX" multi-network controller chip from Hilscher, GmbH

The maximum jitter allowed with hard-stack-based Masters and Slaves is smaller 1 μsec. Using the above stacks yields a jitter similar to SERCOS II (35-70 nanoseconds).

SERCOS III also supports a "Soft Master", using a completely software-based stack for the master interface.\ Since the maximum jitter in such a configuration is dependent upon the operating system of the Master, the maximum jitter may be set by a variable for the SERCOS III network when a Soft Master is employed.

For basic Slaves, such as I/O devices, a license-free core is available. The Easy-I/O core can be downloaded and loaded onto Xilinx Spartan-3 FPGA devices.

Data consistency

A term usually associated with the IT enterprise, data consistency can also apply to real-time control. For this reason, SERCOS III specifies that no data be overwritten (destroyed) during a transmission. Every slave on a network may access input and output data for every other slave on the network.

Addressing

Devices must support Ethernet's MAC addressing, plus the SERCOS III addressing. Other addressing schemes are optional.

SERCOS III address

Each SERCOS III device contains a numeric address used by other devices on the SERCOS III network to exchange data. The address may be any whole integer from 1 to 511.

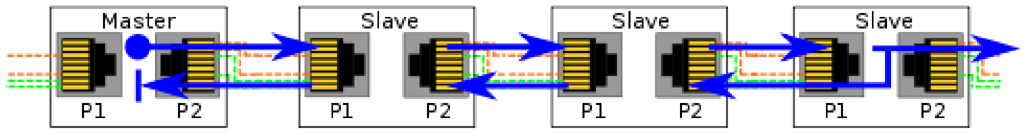
IP address

SERCOS III does not use an IP address for its own operation. Whether a device contains an IP address or not is dependent on its support of other specifications, either independent (exclusive) of SERCOS III operation, or via the NRT portion of the cycle.

Network topologies

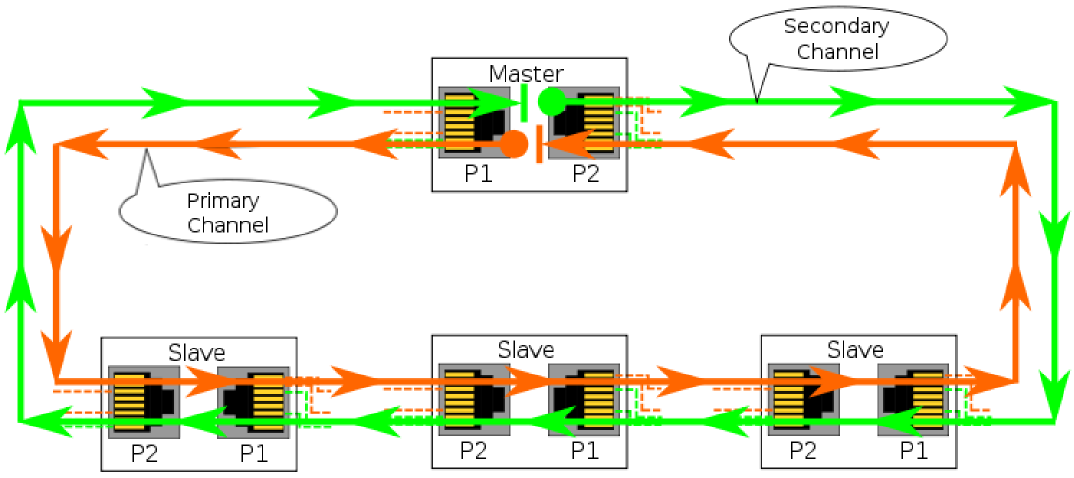
The SERCOS III specification defines two possible network topologies; Ring and Line. To those familiar with other networks, they may appear to both be configured as a Ring. All telegrams begin and end at the Master. The Full Duplex feature of the physical layer is used to achieve this.

