Industrial Computing

Sharla Kenney

First Edition, 2012

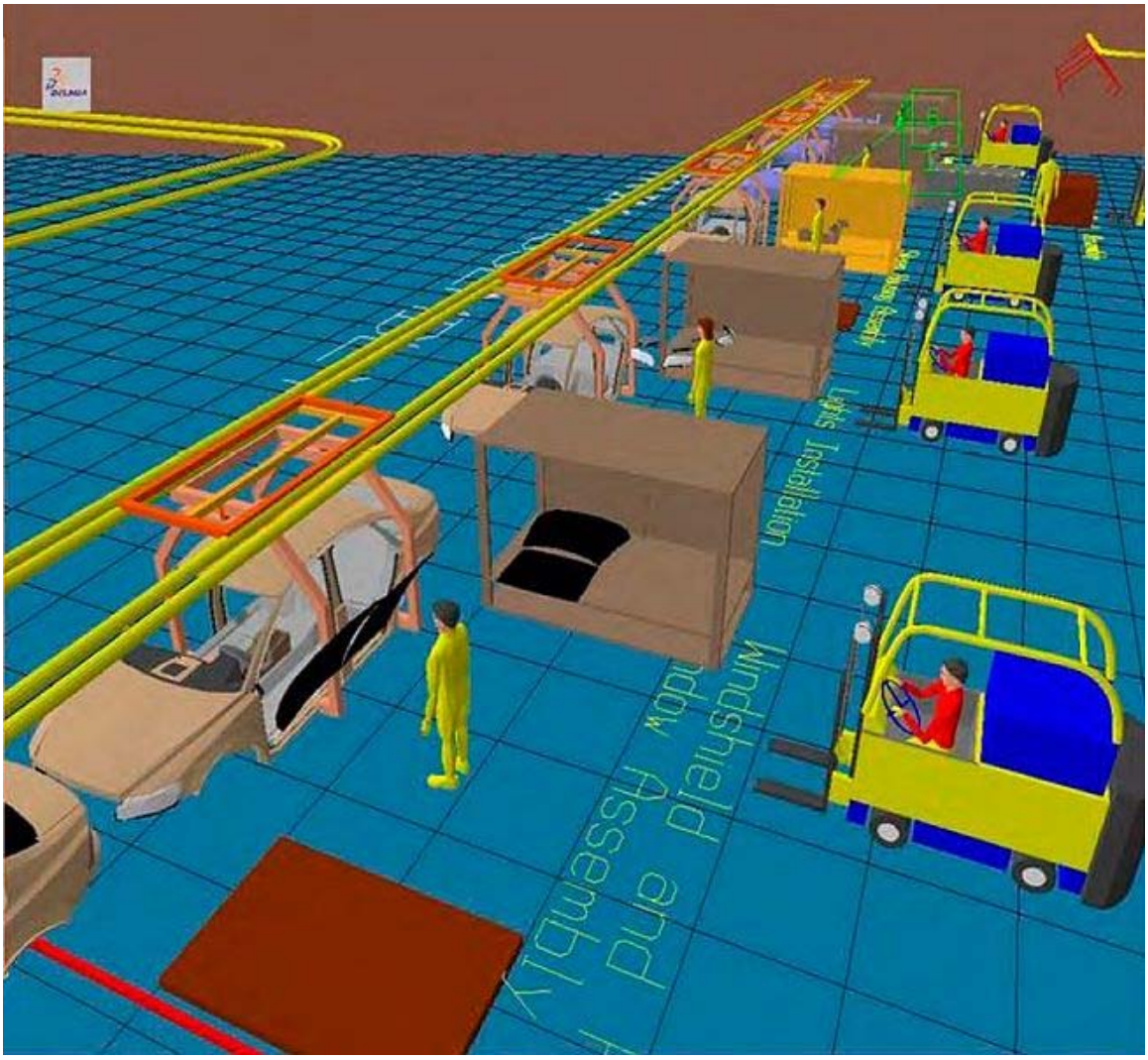
ISBN 978-81-323-3072-1

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

Table of Contents

Chapter 1

Computer-Integrated Manufacturing



Manufacturing Systems Integration Program, NIST 2008.

Computer-integrated manufacturing (**CIM**) is the manufacturing approach of using computers to control the entire production process. This integration allows individual processes to exchange information with each other and initiate actions. Through the integration of computers, manufacturing can be faster and less error-prone, although the main advantage is the ability to create automated manufacturing processes. Typically CIM relies on closed-loop control processes, based on real-time input from sensors. It is also known as flexible design and manufacturing.

Overview

The term "computer-integrated manufacturing" is both a method of manufacturing and the name of a computer-automated system in which individual engineering, production, marketing, and support functions of a manufacturing enterprise are organized. In a CIM system functional areas such as design, analysis, planning, purchasing, cost accounting, inventory control, and distribution are linked through the computer with factory floor functions such as materials handling and management, providing direct control and monitoring of all the operations.

As a method of manufacturing, three components distinguish CIM from other manufacturing methodologies:

- Means for data storage, retrieval, manipulation and presentation;
- Mechanisms for sensing state and modifying processes;
- Algorithms for uniting the data processing component with the sensor/modification component.

CIM is an example of the implementation of information and communication technologies (ICTs) in manufacturing.

CIM implies that there are at least two computers exchanging information, e.g. the controller of an arm robot and a micro-controller of a CNC machine.

Some factors involved when considering a CIM implementation are the production volume, the experience of the company or personnel to make the integration, the level of the integration into the product itself and the integration of the production processes. CIM is most useful where a high level of ICT is used in the company or facility, such as CAD/CAM systems, the availability of process planning and its data.

History

The idea of "digital manufacturing" was prominent the 1980s, when computer-integrated manufacturing was developed and promoted by machine tool manufacturers and the Computer and Automated Systems Association and Society of Manufacturing Engineers (CASA/SME).

"*CIM is the integration of total manufacturing enterprise by using integrated systems and data communication coupled with new managerial philosophies that improve organizational and personnel efficiency*." ERHUM

Computer-integrated manufacturing topics

Key challenges

There are three major challenges to development of a smoothly operating computer-integrated manufacturing system:

- Integration of components from different suppliers: When different machines, such as CNC, conveyors and robots, are using different communications protocols. In the case of AGVs, even differing lengths of time for charging the batteries may cause problems.

- Data integrity: The higher the degree of automation, the more critical is the integrity of the data used to control the machines. While the CIM system saves on labor of operating the machines, it requires extra human labor in ensuring that there are proper safeguards for the data signals that are used to control the machines.

- Process control: Computers may be used to *assist* the human operators of the manufacturing facility, but there must always be a competent engineer on hand to handle circumstances which could not be foreseen by the designers of the control software.

Subsystems in computer-integrated manufacturing

A computer-integrated manufacturing system is not the same as a *"lights-out" factory*, which would run completely independent of human intervention, although it is a big step in that direction. Part of the system involves flexible manufacturing, where the factory can be quickly modified to produce different products, or where the volume of products can be changed quickly with the aid of computers. Some or all of the following subsystems may be found in a CIM operation:

Computer-aided techniques:

- CAD (computer-aided design)
- CAE (computer-aided engineering)
- CAM (computer-aided manufacturing)
- CAPP (computer-aided process planning)
- CAQ (computer-aided quality assurance)
- PPC (production planning and control)
- ERP (enterprise resource planning)
- A business system integrated by a common database.

Devices and equipment required:

- CNC, Computer numerical controlled machine tools
- DNC, Direct numerical control machine tools
- PLCs, Programmable logic controllers
- Robotics
- Computers
- Software
- Controllers
- Networks
- Interfacing
- Monitoring equipment

Technologies:

- FMS, (flexible manufacturing system)
- ASRS, automated storage and retrieval system
- AGV, automated guided vehicle
- Robotics
- Automated conveyance systems

Others:

- Lean manufacturing

CIMOSA

CIMOSA (Computer Integrated Manufacturing Open System Architecture), is a 1990s European proposal for an open system architecture for CIM developed by the AMICE Consortium as a series of ESPRIT projects. The goal of CIMOSA was "to help companies to manage change and integrate their facilities and operations to face world wide competition. It provides a consistent architectural framework for both enterprise modeling and enterprise integration as required in CIM environments".

CIMOSA provides a solution for business integration with four types of products:

- The CIMOSA Enterprise Modeling Framework, which provides a reference architecture for enterprise architecture
- CIMOSA IIS, a standard for physical and application integration.
- CIMOSA Systems Life Cycle, is a life cycle model for CIM development and deployment.
- Inputs to standardization, basics for international standard development.

CIMOSA according to Vernadat (1996), coined the term business process and introduced the process-based approach for integrated enterprise modeling based on a cross-boundaries approach, which opposed to traditional function or activity-based approaches.

With CIMOSA also the concept of an "Open System Architecture" (OSA) for CIM was introduced, which was designed to be vendor-independent, and constructed with standardised CIM modules. Here to the OSA is "described in terms of their function, information, resource, and organizational aspects. This should be designed with structured engineering methods and made operational in a modular and evolutionary architecture for operational use".

Application

There are multiple areas of usage:

- In mechanical engineering
- In electronic design automation (printed circuit board (PCB) and integrated circuit design data for manufacturing)

Chapter 2

AS-Interface

AS-Interface (Actuator Sensor Interface, AS-i) is an industrial networking solution (physical layer, data access method and protocol) used in PLC, DCS and PC-based automation systems. It is designed for connecting simple field I/O devices (e.g. binary ON/OFF devices such as actuators, sensors, rotary encoders, analog inputs and outputs, push buttons, and valve position sensors) in discrete manufacturing and process applications using a single 2-conductor cable.

AS-Interface is an 'open' technology supported by a multitude of automation equipment vendors. Currently, over 16 Million AS-i field devices are installed globally, growing at about 2 million per year.

AS-Interface is a networking alternative to the hard wiring of field devices. It can be used as a partner network for higher level fieldbus networks such as *Profibus, *DeviceNet, Interbus and *Industrial Ethernet, for whom it offers a low-cost remote I/O solution. It is used in automation applications, including conveyor control, packaging machines, process control valves, bottling plants, electrical distribution systems, airport carousels, elevators, bottling lines and food production lines.

AS-Interface provides a basis for Functional Safety in machinery safety/emergency stop applications. Safety devices communicating over AS-Interface follow all the normal data rules. The required level of data verification is provided by dynamic changes in the data. This technology is called Safety as Work and allows safety devices and standard, non-safe devices to be connected to the same network. Using appropriate safe input hardware (e.g. light curtains, e-stop buttons, and door interlock switches), AS-Interface can provide safety support up to SIL (Safety Integrity Level) 3 according to IEC 61508, CAT 4 according to EN954-1 as well as Performance Level e (PL e) according to EN ISO 13849-1.

The AS-Interface specification is managed by AS-International, a member funded organization located in Gelenhausen/Germany. Several international daughter organizations exist around the world.

Overview

AS-Interface is a system that require four basic components:

- Exactly one *network master*, in most cases in the form of a Gateway to a higher level industrial network or a PLC backplane card,
- A number of *network slaves*, in most cases input and output modules,
- A power supply used to power the *network slaves* and enabling communication with the *network master*, and
- The wiring infrastructure, in most case accomplished using the *yellow flat cable*.

The underlying communication procedure is a Master-Slave method, by which the master initiates data exchange with a slave and requires that slave to respond within in define maximum time, making AS-Interface a deterministic networking solution. Strict conformance testing by an independent certification lab assures that certified products from all manufacturers will communicate on a given network. AS-Interface data exchanges are based on a *Master-Call*, where the data frame consists of a 5-bit device addresses and 4-bit data packets (e.g. digital output information). The total length of the *Master-Call* 14 bit. The resulting *Slave-Response* is 7 bit long, containing 4 bit of user information (e.g. the values of slaves inputs).

Voltage levels on the network range between 29,5 .. 31,6 V DC and data protection, in addition to the framing bits, is accomplished via Manchester-II coding, a highly symmetrical, floating layout with Alternating Pulse Modulation. The networks bit time is 6 µs. Segment length is limited to 100 meters.

History

AS-Interface was developed during the late 1980 and early 1990 by a group (consortium) of 11 companies mostly known for their offering of industrial non-contact sensing devices like inductive sensors, photoelectric sensors, capacitive sensors and ultrasonic sensors. Once development was completed the consortium was resolved and a member organization, AS-International, was founded. The first operational system was shown at the 1994 Hannover fair (Hannover Messe).

Original Specification (1994, Version 2.04)

In its original form the network was capable of supporting up to 31 binary I/O devices, where each device could exchange 4 bit of input and 4 bit of output data, resulting in a total of 124 inputs and 124 outputs on a single network. Important features like **Automatic Single Node Replacement** were already part of the system. The network update time is easily calculated by multiplying the number I/O nodes with the deterministic update time for each node (approximately 150 microseconds), for a maximum update time of 5 ms. This simplified calculation does not include the *Management Phase* which is negligible for typical installations.

Enhancements (1998, Version 2.11)

Following its introduction users quickly adopted AS-Interface, driving the demand for additional functionality and features. As a consequence, these demands were addressed with certain specification enhancements allowing the creation of analog input/output devices and increasing the number of possible binary I/O devices to 62. Diagnostics functionality was also enhanced by the creation of the **Peripheral Fault Bit**. In order to retain full forward and backward compatibility, the size of the data frame exchanged between the network master (Scanners and Gateways) was not increased. Instead, one of the four output bits was used to select between the so-called A and B nodes. This enabled each of the 31 addresses to be used twice. The address space was increased to 1A to 31A plus 1B to 31B. As a consequence of using an output bit as the A/B selector, the fourth output bit was not available to the user and binary I/O nodes built to this profile offered a maximum of 4 inputs and 3 outputs, increasing the total amount of I/O on a single network to 248 inputs and 186 outputs. The maximum update time of a fully loaded network is 10 ms.

Additional capabilities (2005/2007, Version 3.0)

By 2005 it became necessary to address additional user requirements. Also, the increased usage of Ethernet based industrial protocols called for a low level solution that overcame the inherent shortcomings of Ethernet (e.g. restricted topology, large data frame, costly usage of switches ...) This specification addressed the users requirements by defining new communication profiles for binary and analog data plus the introduction of a serial data transmission profile. The following is an incomplete list of the new capabilities

- Binary I/O nodes supporting A/B addressing with 4 Inputs and 4 Outputs
- Binary I/O nodes supporting A/B addressing with 8 Inputs and 8 Outputs
- Configurable (8, 12 or 16 bit) fast analog channel
- Full Duplex bit serial data channel

With these new capabilities, AS-Interface becomes the ideal partner network for any of the currently available Ethernet based industrial protocols. Gateways to EtherNet/IP™, PROFINET, Modbus/TCP, SERCOS III and others are available. Some controls experts have voiced the opinion that within the next 10 years networking solutions positioned between AS-Interface and Ethernet will not be used in any new installation. In a worst case scenario, using 62 nodes with 4 inputs and 4 outputs each the update time is 10 ms for the inputs and 20 ms for the outputs.

Components

An AS-Interface network requires only a few basic components falling into the following general categories:

- Scanners and Gateways (also called masters)
- Power supplies and repeaters

- Modules (also called slaves)
- Network cable, installation hardware and useful tool (infrastructure)

Scanners and Gateways

The Scanner/Gateway performs two functions. With respect to the AS-Interface network it is a master, performing the data exchange with the modules and updating its internal I/O image. The functionality of the master is defined in the Master Profile of the AS-Interface specification. As part of specification version 3.0 the M4 Master Profile has been defined. Any given network can only have one Scanner/Gateway. With respect to a connected PLC/DCS or PC the Scanner/Gateway is a slave. The AS-Interface community typically uses the word Gateway when the AS-Interface master connects to an upper-level network like DeviceNet, Profibus or any of the industrial Ethernet flavors. On the other hand, if it resides on the backplane of a PLC it is usually referred to as a Scanner. Since AS-Interface communication is based on the Master-Slave communication method, any network must have only one Master at a time.

Power supply

Any AS-Interface segment must be powered. This is typically accomplished connecting an AS-Interface power supply. These supplies have certain unique characteristics regarding internal circuitry and output voltage. Standard 24VDC power supplies can not be used to directly power a segment. The total length AS-Interface network cable in a single segment must be no more than 100m. If the total network length must be longer repeaters can be used. As the repeater galvanically isolates any two segment a new power supply must be used on *the far side* of the repeater. A common misconception exists concerning the number of repeaters in a network. It has been stated that the maximum length of an AS-Interface network can be 300m, created by using two repeaters. This is not the case at all! What matters is not how many repeaters are using but rather how many repeaters any data packet, originating at a Scanner or Gateway, has to cross before reaching the I/O node. Due to the tight timing constraints defined each packet can at most travel across two repeaters before reaching an AS-Interface node. This has the following consequences:

1. Linear networks with the Scanner/Gateway mounted at one end can be 300m long
2. Linear network with 600m length can be constructed when the Scanner/Gateway is mounted in the middle segment
3. Star shaped networks with virtually no length limitation are possible

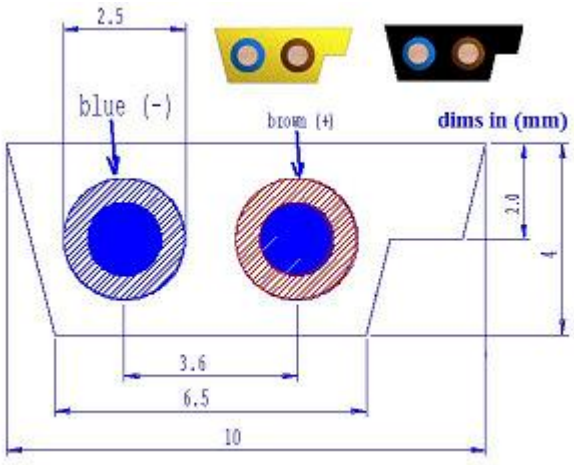
Modules

This is by far the largest group of components and includes binary and analog I/O modules, stack lights, pushbuttons, sensors with integrated ASIC, valve control boxes, E-stops, light curtains; in general any device that can exchange data with the PLC. Each module on the network must have a unique address. For AS-Interface the address space ranges from 0 to 31, where 0 cannot be used, but is reserved for Automatic Single Node

Replacement. Since adoption of specification 2.11 this address space is further divided into A and B extended addresses. As a result, using a module designed to support this addressing mode, it possible to have two modules at each address; one at the A *half* and one at the B *half*. (Ex. 1A and 1B, 17A and 17B)
The current specification Version 3.0 has adds many the ability to construct many new types of I/O combinations, including binary modules with 4 inputs and 4 outputs supporting A/B addressing.

Network cable

The vast majority of AS-Interface installations utilize the AS-Interface flat cable, defined as part of the AS-Interface specifications. A relatively small number of industries (e.g. valve control in process automation) use a round cable, mostly because it is easier to pull through conduit. While the shape of the cable does not matter (any other cable can be used) the electrical characteristics of the selected cable matters greatly. To prevent problems due to improper cable, most professional suggest the AS-Interface flat cable. This cable is designed to make use of a cable piercing technology. When an AS-Interface module is installed on the network, piercing needles penetrate the cable jacket and displace the internal copper strands without cutting them. This allows AS-Interface modules to be installed anywhere on the network without cutting and preparing (i.e. removing cable jacket, stripping insulation and possibly applying a ferule) the cable first. The result is a faster installation without the chance of inadvertent shorts between the leads.



Here is a flat cable drawing

There are several types of cables available. Yellow cable is usually used to power AS-Interface modules and enable communication between the field devices and the scanner or Gateway. This cable is offered in several jacket materials to address specific applications needs. The AS-Interface black cable is typically used to supply modules with 24VDC AUX power. No communication takes place on this cable. Similar to the yellow cable, the black cable is also produced using various jacket material to address the

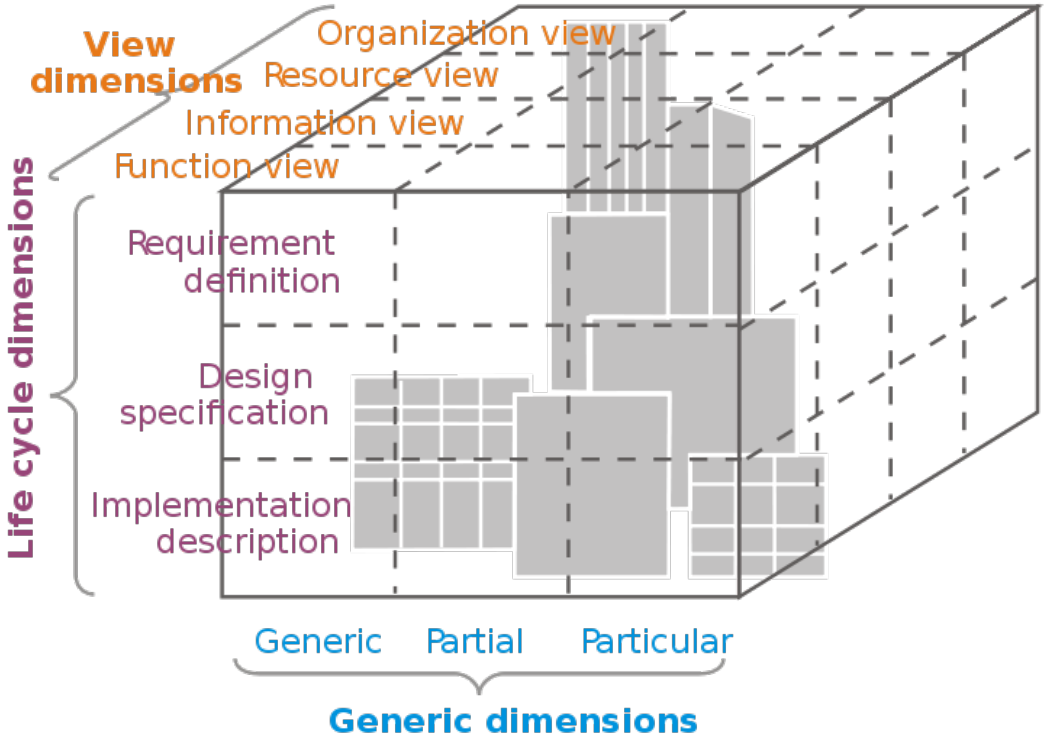
specific needs of the application. A red jacketed cable has been defined but is virtually unused. Its intended use was in applications where AC power is supplied to the field nodes. The two leads inside the AS-Interface cable are brown (+ lead) and blue (- lead) independent of material makeup and outer jacket color.