Line topology



SERCOS III Line Topology

A line topology is the simpler of the two possible arrangements, and provides no redundancy. However, this configuration saves the cost of one cable. In it, only one of the two interfaces on the Master is used. Telegrams are issued out of the transmit PMA on the Master's active port. Either port on the Master may be the active one. SERCOS III determines this during phase-up (initialization). The first Slave receives the telegrams on the connected interface's receive PMA, modifies them as required, and issues them out on the transmit PMA of the second interface. Each cascading Slave does likewise until the last Slave in the Line is reached. That Slave, detecting no SERCOS III connection on its second port, folds the telegram back on the receiving interface's transmit port. The telegram then makes it way through each Slave back to the Master. Note the last Slave also emits all SERCOS III telegrams on its second port, even though no SERCOS III connection is detected. This is for snooping, ring closures, as well as hot-plugging. Keep in mind that since the Ethernet destination field in all SERCOS III telegrams is the broadcast address of 0xFFFF FFFF FFFF (all 1s), all telegrams issued from this open port will be seen by other devices as broadcast telegrams. This behavior is by design, and cannot be disabled. To avoid taxing networks attached to an open SERCOS port, an NRT-Plug can be used, or alternately a managed Ethernet switch programmed to block broadcast telegrams received from the SERCOS port can be used.



SERCOS III Ring Topology

Ring topology

A ring topology simply closes the network by attaching the unused port on the last device in a ring back to the unused port on the Master. When the SERCOS III Master senses that a ring exists, it sets up two counter-rotating telegrams. The same data is issued simultaneously out of the transmit PMAs of both ports on the Master. From there both telegrams are managed essentially identically as they make their way through each Slave, ending back at the opposite port on the Master they were emitted from. Advantages to this topology include tighter synchronization, as well as automatic infrastructure redundancy.

Other network topologies

With both the line or ring structure, SERCOS III operates in a “circular” approach. All telegrams leave the Master, and return there. As with any network that operates in this manner, modified structures can be constructed to appear as a tree or star network, utilizing hardware that manages the branches, but the structure is still circular in nature.

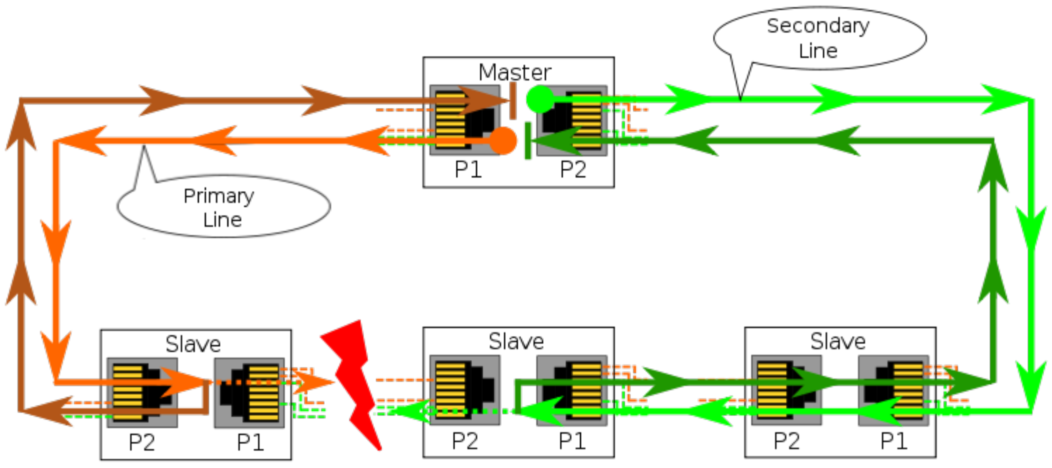
Infrastructure hardware

SERCOS III is designed in such a way that no additional network infrastructure (standard Ethernet switches, Hubs, etc.) is required to operate. In fact, no additional standard Ethernet (non-SERCOS III capable) components may be placed within a SERCOS III network, as their presence will adversely affect the timing and synchronization of the network.