Other components

Passive taps, flat-to-round cable adapters, handheld addressing tools and many other accessories are designed to further assist in the installation of AS-Interface networks.

Chapter 3

CIMOSA



CIMOSA cube: The basics of the reference architecture from which a particular architecture is developed.

CIMOSA stands for "Computer Integrated Manufacturing Open System Architecture", is a enterprise modeling framework, which aims to support the enterprise integration of machines, computers and people. The framework is based on the system life cycle concept, and offers a modelling language, methodology and supporting technology to support these goals.

It was developed in the 1990s by the AMICE Consortium, in an EU project. Eventually the non-profit organization CIMOSA Association has been established to keep ownership of the CIMOSA specification, to promote it and to support its further evolution.

Overview

The original aim of CIMOSA (1992) has been "to elaborate an open system architecture for CIM and to define a set of concepts and rules to facilitate the building of future CIM systems". One of the main idea of CIMOSA is the categorization of manufacturing operations in:

- Generic functions : Generic parts of every enterprise, independent of its organisation-structure or business area.
- Specific (Partial and Particular) functions : Specific for individual enterprises.

Eventually the development of CIMOSA has resulted in two key items:

- *Modeling Framework* : This framework supports "all phases of the CIM system life-cycle from requirements definition, through design specification, implementation description and execution of the daily enterprise operation".
- *Integrating Infrastructure* : This infrastructure provides "specific information technology services for the execution of the Particular Implementation Model", which has providen to be vendor independent and portable.

The framework furthermore offers an "event-driven, process-based modeling approach with the goal to cover essential enterprise aspects in one integrated model. The main aspects are the functional, behavioral, resource, information and organizational aspect".

CIMOSA can be applied in process simulation and analysis. Standardized CIMOSA models "can also be used on line in the manufacturing enterprise for scheduling, dispatching, monitoring and providing process information". One of the standards based on CIMOSA is the Generalised Enterprise Reference Architecture and Methodology (GERAM).

Building blocks

The main focus of CIMOSA has been to construct:

- a framework for enterprise modelling, a reference architecture
- an enterprise modelling language
- an integrating infrastructure for model enactment supported by
- a common terminology

A close liaison with European and international standardisation organisations has been established to stimulate the standardisation process for enterprise integration.

CIMOSA aims at integrating enterprise operations by means of efficient information exchange within the enterprise. CIMOSA models enterprises using four perspectives:

- the *function view* describes the functional structure required to satisfy the objectives of an enterprise and related control structures;
- the *information view* describes the information required by each function;
- the *resource view* describes the resources and their relations to functional and control structures; and
- the *organization view* describes the responsibilities assigned to individuals for functional and control structures.

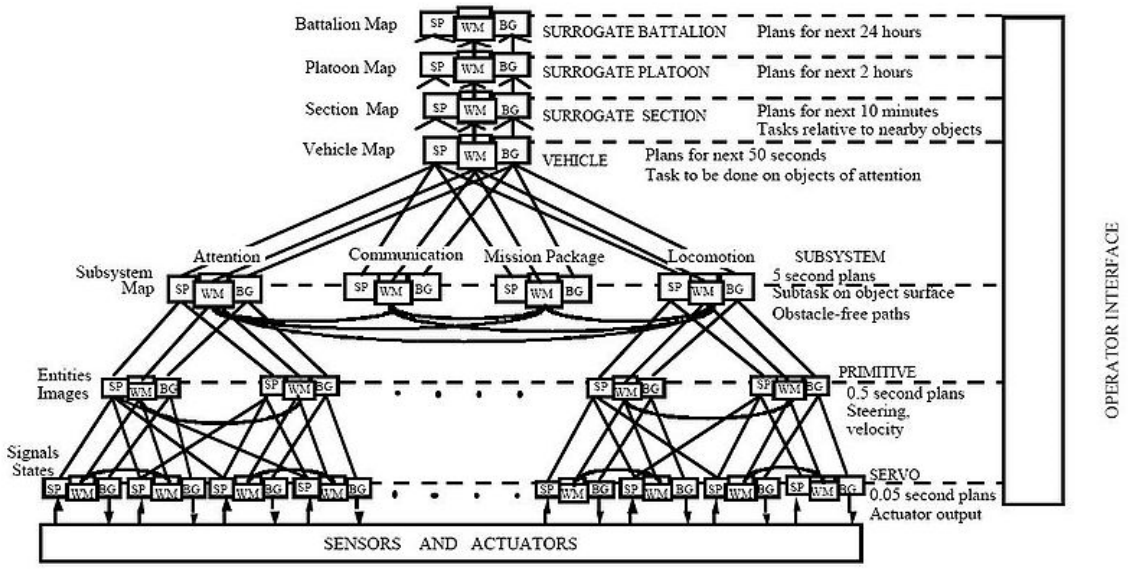
CIMOSA Association

Begin 1990s the CIMOSA Association (COA) was founded by the AMICE Consortium as a non-profit organisation aiming to promote Enterprise Engineering and Integration (EE&I) based on CIMOSA. It has extended its goals in the new millennium towards "upcoming new enterprise paradigms of extended, virtual and agile enterprises, which cause new requirements on organisational concepts and supporting technologies. Enhanced decision support and operation monitoring and control are some of the needs of today and tomorrow. Capturing knowledge and using it across organisational boundaries will be a major challenge in the new types of businesses. This real-time knowledge needed to support the establishment, deployment and discontinuation of the inter and intra organisational relations".

From the start CIMOSA has been an active supporter for national, European and international standardisation of Enterprise Integration.

Chapter 4

4D-RCS Reference Model Architecture



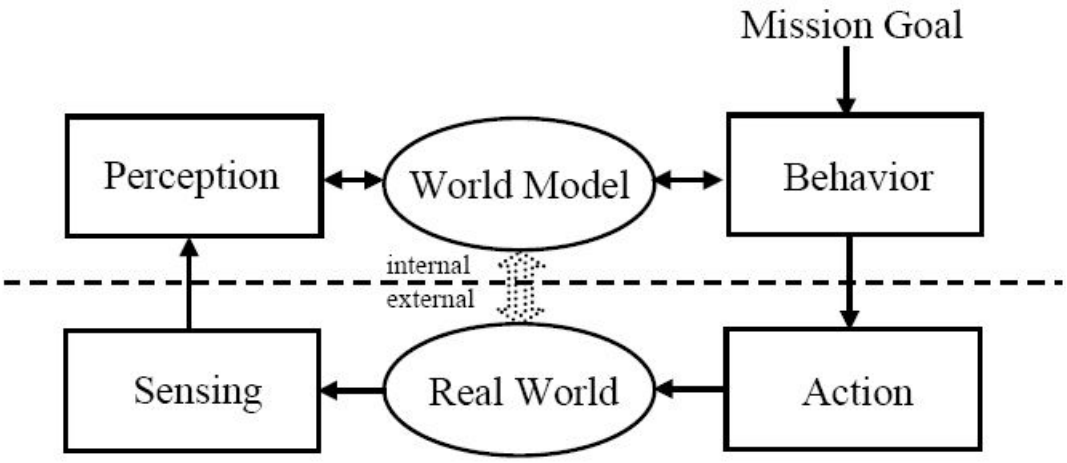
4D-RCS reference model architecture for an individual vehicle. It contains many layers of computational nodes each containing elements of sensory processing, world modeling, value judgment, and behavior generation.

The **4D/RCS Reference Model Architecture** is a reference model for military unmanned vehicles on how their software components should be identified and organized.

4D/RCS has been developed by the National Institute of Standards and Technology (NIST). It is based on the general Real-time Control System (RCS) Reference Model Architecture , and has been applied to many kinds of robot control, including autonomous vehicle control.

Overview

4D/RCS is a reference model architecture that provides a theoretical foundation for designing, engineering, integrating intelligent systems software for unmanned ground vehicles.



Fundamental structure of a 4D/RCS control loop.

According to Balakirsky (2003) 4D/RCS is an example of deliberative agent architecture. These architectures "include all systems that plan to meet future goal or deadline. In general, these systems plan on a model of the world rather than planning directly on processed sensor output. This may be accomplished by real-time sensors, a priori information, or a combination of the two in order to create a picture or snapshot of the world that is used to update a world model". The course of action of a deliberative agent architecture is based on the world model and the commanded mission goal, see image. This goal "may be a given system state or physical location. To meet the goal systems of this kind attempts to compute a path through a multi-dimensional space contained in the real world".

The 4D/RCS is a hierarchical deliverative architecture, that "plans up to the subsystem level to compute plans for an autonomous vehicle driving over rough terrain. In this system, the world model contains a pre-computed dictionary of possible vehicle trajectories known as an ego-graph as well as information from the real-time sensor processing. The trajectories are computed based on a discrete set of possible vehicle velocities and starting steering agles. All of the trajectories are guaranteed to be dynamically correct for the given velocity and steering angle. The systems runs under a fixed planning cycle, with the sensed information being updated into the world model at the beginning of the cycle. These update information include information on what area is currently under observation by the sensors, where detected obstacles exist, and vehicle status".

History

The National Institute of Standards and Technology's (NIST) Intelligent Systems Division (ISD) has been developing the RCS reference model architecture for over 30 years. 4D/RCS is the most recent version of RCS developed for the Army Research Lab Experimental Unmanned Ground Vehicle program. The 4D in 4D/RCS signifies adding time as another dimension to each level of the three dimensional (sensor processing, world modeling, behavior generation), hierarchical control structure. ISD has studied the use of 4D/RCS in defense mobility, transportation, robot cranes, manufacturing, and several other applications.

4D/RCS integrates the NIST Real-time Control System (RCS) architecture with the German (Bundeswehr University of Munich) VaMoRs 4-D approach to dynamic machine vision. It incorporates many concepts developed under the U.S. Department of Defense Demo I, Demo II, and Demo III programs, which demonstrated increasing levels of robotic vehicle autonomy. The theory embodied in 4D/RCS borrows heavily from cognitive psychology, semiotics, neuroscience, and artificial intelligence.

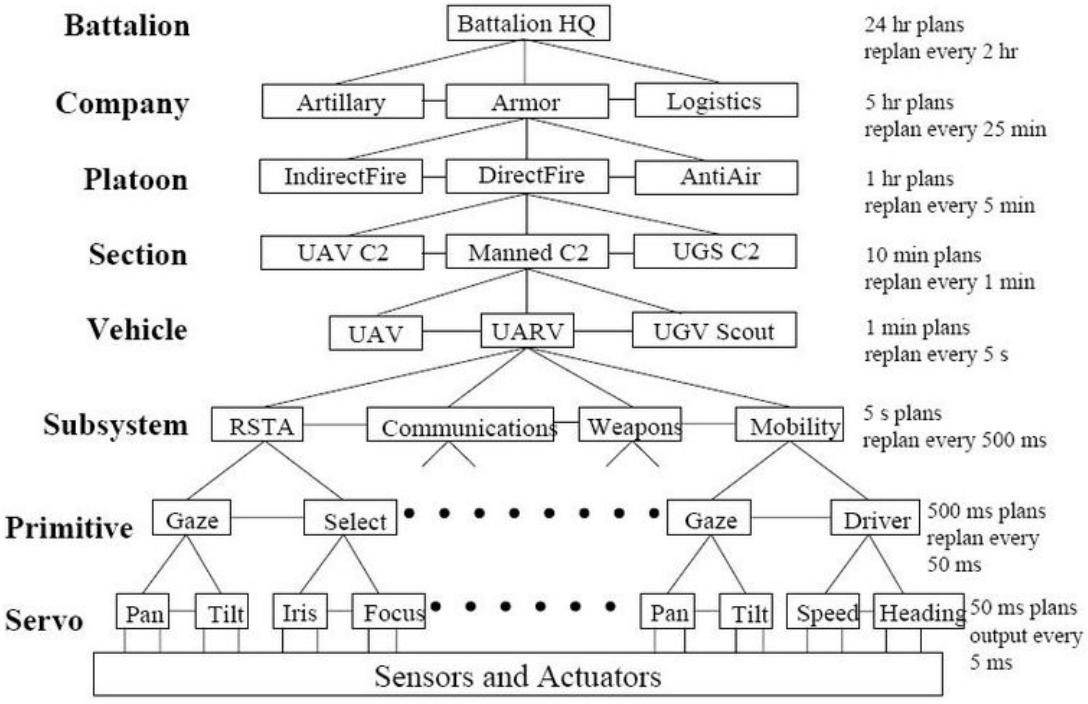
Three US Government funded military efforts known as Demo I (US Army), Demo II (DARPA), and Demo III (US Army), are currently underway. Demo III (2001) demonstrated the ability of unmanned ground vehicles to navigate miles of difficult off-road terrain, avoiding obstacles such as rocks and trees. James Albus at NIST provided the Real-time Control System which is a hierarchical control system. Not only were individual vehicles controlled (e.g. throttle, steering, and brake), but groups of vehicles had their movements automatically coordinated in response to high level goals.

In 2002, the DARPA Grand Challenge competitions were announced. The 2004 and 2005 DARPA competitions allowed international teams to compete in fully autonomous vehicle races over rough unpaved terrain and in a non-populated suburban setting. The 2007 DARPA challenge, the DARPA urban challenge, involved autonomous cars driving in an urban setting.

4D/RCS Building blocks

The 4D/RCS architecture is characterized by a generic control node at all the hierarchical control levels. The 4D/RCS hierarchical levels are scalable to facilitate systems of any degree of complexity. Each node within the hierarchy functions as a goal-driven, model-based, closed-loop controller. Each node is capable of accepting and decomposing task commands with goals into actions that accomplish task goals despite unexpected conditions and dy-namic perturbations in the world.

4D/RCS Hierarchy



A high level block diagram of a typical 4D/RCS reference model architecture. UAV = Unmanned Air Vehicle, UARV = Unmanned Armed Reconnaissance Vehicle, UGS = Unattended Ground Sensors.

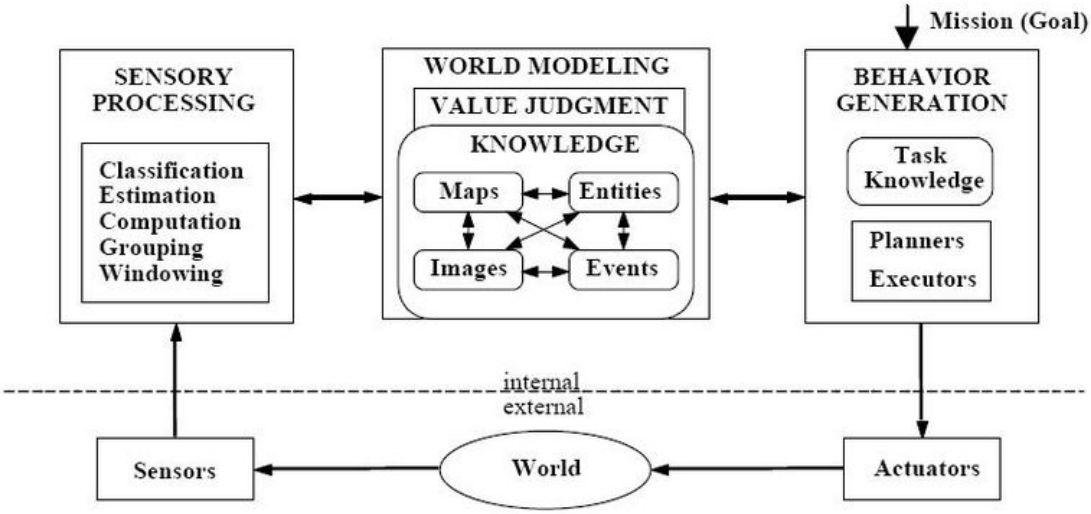
4D/RCS prescribes a hierarchical control principle that decomposed high level commands into actions that employ physical actuators and sensors. The figure for example shows a high level block diagram of a 4D/RCS reference model architecture for a notional Future Combat System (FCS) battalion. Commands flow down the hierarchy, and status feedback and sensory information flows up. Large amounts of communication may occur between nodes at the same level, particularly within the same subtree of the command tree:

- At the *Servo level* : Commands to actuator groups are decomposed into control signals to individual actuators.
- At the *Primitive level* : Multiple actuator groups are coordinated and dynamical interactions between actuator groups are taken into account.
- At the *Subsystem level* : All the components within an entire subsystem are coordinated, and planning takes into consideration issues such as obstacle avoidance and gaze control.
- At the *Vehicle level* : All the subsystems within an entire vehicle are coordinated to generate tactical behaviors.
- At the *Section level* : Multiple vehicles are coordinated to generate joint tactical behaviors.

- At the *Platoon level* : Multiple sections containing a total of 10 or more vehicles of different types are coordinated to generate platoon tactics.
- At the *Company level* : Multiple platoons containing a total of 40 or more vehicles of different types are coordinated to generate company tactics.
- At the *Battalion level* : Multiple companies containing a total of 160 or more vehicles of different types are coordinated to generate battalion tactics.

At all levels, task commands are decomposed into jobs for lower level units and coordinated schedules for subordinates are generated. At all levels, communication between peers enables coordinated actions. At all levels, feedback from lower levels is used to cycle subtasks and to compensate for deviations from the planned situations.

4D/RCS control loop



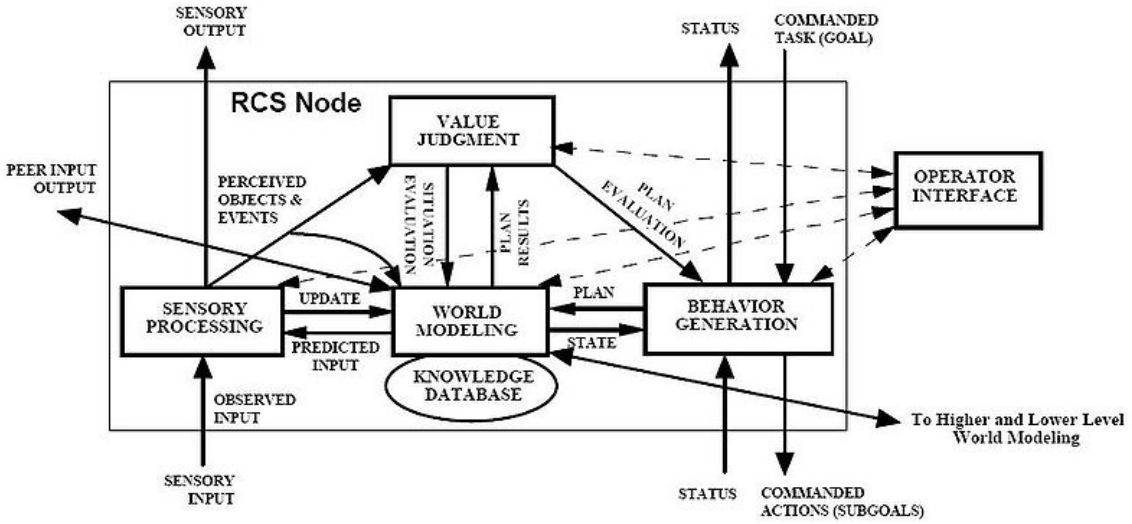
4D-RCS control loop basic internal structure.

At the heart of the control loop through each node is the world model, which provides the node with an internal model of the external world. The world model provides a site for data fusion, acts as a buffer between perception and behavior, and supports both sensory process-ing and behavior generation.

A high level diagram of the internal structure of the world model and value judgment system is shown in the figure. Within the knowledge database, iconic information (images and maps) is linked to each other and to symbolic information (entities and events). Situations and relationships between entities, events, images, and maps are represented by pointers. Pointers that link symbolic data struc-tures to each other form syntactic, semantic, causal, and situational networks. Pointers that link symbolic data structures to regions in images and maps provide symbol grounding and enable the world model to project its understanding of reality onto the physical world.

Sensory processing performs the functions of windowing, grouping, computation, estimation, and classification on input from sensors. World modeling maintains knowledge in the form of images, maps, entities, and events with states, attributes, and values. Relationships between images, maps, entities, and events are defined by pointers. These relationships include class membership, ontologies, situations, and inheritance. Value judgment provides criteria for decision making. Behavior generation is responsible for planning and execution of behaviors.

Computional nodes



RCS NODE Internal structure.