Features

Application layer (profiles)

The SERCOS III Specification defines a broad range of variables developed by a consortium of product suppliers to provide interoperability between components (motion controls, drives, etc.). All Traffic across a SERCOS III network consists of Idents (parameters) with attributes. This method was first defined in SERCOS I, as an essentially flat set of Idents. They were later grouped into application sets to aid in selection of pertinent Idents required for a given industry, such as the “Pack Profile” for use with packaging machinery. During the development of the SERCOS III specification, this methodology was further refined to group the Idents logically by device class. The definition of the legacy Idents has remained largely untouched; rather their grouping has been re-evaluated for a more understandable architecture. This has also enabled the separation of communication Idents into a logical subset, simplifying migration from SERCOS I/II to SERCOS III, and providing a clear overview to users.



SERCOS III Redundancy healing a Ring Break

Redundancy

When a ring network is employed, SERCOS III provides for automatic infrastructure redundancy. If any interconnection point in the ring ceases to function, the associated SERCOS III nodes will detect a “ring break” and “loop back” the end nodes, effectively operating as two lines rather than one ring.

The operation is “bump-less”, as the detection & recovery time to such a break is less than 25 µsecs, which is less than the minimum SERCOS III cycle time. SERCOS III can also recover from ring breaks and “heal” with no interruption in operation. Since SERCOS III telegrams continue to be emitted by transmit PMAs on unconnected ports, and receive PMAs on unconnected ports continue to monitor for incoming data, when a

SERCOS III port recognizes that a ring has by physically re-closed, it will re-activate the counter-rotating telegrams to functionally close the rings again. This operation is also bump-less.

Peer communications

To ensure the determinism required, most Real-time Ethernet standards enforce a master-to-slave-only method of communications. This can conflict with the need for a node in the system to exchange data efficiently with a node other than the network master. The conventional method to achieve this in a master-slave network is to pass data from one slave node to the master, where it is reissued to one or more different slaves. For example, if several servo drives on a network are to be synchronized to a signal from another drive on the network, the master must fetch the signal from this drive and reissue it to all other drives on the network. Disadvantages to this method are that delays are induced due to the multiple cycles required, and the master's processing load is increased as it must actively participate in the function, although it contributes nothing. Since no data is destroyed in a SERCOS III telegram, data to and from any slave can be accessed by another node on the network without any additional cycle delay or master intervention. Additionally, as telegrams pass each node twice in a cycle (for both topology types), a node can even have the opportunity to access data supplied by a subsequent node. Two peer communication methods are defined in the SERCOS III specification: Controller to Controller (C2C) for multiple masters to communicate with one another, and Cross Communication (CC) for multiple slaves.

Hot-plugging

Another feature of SERCOS III is hot-plugging, which is the ability to add devices to an active network. Using the features described for redundancy, a network can detect when a new device is attached to an active network. Processes exist that configure the new device, and announce it's availability to the Master control. After that, the Master control can select to make use of the new device based on the application currently running.

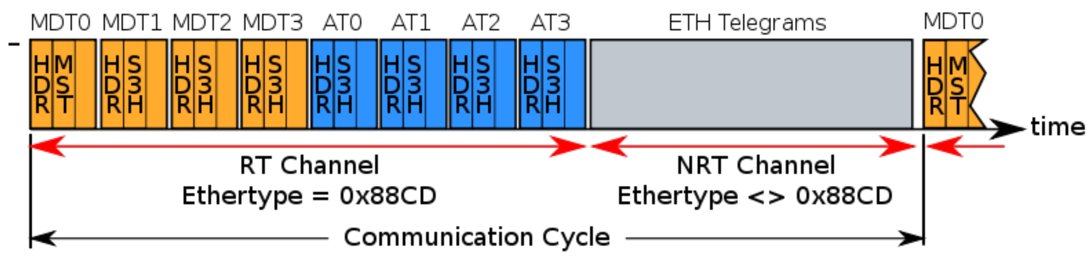
Non-real-time (NRT) channel

The time between the end of the transmission of all SERCOS III Real Time (RT) cyclic telegrams, and the beginning of the next communication cycle is defined as the "SERCOS III Non Real Time Channel" (NRT Channel). During this time period, the SERCOS Network is opened to allow transmission of Ethernet-compliant frames for other services and protocols. For example:

1. Web servers can be embedded in SERCOS III-compliant devices to respond to standard Hypertext Transfer Protocol (HTTP) messages received via the NRT Channel.
2. Frames from other Fieldbus standards that conform to Ethernet frame formatting may be transmitted across a SERCOS III network.