The 4D/RCS nodes have internal structure such as shown in the figure. Within each node there typically are four functional elements or processes :

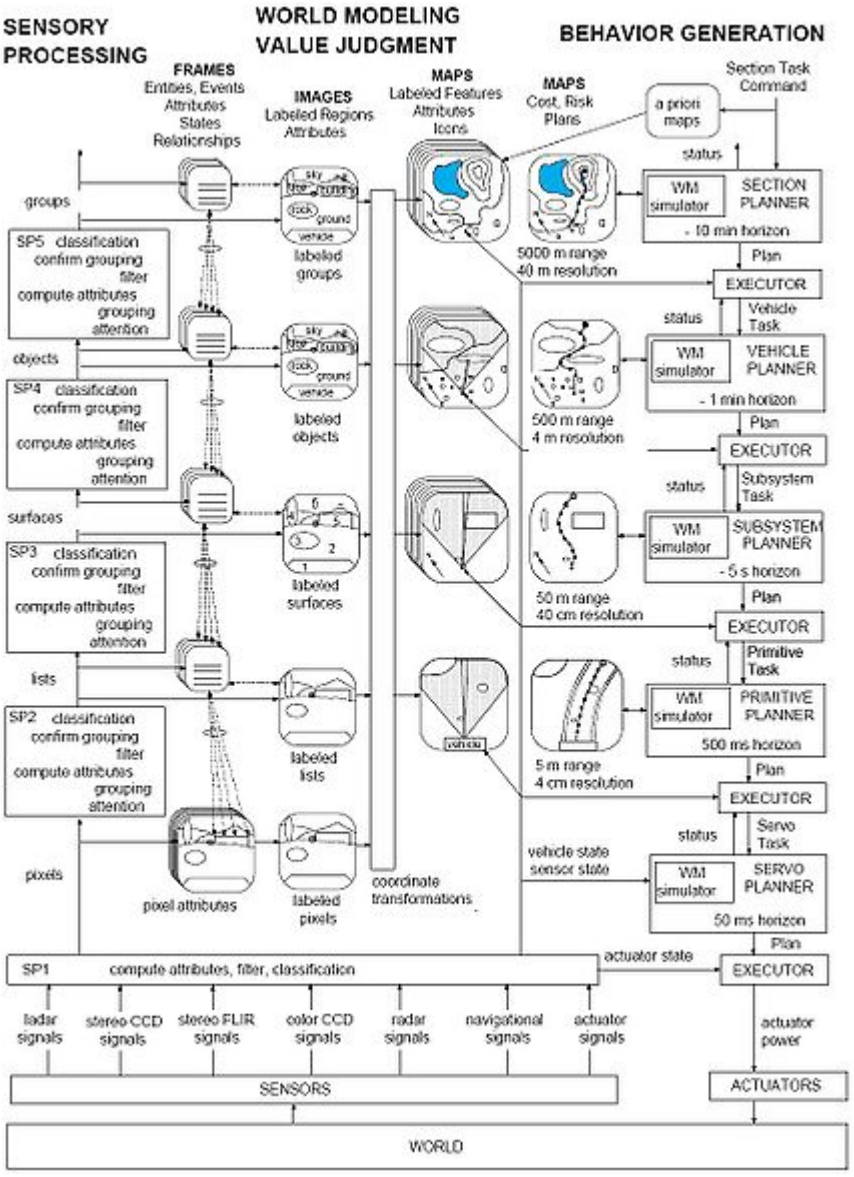
1. behavior generation,
2. world modeling,
3. sensory processing, and
4. value judgment.

There is also a knowledge database that represents the node's best estimate of the state of the world at the range and resolution that are appropriate for the behavioral decisions that are the responsibility of that node.

These are supported by a knowledge database, and a communication system that interconnects the functional processes and the knowledge database. Each functional element in the node may have an operator interface. The connections to the Operator Interface enable a human operator to input commands, to override or modify system behavior, to perform various types of teleoperation, to switch control modes (e.g., automatic, teleoperation, single step, pause), and to observe the values of state variables,

images, maps, and entity attributes. The Operator Interface can also be used for programming, debugging, and maintenance.

Five levels of the architecture



Five levels of the 4D/RCS architecture for Demo III.

The figure is a computational hierarchy view of the first five levels in the chain of command containing the Autonomous Mobility Subsystem in the 4D/RCS architecture developed for Demo III. On the right of figure, Behavior Generation (consisting of Planner and Executor) decompose high level mission commands into low level actions. The text inside the Planner at each level indicates the planning horizon at that level.

In the center of the figure, each map has a range and resolution that is appropriate for path planning at its level. At each level, there are symbolic data structures and segmented images with labeled regions that describe entities, events, and situations that are relevant to decisions that must be made at that level. On the left is a sensory processing hierarchy that extracts information from the sensory data stream that is needed to keep the world model knowledge database current and accurate.

The bottom (Servo) level has no map representation. The Servo level deals with actuator dynamics and reacts to sensory feedback from actuator sensors. The Primitive level map has range of 5 m with resolution of 4 cm. This enables the vehicle to make small path corrections to avoid bumps and ruts during the 500 ms planning horizon of the Primitive level. The Primitive level also uses accelerometer data to control vehicle dynamics and prevent rollover during high speed driving.

At all levels, 4D/RCS planners are designed to generate new plans well before current plans become obsolete. Thus, action always takes place in the context of a recent plan, and feedback through the executors closes reactive control loops using recently selected control parameters. To meet the demands of dynamic battlefield environments, the 4D/RCS architecture specifies that replanning should occur within about one-tenth of the planning horizon at each level.

Inter-Node Interactions within a Hierarchy

Sensory processing and behavior generation are both hierarchical processes, and both are embedded in the nodes that form the 4D/RCS organizational hierarchy. However, the SP and BG hierarchies are quite different in nature and are not directly coupled. Behavior generation is a hierarchy based on the decomposition of tasks and the assignment of tasks to operational units. Sensory processing is a hierarchy based on the grouping of signals and pixels into entities and events. In 4D/RCS, the hierarchies of sensory processing and behavior generation are separated by a hierarchy of world modeling processes. The WM hierarchy provides a buffer between the SP and BG hierarchies with interfaces to both.

Criticisms

There have been major criticisms of this architectural form, according to Balakirsky (2003) due to the fact that "the planning is performed on a model of the world rather than on the actual world, and the complexity of the computing large plans... Since the world is not static, and may change during this time delay that occurs between sensing, plan conception, and final execution, the validation of the computed plans have been called into question".

Chapter 5

EPICS & DeviceNet

EPICS

The **Experimental Physics and Industrial Control System** (**EPICS**) is a software environment used to develop and implement distributed control systems to operate devices such as particle accelerators, telescopes and other large experiments. EPICS also provides SCADA capabilities. The tool is designed to help develop systems which often feature large numbers of networked computers providing control and feedback.

EPICS uses client/server and publish/subscribe techniques to communicate between the various computers. One set of computers (the servers or input/output controllers), collect experiment and control data in real-time using the measurement instruments attached to it. This information is given to another set of computers (the clients) using the **Channel Access** (CA) network protocol. CA is a high bandwidth networking protocol, which is well suited to soft real-time applications such as scientific experiments.

Look and feel

EPICS interfaces to the real world with IOCs (Input Output Controllers) . These are either stock-standard PCs or VME standard embedded system processors that manage a variety of "plug and play" modules (GPIB, RS-232, IP Carrier etc.) which interface to control system instruments (oscilloscopes, network analyzers) and devices (motors, thermocouples, switches, etc.). Some instruments also can come with EPICS already embedded within them, like certain Oscilloscopes . The IOC holds and runs a database of 'records' which represent either devices or aspects of the devices to be controlled. IOC software used for hard-real-time normally use RTEMS or VxWorks, though work has been ongoing in porting to other systems. Soft real-time IOC software sometimes runs on Linux or MS-Windows based machines.

Other computers on the network can interact with the IOC via the concept of **channels**. Take, for example a particle accelerator with shutters between sectors. There would typically be several channels corresponding to a shutter: an output channel to activate shutter motion, an input channel to see the status of the shutter (e.g. shut, open, moving, etc.), and probably some additional analog input channels representing temperatures and pressures on each side of the shutter. Channel names are typically in the form EQUIPMENT:SIGNALNAME (e.g. ACCELERATOR_RING:TEMP_PROBE_4, although they can be much less verbose to save time).

Most operations are driven directly from a standalone GUI package such as EDM (extensible display manager) or MEDM (Motif/EDM). These allow creation of GUI screens with dials, gauges, text boxes, simple animations, etc. Newer control systems and GUI interfaces such as CSS/BOY are being investigated.

However it is not just GUI software which can interact with EPICS: any software which can speak the CA protocol can get and put values of records. For example on the EPICS website there are several **extension** packages which allow CA support in things like MATLAB, LabVIEW, Perl, Python, Tcl, ActiveX, etc. Hence it is easy to do things like make scripts which can activate EPICS controlled equipment.

Record types

There are different types of records available in EPICS. Here are some common types. Note that in addition to the other records not mentioned here, it is possible to create your own record type to interact with a device.

Each record has various **fields** in it, which are used for various tasks. **AI and AO** Analog Input and Output records can obviously store an analog value, and are typically used for things like set-points, temperatures, pressure, flow rates, etc. A limited amount of conversion to and from raw device data is available natively in the record (typically scaling and offsetting, but not advanced conversion like two's complement or logarithmic). **BI and BO** Binary Input and Output records are generally used for commands and statuses to and from equipment. **Calc and Calcout** These two records can access other records and perform a calculation based on their values. (E.g. calculate the efficiency of a motor by a function of the current and voltage input and output, and converting to a percentage for the operator to read). **Stepper Motor** Control of a stepper motor. Allows settings of things like accelerations and velocities, as well as position.

Record processing

Records in EPICS must have a designated **scan time**, otherwise they are automatically set as **passive**. A passive record will never process (unless its PROC field is written to). Mostly records are set to process on a periodic scan (such every 0.1 second).

Alternately records may be set to process only on **events**.

Facilities using EPICS

A partial list of facilities using EPICS:

- Australia
  - Australian Synchrotron
  - ANTARES - Australian Nuclear Science and Technology Organisation
  - ASKAP (Australian Square Kilometre Array Pathfinder) - CSIRO
  - Heavy Ion Accelerator at the Australian National University
- Asia
  - KSTAR - Korea Superconducting Tokamak Advanced Research (Republic of Korea)
  - J-PARC - Joint Facility for High Intensity Proton Accelerators (Japan)
  - RIBF - RIKEN RI Beam Factory Project (Japan)
  - BSRF - Beijing Synchrotron Radiation Laboratory (China)
- Europe
  - Berliner Elektronenspeicherring-Gesellschaft für Synchrotronstrahlung (BESSY II) - Helmholtz-Zentrum Berlin (Germany)
  - Deutsches Elektronen Synchrotron (DESY) (Germany)
  - Diamond Light Source - Rutherford Appleton Laboratory (England)
  - International Thermonuclear Experimental Reactor (ITER) (France)
  - Laboratori Nazionali di Legnaro (Italy)
  - Swiss Light Source - Paul Scherrer Institut (Switzerland)
  - GSI/FAIR - (Germany)
  - IFMIF - International Fusion Materials Irradiation Facility (Japan, European Union, United States, and Russia)

- North America
  - Gemini Observatory (United States)
  - Advanced Light Source - Lawrence Berkeley National Laboratory (United States)
  - Advanced Photon Source - Argonne National Laboratory (United States)
  - Canadian Light Source Synchrotron - University of Saskatchewan (Canada)
  - FNAL - Fermi National Accelerator Laboratory (United States)
  - W. M. Keck Observatory (United States)
  - Laser Interferometer Gravitational-Wave Observatory (LIGO) (United States)
  - Los Alamos Neutron Science Center - Los Alamos National Laboratory (United States)
  - National Superconducting Cyclotron Laboratory - Michigan State University (United States)
  - National Synchrotron Light Source - Brookhaven National Laboratory (United States)
  - Spallation Neutron Source - Oak Ridge National Laboratory (United States)

- Stanford Synchrotron Radiation Laboratory - Stanford University (United States)
- Linac Coherent Light Source - SLAC National Accelerator Laboratory (United States)
- TJNAF - Thomas Jefferson National Accelerator Facility (United States)
- TRIUMF - Located on the campus of the University of British Columbia (Canada)

DeviceNet

DeviceNet is a network system used in the automation industry to interconnect control devices for data exchange. It uses Controller Area Network as the backbone technology and defines an application layer to cover a range of device profiles. Typical applications include information exchange, safety devices, and large I/O control networks.

History

DeviceNet was originally developed by American company Allen-Bradley (now owned by Rockwell Automation). It is layered on top of the CAN (Controller Area Network) technology, developed by Bosch. DeviceNet adapts the technology from ControlNet, which is another industrial protocol developed by Allen-Bradley, and takes advantage of CAN, making it low-cost and robust compared to the traditional RS-485 based protocols.

In order to promote the use of DeviceNet worldwide, Rockwell Automation has adopted the "open" concept and decided to share the technology to third party vendors. Hence, it is now managed by the Open DeviceNet Vendors Association (ODVA), an independent organization located in North America. ODVA maintains specifications of DeviceNet and oversees advances to DeviceNet. In addition, ODVA ensures compliance to DeviceNet standards by providing conformance testing and vendor conformity.

ODVA later decided to bring DeviceNet back to its predecessor's umbrella and collectively call the technology as Common Industrial Protocol or (CIP), which includes the following technologies:

- EtherNet/IP (take note of the capital 'N', and "IP" here means "Industrial Protocol")
- ControlNet
- DeviceNet
- CompoNet

ODVA claims high integrity between the three technologies due to the common protocol adaptation, which makes industrial controls much simpler compared to other technologies.

Technical Snapshot

1. Defines the Media, Physical, Data-Link, and Application layers of the ISO/OSI 7-layer model
2. Incorporates trunkline topology with separate buses for signal and power (Typical configuration: two twisted pairs and a single shield)
3. Baudrates defined: 125 kbit/s, 250 kbit/s, and 500 kbit/s
4. Trunk length is inversely proportional to the speed, i.e. 500, 250 and 100 meters respectively
5. A not-so new flat cable was added to the specification to allow the use of the quick-fix connector
6. Up to 64 nodes on a single logical network. (Node addresses range from 0 - 63)
7. Supports master/slave as well as peer-to-peer communication, although majority of the devices work in the master/slave configuration
8. Allows multiple masters on a single logical network
9. Network cable can supply device power along same cable as communication cable (Generally smaller devices such as photo-eyes, limit switches, and proximity switches).
10. Networked devices can be simultaneously controlled and configured
11. Engineered to withstand noisy environments

Architecture

Physical Layer

Nodes are distributed along a DeviceNet network by the means of a trunkline-dropline topology. This topology allows for ease in wiring and access to the network from multiple taps. In addition, nodes can be easily removed and added to reduce production downtime, increase network flexibility, and decrease troubleshooting time. Since the physical layer is optically isolated from device, communication power and device power can share the same bus (Further reducing the complexity of the network and components within). (*Introduction*)

DeviceNet supports 125 kbit/s, 250 kbit/s and 500 kbit/s data rates. Depending on the chosen cable type, DeviceNet can support communication up to 500 meters (Round thick cable). Typical round cable supports up to 100 meters. While flat style cable supports up to 380 meters at 125 kbit/s and 75 meters at 500 kbit/s.(*Physical Layer*)

Data Link Layer

DeviceNet uses a differential serial bus (Controller Area Network) as its Data Link Layer. Using CAN as a backbone, DeviceNet requires minimal bandwidth to transmit and package messages. In addition, a smaller processor may be selected in the design of device thanks to data frame format and the ease at which the processor can parse through the data.

CAN Data Frame Format

```
1 Bit      => Start of Frame
11 Bits    => Identifier
1 Bit      => RTR Bit
6 Bits     => Control Field
0-8 Bytes  => Data Field
15 Bits    => CRC Sequence
1 Bit      => CRC Delimiter
1 Bit      => Acknowledge
1 Bit      => Ack Delimiter
7 Bits     => End of Frame
>2 Bits    => Interframe Space
```

Reference: *Table: Data Frame Format*.

Upon transmitting the first packet of data, the "Start of Frame" bit is sent to synchronize all receivers on the network. The CAN identifier (denoted from 0-63) and RTR bit combine to set priority at which the data can be accessed or changed. Lower identifiers have priority over higher identifiers. In addition to transmitting this data to other devices, the device also monitors the data sent. This redundancy validates the data transmitted and eliminates simultaneous transmissions. If a node is transmitting at the same time as another node, the node with the lower 11 bit identifier will continue to transmit while the device with the higher 11 bit identifier will stop.(*Introduction & Physical Layer*.)

The following 6 bits contain information for specifying the Control Field. The initial two bits are fixed, while the last four are used to specify length field of the Data Field. The Data Field contains from zero to eight bytes of usable data.(*Physical Layer*.)

The following data frame is the CRC (Cyclic Redundancy Check) Field. The frame consists of 15 bits to detect frame errors and maintains numerous format delimiters. Due to ease of implementation and immunity to most noisy networks, CAN provides a high level of error checking and fault confinement.(*Physical Layer*.)

Network

DeviceNet incorporates a connection-based network. A connection must initially be established by either an UCMM (Unconnected Message Manager) or a Group 2 Unconnected Port. From there, Explicit and Implicit messages can be sent and received. Explicit messages are packets of data that general require a response from another device. Typical messages are configurations or non-time sensitive data collection. Implicit messages are packets of data that are time critical and generally communicate real-time data over the network. An Explicit Message Connection has to be used to established first before an Implicit Message Connection is made. Once the connection is made, the CAN identifier routes data to the corresponding node.(*The Network and Transport Layers*.)

Conformance Test

To declare your product as DeviceNet compatible, a vendor needs to send their product to the ODVA test lab for the certification. ODVA used to have a few other test labs around the world, i.e. UK, Japan, and China. It has since been consolidated into one facility in North America.

A full-test version is called the Composite test. It consists of:

1. Conformance test. Test against the protocol specification.
2. Interoperability test. Test against devices from various vendors on a single, fully populated, network.

Conformance Test Procedure

The following procedure shows you how to get your product certified.

1. Register as vendor with ODVA. You will be given a vendor ID.
2. Purchase a copy of the DeviceNet specification. A hard and soft copy will be sent to you.
3. Purchase the conformance test software and corresponding hardware interface card. Note that only selected interface cards from a few vendors can be used.
4. Develop and test product in-house.
5. Submit your product to ODVA test lab for independent verification.
6. Repeat the above two steps until your product successfully pass the independent test.

Chapter 6

Fieldbus

Fieldbus is the name of a family of industrial computer network protocols used for real-time distributed control, now standardized as **IEC 61158**.

A complex automated industrial system — such as a manufacturing assembly line — usually needs an organized hierarchy of controller systems to function. In this hierarchy there is usually a Human Machine Interface (HMI) at the top, where an operator can monitor or operate the system. This is typically linked to a middle layer of programmable logic controllers (PLC) via a non-time-critical communications system (e.g. Ethernet). At the bottom of the control chain is the fieldbus which links the PLCs to the components which actually do the work such as sensors, actuators, electric motors, console lights, switches, valves and contactors.

Description

Fieldbus is an industrial network system for real-time distributed control. It is a way to connect instruments in a manufacturing plant. Fieldbus works on a network structure which typically allows daisy-chain, star, ring, branch, and tree network topologies. Previously computers were connected using RS-232 (serial connections) by which only two devices could communicate. This would be the equivalent of the currently used 4-20 mA communication scheme which requires that each device has its own communication point at the controller level, while the fieldbus is the equivalent of the current LAN-type connections, which require only one communication point at the controller level and allow multiple (hundreds) of analog and digital points to be connected at the same time. This reduces both the length of the cable required and the number of cables required. Furthermore, since devices that communicate through fieldbus require a microprocessor, multiple points are typically provided by the same device. Some fieldbus devices now support control schemes such as PID control on the device side instead of forcing the controller to do the processing.

History

Although fieldbus technology has been around since 1988, with the completion of the ISA S50.02 standard, the development of the international standard took many years. In 1999, the IEC SC65C/WG6 standards committee met to resolve difference in the draft IEC fieldbus standard. The result of this meeting was the initial form of the IEC 61158 standard with eight different protocol sets called "Types" as follows:

- Type 1 Foundation Fieldbus H1
- Type 2 ControlNet
- Type 3 PROFIBUS
- Type 4 P-Net
- Type 5 FOUNDATION Fieldbus HSE (High Speed Ethernet)
- Type 6 SwiftNet (a protocol developed for Boeing, since withdrawn)
- Type 7 WorldFIP
- Type 8 Interbus

This form of "standard" was first developed for the European Common Market, concentrates less on commonality, and achieves its primary purpose — elimination of restraint of trade between nations. Issues of commonality are now left to the international consortia that support each of the fieldbus standard types. Almost as soon as this "8-headed monster" was approved, the IEC standards development work ceased and the committee was dissolved. A new IEC committee SC65C/MT-9 was formed to resolve the conflicts in form and substance within the more than 4000 pages of IEC 61158. The work on the above protocol types is substantially complete. New protocols, such as for safety fieldbuses or realtime ethernet-based fieldbuses are being accepted into the definition of the international fieldbus standard during a typical 5-year maintenance cycle.

Both Foundation Fieldbus and Profibus technologies are now commonly implemented within the process control field, both for new developments and major refits. In 2006, China saw the largest FF (Foundation Fieldbus) systems installations at NanHai and SECCO, each with around 15000 fieldbus devices connected.

IEC 61158 specification

There were many competing technologies for fieldbus and the original hope for one single unified communications mechanism has not been realised. This should not be unexpected since fieldbus technology needs to be implemented differently in different applications; automotive fieldbus is functionally different from process plant control. The final edition of IEC standard IEC 61158 allows 8 technologies.This are the some hierarchic layer of the automation protocols.