Every SERCOS III-compliant node must support the passing of NRT frames through its SERCOS III interface. Whether a SERCOS III node actively makes use of the NRT feature is determined by the feature set of the product. If, for example, the device has an embedded web server, it could make available its IP address for access by other devices.

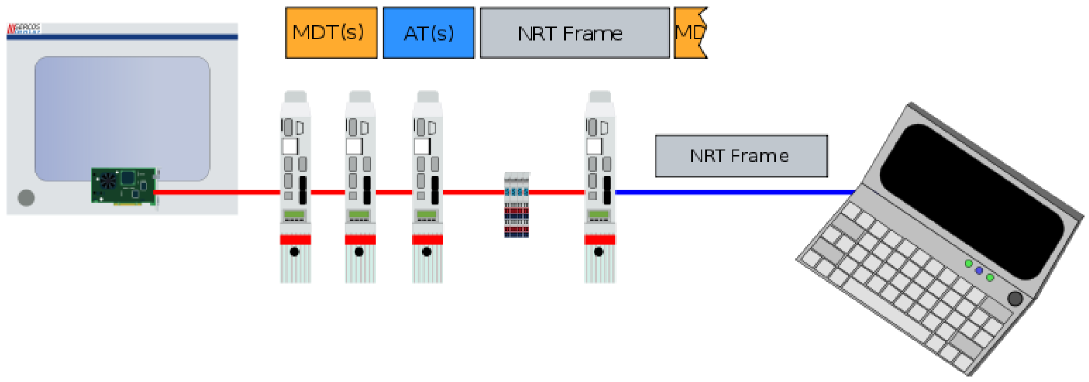
A SERCOS III network will always pass NRT frames, even when cyclic operation has not been initialized. This means that devices always have access to the network for NRT messages, as long as the ports are powered.