IEC 61158 consists of the following parts, under the general title *Digital data communications for measurement and control – Fieldbus for use in industrial control systems*:

- Part 1: Overview and guidance for the IEC 61158 series
- Part 2: Physical Layer specification and service definition
- Part 3: Data Link Service definition
- Part 4: Data Link Protocol specification
- Part 5: Application Layer Service definition
- Part 6: Application Layer Protocol specification

Standards

There are a wide variety of concurring fieldbus standards. Some of the most widely used ones include:

- AS-Interface
- CAN
- Interbus
- LonWorks
- Modbus
- PROFIBUS
- BITBUS
- CompoNet
- SafetyBUS p
- SERCOS interface

Cost advantage

The amount of cabling required is much lower in Fieldbus than in 4-20mA installations. This is because many devices share the same set of cables in a multi-dropped fashion rather than requiring a dedicated set of cables per device as in the case of 4-20mA devices. Moreover, several parameters can be communicated per device in a Fieldbus network whereas only one parameter can be transmitted on a 4-20mA connection.

Networking

With the exception of ARCNET, which was conceived as early as 1975 for office connectivity and later found uses in industry, the majority of fieldbus standards were developed in the 1980s and became fully established in the marketplace during the mid-1990s. In the United States, Allen-Bradley developed standards that eventually grew into DeviceNet and ControlNet; in Europe, Siemens and other manufacturers developed a protocol which evolved into PROFIBUS.

During the 1980s, to solve communication problems between different control systems in cars, the German company Robert Bosch GmbH first developed the Controller Area Network (CAN). The concept of CAN was that every device can be connected by a single set of wires, and every device that is connected can freely exchange data with any other device. CAN soon migrated into the factory automation marketplace (with many others).

Despite each technology sharing the generic name of fieldbus the various fieldbus are not readily interchangeable. The differences between them are so profound that they cannot be easily connected to each other. To understand the differences among fieldbus standards, it is necessary to understand how fieldbus networks are designed. With reference to the OSI model, fieldbus standards are determined by the physical media of the cabling, and layers one, two and seven of the reference model.

For each technology the physical medium and the physical layer standards fully describe, in detail, the implementation of bit timing, synchronization, encoding/decoding, band rate, bus length and the physical connection of the transceiver to the communication wires. The data link layer standard is responsible for fully specifying how messages are assembled ready for transmission by the physical layer, error handling, message-filtering and bus arbitration and how these standards are to be implemented in hardware. The application layer standard, in general defines how the data communication layers are interfaced to the application that wishes to communicate. It describes message specifications, network management implementations and response to the request from the application of services. Layers three to six are not described in fieldbus standards.

Technical committees, with representatives of many different companies, have been responsible for turning the original specifications into international ISO standards. Bury, among others, reports that work is underway to implement a common fieldbus protocol. This will entail a common set of application-layer services that can be provided regardless of the lower-layer implementation details. Although very much in its infancy, it is expected that this protocol may become reality by 2010. Whether designed for low-level sensor communications or high-level machine connectivity (or both), a Fieldbus is an important enabling technology for an open architecture controller.

Features

Different field busses offer different sets of features and performance. It is difficult to make a general comparison of field bus performance because of fundamental differences in data transfer methodology. In the comparison table below it is simply noted if the field bus in question typically supports data update cycles of 1 millisecond or faster.

Field bus	Bus power	Cabling redundancy	Max devices	Synchronisation	Sub millisecond cycle
AS-Interface	Yes	No	62	No	No
CANOpen	No	No	127	Yes	No
ControlNet	No	Yes	99	No	No
CC-Link	No	No	64	No	No
DeviceNet	Yes	No	64	No	No
EtherCAT	No	Yes	65536	Yes	Yes
Ethernet Powerlink	No	Optional	240	Yes	Yes

Field bus	Bus power	Cabling redundancy	Max devices	Synchronisation	Sub millisecond cycle
EtherNet/IP	No	Optional	Almost unlimited	Under development	No
Interbus	No	No	511	No	No
LonWorks	No	No	32000	No	No
Modbus	No	No	246	No	No
PROFIBUS DP	No	Optional	126	Yes	No
PROFIBUS PA	Yes	No	126	No	No
PROFINET IO	No	Optional	Almost unlimited	No	No
PROFINET IRT	No	Optional	Almost unlimited	Yes	Yes
SERCOS III	No	Yes	511	Yes	Yes
SERCOS interface	No	No	254	Yes	Yes
Foundation_Fieldbus_H1	Yes	No	240	No	No

Disadvantages

There are disadvantages to using fieldbus compared to the 4-20 mA analog signal standard (or to 4-20 mA with HART):

- Fieldbus systems are more complex, so users need to be more extensively trained or more highly qualified
- The price of fieldbus components is higher
- Fieldbus test devices are more complex compared to a (high-spec) multimeter that can be used to read and simulate analog 4-20 mA signals
- Slightly longer reaction times with fieldbus, depending on the system
- Device manufacturers have to offer different versions of their devices (e.g. sensors, actuators) due to the number of different (incompatible) fieldbus standards. This can add to the cost of the devices and to the difficulty of device selection and availability.
- One or more fieldbus standards may predominate in future and others may become obsolete. This increases the investment risk when implementing fieldbus.

Current developments

Recently a number of Ethernet-based industrial communication systems have been established, most of them with extensions for real-time communication. These have the potential to replace the traditional field buses in the long term.

Here is a partial list of the new Ethernet-based industrial communication systems:

- EtherCAT
- EtherNet/IP
- Ethernet Powerlink
- PROFINET IO
- PROFINET IRT
- SafetyNET p
- SERCOS III
- TTEthernet
- VARAN

Safety

Fieldbus can be used for systems which must meet safety-relevant standards like IEC 61508 or EN 954-1. Depending on the actual protocol, fieldbus can provide measures like counters, CRC's, echo, timeout, unique sender and receiver ID's or cross check. Ethernet/IP and SERCOS III both use the CIP Safety protocol, Ethernet Powerlink uses openSAFETY, while FOUNDATION Fieldbus and Profibus (PROFIsafe) have varieties of their communications protocol which are compatible with safety systems.

Market

In process control systems, the market is dominated by FOUNDATION fieldbus and PROFIBUS PA. Both technologies use the same physical layer (2-wire manchester-encoded current modulation at 31.25 kHz) but are not interchangeable. As a general guide, applications which are controlled and monitored by PLCs (programmable logic controllers) tend towards PROFIBUS, and applications which are controlled and monitored by a DCS (digital/distributed control system) tend towards FOUNDATION Fieldbus. PROFIBUS technology is made available through Profibus International with headquarters in Karlsruhe, Germany. FOUNDATION Fieldbus technology is owned and distributed by the Fieldbus Foundation of Austin, Texas.

Chapter 7

Programmable Logic Controller

Siemens Simatic S7-400 system at rack, left-to-right: power supply unit PS407 4A,CPU 416-3, interface module IM 460-0 and communication processor CP 443-1.

A **programmable logic controller** (**PLC**) or **programmable controller** is a digital computer used for automation of electromechanical processes, such as control of machinery on factory assembly lines, amusement rides, or lighting fixtures. PLCs are used in many industries and machines. Unlike general-purpose computers, the PLC is designed for multiple inputs and output arrangements, extended temperature ranges, immunity to electrical noise, and resistance to vibration and impact. Programs to control machine operation are typically stored in battery-backed or non-volatile memory. A PLC is an example of a *hard* real time system since output results must be produced in response to input conditions within a bounded time, otherwise unintended operation will result.

History

The PLC was invented in response to the needs of the American automotive manufacturing industry. Programmable logic controllers were initially adopted by the automotive industry where software revision replaced the re-wiring of hard-wired control panels when production models changed.

Before the PLC, control, sequencing, and safety interlock logic for manufacturing automobiles was accomplished using hundreds or thousands of relays, cam timers, and drum sequencers and dedicated closed-loop controllers. The process for updating such facilities for the yearly model change-over was very time consuming and expensive, as electricians needed to individually rewire each and every relay.

In 1968 GM Hydramatic (the automatic transmission division of General Motors) issued a request for proposal for an electronic replacement for hard-wired relay systems. The winning proposal came from Bedford Associates of Bedford, Massachusetts. The first PLC, designated the 084 because it was Bedford Associates' eighty-fourth project, was the result. Bedford Associates started a new company dedicated to developing, manufacturing, selling, and servicing this new product: Modicon, which stood for MOdular DIgital CONtroller. One of the people who worked on that project was Dick Morley, who is considered to be the "father" of the PLC. The Modicon brand was sold in 1977 to Gould Electronics, and later acquired by German Company AEG and then by French Schneider Electric, the current owner.

One of the very first 084 models built is now on display at Modicon's headquarters in North Andover, Massachusetts. It was presented to Modicon by GM, when the unit was retired after nearly twenty years of uninterrupted service. Modicon used the 84 moniker at the end of its product range until the 984 made its appearance.

The automotive industry is still one of the largest users of PLCs.

Development

Early PLCs were designed to replace relay logic systems. These PLCs were programmed in "ladder logic", which strongly resembles a schematic diagram of relay logic. This

program notation was chosen to reduce training demands for the existing technicians. Other early PLCs used a form of instruction list programming, based on a stack-based logic solver.

Modern PLCs can be programmed in a variety of ways, from ladder logic to more traditional programming languages such as BASIC and C. Another method is State Logic, a very high-level programming language designed to program PLCs based on state transition diagrams.

Many early PLCs did not have accompanying programming terminals that were capable of graphical representation of the logic, and so the logic was instead represented as a series of logic expressions in some version of Boolean format, similar to Boolean algebra. As programming terminals evolved, it became more common for ladder logic to be used, for the aforementioned reasons. Newer formats such as State Logic and Function Block (which is similar to the way logic is depicted when using digital integrated logic circuits) exist, but they are still not as popular as ladder logic. A primary reason for this is that PLCs solve the logic in a predictable and repeating sequence, and ladder logic allows the programmer (the person writing the logic) to see any issues with the timing of the logic sequence more easily than would be possible in other formats.

Programming

Early PLCs, up to the mid-1980s, were programmed using proprietary programming panels or special-purpose programming terminals, which often had dedicated function keys representing the various logical elements of PLC programs. Programs were stored on cassette tape cartridges. Facilities for printing and documentation were very minimal due to lack of memory capacity. The very oldest PLCs used non-volatile magnetic core memory.

More recently, PLCs are programmed using application software on personal computers. The computer is connected to the PLC through Ethernet, RS-232, RS-485 or RS-422 cabling. The programming software allows entry and editing of the ladder-style logic. Generally the software provides functions for debugging and troubleshooting the PLC software, for example, by highlighting portions of the logic to show current status during operation or via simulation. The software will upload and download the PLC program, for backup and restoration purposes. In some models of programmable controller, the program is transferred from a personal computer to the PLC though a programming board which writes the program into a removable chip such as an EEPROM or EPROM.