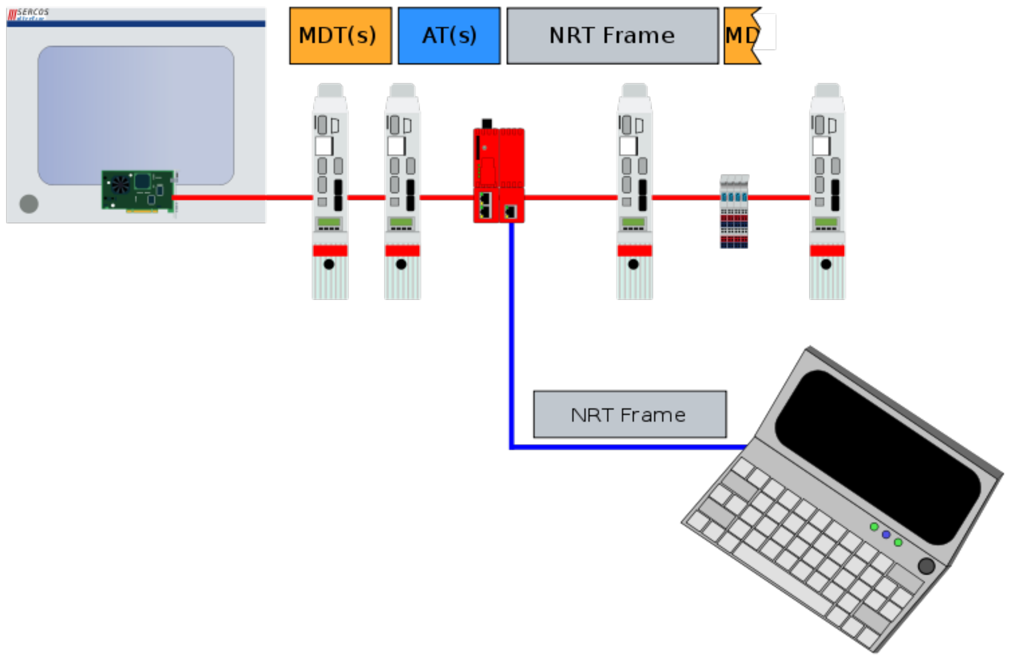
NRT Channel

SERCOS III does not define whether a port should operate in cut-through switching or store-and-forward mode when handling NRT frames. There are SERCOS III products currently on the market that support both modes. Likewise, SERCOS III does not define whether a port should intelligently process NRT telegrams, such as learn the network topology.

The time allotted for NRT is dictated by the amount of data transmitted during the RT portion of the cycle. In real-world applications, there is a significant amount of bandwidth available for NRT frames. For example, in a typical application with 8 axes of motion and a cycle rate of 250 microseconds, the equivalent of 85 Mbps is available for NRT use. This amount of time means the NRT frames in this example can be as long as the maximum defined for Ethernet (Maximum Transmission Unit [MTU] =1500). Using the same example of 8 axes, but with a cycle time of 62.5 microseconds, the effective bandwidth available for NRT frames would be 40 Mbps, and the MTU would be reduced to 325. As with any network where time on the bus is shared, MTU values should be configured to ensure reliable communication. Properly configured SERCOS networks will set the SERCOS parameter "Requested MTU" (S-0-1027.0.1) to the recommended MTU value, which can then be read by other devices to match their MTU settings. Regardless of the value of this parameter, a SERCOS node will allow non-SERCOS traffic to pass for the entire NRT channel time period (i.e., telegrams longer than the MTU setting are not discarded by the SERCOS stack). SERCOS parameter S-0-1027.0.1 is set by default to 576, the minimum value called out in RFC 791.



NRT Access via open port



NRT Access via an NRT-Plug

NRT access

NRT frames may only enter a SERCOS III network through a SERCOS III-compliant port. This can be achieved two different ways. One is to employ the unused SERCOS III port at the end of a SERCOS III network configured in line topology, as shown to the right.

In a network configured in ring topology, the ring can be temporarily broken at any point to also attach a device. Since the redundancy feature of SERCOS III will reconfigure the network in a bump-less manner (responding in less than one cycle), no disruption of network transmission will occur. The ring can again be closed after the access is no longer required.

If access is desired in the middle of a line topology (where no free ports are available), or it is undesirable to break a ring topology for extended periods of time, the SERCOS III specification permits a device called an “NRT-plug” that can be used to provide access to the NRT channel anywhere along the network. NRT-plugs supply two SERCOS III-compliant ports, and one or more ports for NRT access.

Commercially available NRT Switches block the transmission of SERCOS III broadcast telegrams out their non-SERCOS III port(s), to prevent flooding of non-SERCOS III networks with SERCOS III cyclic data.

Functional safety support

"Functional safety" is a general term referring to the design of a system that reduces the risk that a hazardous event harmful to humans can occur with a system. The main definition is contained in the international standard IEC 61508. Most industrial networks contain some type of features to conform to functional safety requirements. Rather than define a unique specification for this functional safety, SERCOS III Safety is based upon the CIP Safety safety protocol developed by the Open DeviceNet Vendors Association (ODVA). This provides interoperability at the safety level with all networks based upon the Common Industry Protocol (CIP), including DeviceNet and EtherNet/IP.

Chapter-12

Modbus

Modbus is a serial communications protocol published by Modicon in 1979 for use with its programmable logic controllers (PLCs). Simple and robust, it has since become one of the de facto standard communications protocols in the industry, and it is now amongst the most commonly available means of connecting industrial electronic devices. The main reasons for the extensive use of Modbus in the industrial environment are:

1. It has been developed with industrial applications in mind
2. It is openly published and royalty-free
3. It is easy to deploy and maintain
4. It moves raw bits or words without placing many restrictions on vendors

Modbus allows for communication between many (approximately 240) devices connected to the same network, for example a system that measures temperature and humidity and communicates the results to a computer. Modbus is often used to connect a supervisory computer with a remote terminal unit (RTU) in supervisory control and data acquisition (SCADA) systems. Many of the data types are named from its use in driving relays: a single-bit physical output is called a *coil*, and a single-bit physical input is called a *discrete input* or a *contact*.

The development and update of Modbus protocols are managed by the Modbus Organization, formed of independent users and suppliers of Modbus compliant devices. Some of its prominent members are Precision Digital Corporation, Motor Protection Electronics and FieldServer Technologies. Other companies, such as SoftDEL Systems and SATEC Ltd., offer Modbus devices without being formal members of Modbus Organization.

Protocol versions

Versions of the Modbus protocol exist for serial port and for Ethernet and other networks that support the Internet protocol suite. Most Modbus devices communicate over a serial EIA-485 physical layer . There are many variants of Modbus protocols

- *Modbus RTU* — This is used in serial communication & makes use of a compact, binary representation of the data for protocol communication. The RTU format follows the commands/data with a cyclic redundancy check checksum as an error check mechanism to ensure the reliability of data. Modbus RTU is the most common implementation available for Modbus. A Modbus RTU message must be transmitted continuously without inter-character hesitations. Modbus messages are framed (separated) by idle (silent) periods.

- *Modbus ASCII* — This is used in serial communication & makes use of ASCII characters for protocol communication. The ASCII format uses a longitudinal redundancy check checksum. Modbus ASCII messages are framed by leading colon (':') and trailing newline (CR/LF).

- *Modbus TCP/IP or Modbus TCP* — This is a Modbus variant used for communications over TCP/IP networks, connecting over port 502. It does not require a checksum calculation as lower layers already provide checksum protection.

- *Modbus over TCP/IP or Modbus over TCP or Modbus RTU/IP* — This is a Modbus variant that differs from Modbus TCP in that a checksum is included in the payload as with Modbus RTU.

- *Modbus over UDP* — Some have experimented with using Modbus over UDP on IP networks, which removes the overheads required for TCP

- *Modbus Plus (Modbus+, MB+ or MBP)* — An extended version, Modbus Plus (Modbus+ or MB+), also exists, but remains proprietary to SCHNEIDER ELECTRIC. It requires a dedicated co-processor to handle fast HDLC-like token rotation. It uses twisted pair at 1 Mbit/s and includes transformer isolation at each node, which makes it transition/edge triggered instead of voltage/level triggered. Special interfaces are required to connect Modbus Plus to a computer, typically a card made for the ISA (SA85), PCI or PCMCIA bus.

Data model and function calls are identical for the first 4 variants of protocols; only the encapsulation is different. However the variants are not interoperable as the frame formats are different.

Communication and devices

Each device intended to communicate using Modbus is given a unique address. In serial and MB+ networks only the node assigned as the Master may initiate a command, but on Ethernet, any device can send out a Modbus command, although usually only one master device does so. A Modbus command contains the Modbus address of the device it is intended for. Only the intended device will act on the command, even though other devices might receive it (an exception is specific broadcastable commands sent to node 0 which are acted on but not acknowledged). All Modbus commands contain checking information, ensuring that a command arrives undamaged. The basic Modbus commands can instruct an RTU to change a value in one of its registers, control or read an I/O port, as well as commanding the device to send back one or more values contained in its registers.

There are many modems and gateways that support Modbus, as it is a very simple protocol and often copied. Some of them were specifically designed for this protocol. Different implementations use wireline, wireless communication, such as in the ISM band, and even SMS or GPRS. One of the more common designs of wireless networks makes use of the mesh topology. Typical problems the designers have to overcome include high latency and timing problems.

Frame Format

All modbus variants choose different frame formats.

Modbus RTU Frame Format		
Name	**Length**	**Function**
Start	3.5c idle	*at least 3-1/2 character times of silence (MARK condition)*
Address	8 bits	*Station Address*
Function	8 bits	*Indicates the function codes like read coils / inputs*
Data	n * 8 bits	*Data + length will be filled depending on the message type*
CRC Check	16 bits	*Error checks*
End	3.5c idle	*at least 3-1/2 character times of silence between frames*

Modbus ASCII Frame Format

Name	Length	Function
Start	1 char	*starts with colon (:) (ASCII value is 3A hex)*
Address	2 chars	*Station Address*
Function	2 chars	*Indicates the function codes like read coils / inputs*
Data	n chars	*Data +length will be filled depending on the message type*
LRC Check	2 chars	*Error checks*
End	2 chars	*carriage return – line feed(CRLF) pair (ASCII values of 0D & 0A hex)*

Modbus TCP Frame Format		
Name	**Length**	**Function**
Transaction Identifier	2 bytes	*For synchronization between messages of server* & client
Protocol Identifier	2 bytes	*Zero for MODBUS/TCP*
Length Field	2 bytes	*Number of remaining bytes in this frame*
Unit Identifier	1 byte	*Slave Address (255 if not used)*
Function code	1 byte	*Function codes as in other variants*
Data bytes	n bytes	*Data as response or commands*

Unit identifier is used with MODBUS/TCP devices that are composites of several MODBUS devices, e.g. on MODBUS/TCP to MODBUS RTU gateways. In such case, the unit identifier tells the Slave Address of the device behind the gateway. Natively MODBUS/TCP-capable devices usually ignore the Unit Identifier.

Supported Function Codes

The various reading, writing and other operations are categorised as follows. The most primitive reads and writes are shown in bold. A number of sources use alternative terminology, for example *Force Single Coil* where the standard uses *Write Single Coil*.

			Function Name	**Function Code**
Data Accesss	Bit access	Physical Discrete Inputs	**Read Discrete Inputs**	2
		Internal Bits or Physical Coils	**Read Coils**	1
			Write Single Coil	5
			Write Multiple Coils	15
	16-bit access	Physical Input Registers	**Read Input Register**	4
		Internal Registers or Physical Output Registers	**Read Holding Registers**	3
			Write Single Register	6
			Write Multiple Registers	16
			Read/Write Multiple Registers	23
			Mask Write Register	22
			Read FIFO Queue	24
	File Record Access		Read File Record	20
			Write File Record	21
Diagnostics			Read Exception Status	7
			Diagnostic	8
			Get Com Event Counter	11
			Get Com Event Log	12
			Report Slave ID	17
			Read Device Identification	43
Other			Encapsulated Interface Transport	43

Implementations

Almost all implementations have variations from the official standard. Different varieties might not communicate correctly between equipment of different suppliers. Some of the most common variations are:

- Data types
  - Floating point IEEE
  - 32-bit integer
  - 8-bit data
  - Mixed data types
  - Bit fields in integers
  - Multipliers to change data to/from integer. 10, 100, 1000, 256 ...

- Protocol extensions
    - 16-bit slave addresses
    - 32-bit data size (1 address = 32 bits of data returned.)
    - Word swapped data

Limitations

- Since Modbus was designed in the late 1970s to communicate to programmable logic controllers, the number of data types is limited to those understood by PLCs at the time. Large binary objects are not supported.

- No standard way exists for a node to find the description of a data object, for example, to determine if a register value represents a temperature between 30 and 175 degrees.

- Since Modbus is a master/slave protocol, there is no way for a field device to "report by exception" (except over Ethernet TCP/IP, called open-mbus)- the master node must routinely poll each field device, and look for changes in the data. This consumes bandwidth and network time in applications where bandwidth may be expensive, such as over a low-bit-rate radio link.

- Modbus is restricted to addressing 247 devices on one data link, which limits the number of field devices that may be connected to a master station (once again Ethernet TCP/IP proving the exception).

- Modbus transmissions must be contiguous which limits the types of remote communications devices to those that can buffer data to avoid gaps in the transmission.

- Modbus protocol provides no security against unauthorized commands or interception of data.