Functionality

The functionality of the PLC has evolved over the years to include sequential relay control, motion control, process control, distributed control systems and networking. The data handling, storage, processing power and communication capabilities of some modern PLCs are approximately equivalent to desktop computers. PLC-like programming combined with remote I/O hardware, allow a general-purpose desktop

computer to overlap some PLCs in certain applications. Regarding the practicality of these desktop computer based logic controllers, it is important to note that they have not been generally accepted in heavy industry because the desktop computers run on less stable operating systems than do PLCs, and because the desktop computer hardware is typically not designed to the same levels of tolerance to temperature, humidity, vibration, and longevity as the processors used in PLCs. In addition to the hardware limitations of desktop based logic, operating systems such as Windows do not lend themselves to deterministic logic execution, with the result that the logic may not always respond to changes in logic state or input status with the extreme consistency in timing as is expected from PLCs. Still, such desktop logic applications find use in less critical situations, such as laboratory automation and use in small facilities where the application is less demanding and critical, because they are generally much less expensive than PLCs.

In more recent years, small products called PLRs (programmable logic relays), and also by similar names, have become more common and accepted. These are very much like PLCs, and are used in light industry where only a few points of I/O (i.e. a few signals coming in from the real world and a few going out) are involved, and low cost is desired. These small devices are typically made in a common physical size and shape by several manufacturers, and branded by the makers of larger PLCs to fill out their low end product range. Popular names include PICO Controller, NANO PLC, and other names implying very small controllers. Most of these have between 8 and 12 digital inputs, 4 and 8 digital outputs, and up to 2 analog inputs. Size is usually about 4" wide, 3" high, and 3" deep. Most such devices include a tiny postage stamp sized LCD screen for viewing simplified ladder logic (only a very small portion of the program being visible at a given time) and status of I/O points, and typically these screens are accompanied by a 4-way rocker push-button plus four more separate push-buttons, similar to the key buttons on a VCR remote control, and used to navigate and edit the logic. Most have a small plug for connecting via RS-232 or RS-485 to a personal computer so that programmers can use simple Windows applications for programming instead of being forced to use the tiny LCD and push-button set for this purpose. Unlike regular PLCs that are usually modular and greatly expandable, the PLRs are usually not modular or expandable, but their price can be two orders of magnitude less than a PLC and they still offer robust design and deterministic execution of the logic.

PLC Topics

Features

Control panel with PLC (grey elements in the center). The unit consists of separate elements, from left to right; power supply, controller, relay units for in- and output

The main difference from other computers is that PLCs are armored for severe conditions (such as dust, moisture, heat, cold) and have the facility for extensive input/output (I/O) arrangements. These connect the PLC to sensors and actuators. PLCs read limit switches, analog process variables (such as temperature and pressure), and the positions of complex positioning systems. Some use machine vision. On the actuator side, PLCs operate electric motors, pneumatic or hydraulic cylinders, magnetic relays, solenoids, or analog

outputs. The input/output arrangements may be built into a simple PLC, or the PLC may have external I/O modules attached to a computer network that plugs into the PLC.

System scale

A small PLC will have a fixed number of connections built in for inputs and outputs. Typically, expansions are available if the base model has insufficient I/O.

Modular PLCs have a chassis (also called a rack) into which are placed modules with different functions. The processor and selection of I/O modules is customised for the particular application. Several racks can be administered by a single processor, and may have thousands of inputs and outputs. A special high speed serial I/O link is used so that racks can be distributed away from the processor, reducing the wiring costs for large plants.

User interface

PLCs may need to interact with people for the purpose of configuration, alarm reporting or everyday control.

A Human-Machine Interface (HMI) is employed for this purpose. HMIs are also referred to as MMIs (Man Machine Interface) and GUIs (Graphical User Interface).

A simple system may use buttons and lights to interact with the user. Text displays are available as well as graphical touch screens. More complex systems use programming and monitoring software installed on a computer, with the PLC connected via a communication interface.

Communications

PLCs have built in communications ports, usually 9-pin RS-232, but optionally EIA-485 or Ethernet. Modbus, BACnet or DF1 is usually included as one of the communications protocols. Other options include various fieldbuses such as DeviceNet or Profibus. Other communications protocols that may be used are listed in the List of automation protocols.

Most modern PLCs can communicate over a network to some other system, such as a computer running a SCADA (Supervisory Control And Data Acquisition) system or web browser.

PLCs used in larger I/O systems may have peer-to-peer (P2P) communication between processors. This allows separate parts of a complex process to have individual control while allowing the subsystems to co-ordinate over the communication link. These communication links are also often used for HMI devices such as keypads or PC-type workstations.

Programming

PLC programs are typically written in a special application on a personal computer, then downloaded by a direct-connection cable or over a network to the PLC. The program is stored in the PLC either in battery-backed-up RAM or some other non-volatile flash memory. Often, a single PLC can be programmed to replace thousands of relays.

Under the IEC 61131-3 standard, PLCs can be programmed using standards-based programming languages. A graphical programming notation called Sequential Function Charts is available on certain programmable controllers. Initially most PLCs utilized Ladder Logic Diagram Programming, a model which emulated electromechanical control panel devices (such as the contact and coils of relays) which PLCs replaced. This model remains common today.

IEC 61131-3 currently defines five programming languages for programmable control systems: FBD (Function block diagram), LD (Ladder diagram), ST (Structured text, similar to the Pascal programming language), IL (Instruction list, similar to assembly language) and SFC (Sequential function chart). These techniques emphasize logical organization of operations.

While the fundamental concepts of PLC programming are common to all manufacturers, differences in I/O addressing, memory organization and instruction sets mean that PLC programs are never perfectly interchangeable between different makers. Even within the same product line of a single manufacturer, different models may not be directly compatible.

PLC compared with other control systems

Allen-Bradley PLC installed in a control panel

PLCs are well-adapted to a range of automation tasks. These are typically industrial processes in manufacturing where the cost of developing and maintaining the automation system is high relative to the total cost of the automation, and where changes to the system would be expected during its operational life. PLCs contain input and output devices compatible with industrial pilot devices and controls; little electrical design is required, and the design problem centers on expressing the desired sequence of operations. PLC applications are typically highly customized systems so the cost of a packaged PLC is low compared to the cost of a specific custom-built controller design. On the other hand, in the case of mass-produced goods, customized control systems are economic due to the lower cost of the components, which can be optimally chosen instead of a "generic" solution, and where the non-recurring engineering charges are spread over thousands or millions of units.

For high volume or very simple fixed automation tasks, different techniques are used. For example, a consumer dishwasher would be controlled by an electromechanical cam timer costing only a few dollars in production quantities.

A microcontroller-based design would be appropriate where hundreds or thousands of units will be produced and so the development cost (design of power supplies, input/output hardware and necessary testing and certification) can be spread over many sales, and where the end-user would not need to alter the control. Automotive

applications are an example; millions of units are built each year, and very few end-users alter the programming of these controllers. However, some specialty vehicles such as transit busses economically use PLCs instead of custom-designed controls, because the volumes are low and the development cost would be uneconomic.

Very complex process control, such as used in the chemical industry, may require algorithms and performance beyond the capability of even high-performance PLCs. Very high-speed or precision controls may also require customized solutions; for example, aircraft flight controls.

Programmable controllers are widely used in motion control, positioning control and torque control. Some manufacturers produce motion control units to be integrated with PLC so that G-code (involving a CNC machine) can be used to instruct machine movements.

PLCs may include logic for single-variable feedback analog control loop, a "proportional, integral, derivative" or "PID controller". A PID loop could be used to control the temperature of a manufacturing process, for example. Historically PLCs were usually configured with only a few analog control loops; where processes required hundreds or thousands of loops, a distributed control system (DCS) would instead be used. As PLCs have become more powerful, the boundary between DCS and PLC applications has become less distinct.

PLCs have similar functionality as Remote Terminal Units. An RTU, however, usually does not support control algorithms or control loops. As hardware rapidly becomes more powerful and cheaper, RTUs, PLCs and DCSs are increasingly beginning to overlap in responsibilities, and many vendors sell RTUs with PLC-like features and vice versa. The industry has standardized on the IEC 61131-3 functional block language for creating programs to run on RTUs and PLCs, although nearly all vendors also offer proprietary alternatives and associated development environments.

Digital and analog signals

Digital or discrete signals behave as binary switches, yielding simply an On or Off signal (1 or 0, True or False, respectively). Push buttons, limit switches, and photoelectric sensors are examples of devices providing a discrete signal. Discrete signals are sent using either voltage or current, where a specific range is designated as *On* and another as *Off*. For example, a PLC might use 24 V DC I/O, with values above 22 V DC representing *On*, values below 2VDC representing *Off*, and intermediate values undefined. Initially, PLCs had only discrete I/O.

Analog signals are like volume controls, with a range of values between zero and full-scale. These are typically interpreted as integer values (counts) by the PLC, with various ranges of accuracy depending on the device and the number of bits available to store the data. As PLCs typically use 16-bit signed binary processors, the integer values are limited between -32,768 and +32,767. Pressure, temperature, flow, and weight are often

represented by analog signals. Analog signals can use voltage or current with a magnitude proportional to the value of the process signal. For example, an analog 0 - 10 V input or 4-20 mA would be converted into an integer value of 0 - 32767.

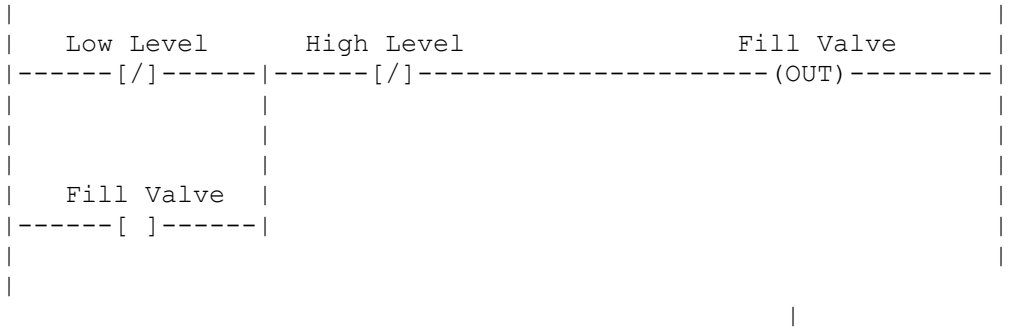
Current inputs are less sensitive to electrical noise (i.e. from welders or electric motor starts) than voltage inputs.

Example

As an example, say a facility needs to store water in a tank. The water is drawn from the tank by another system, as needed, and our example system must manage the water level in the tank.

Using only digital signals, the PLC has two digital inputs from float switches (Low Level and High Level). When the water level is above the switch it closes a contact and passes a signal to an input. The PLC uses a digital output to open and close the inlet valve into the tank.

When the water level drops enough so that the Low Level float switch is off (down), the PLC will open the valve to let more water in. Once the water level rises enough so that the High Level switch is on (up), the PLC will shut the inlet to stop the water from overflowing. This rung is an example of seal-in (latching) logic. The output is sealed in until some condition breaks the circuit.



An analog system might use a water pressure sensor or a load cell, and an adjustable (throttling) dripping out of the tank, the valve adjusts to slowly drip water back into the tank.

In this system, to avoid 'flutter' adjustments that can wear out the valve, many PLCs incorporate "hysteresis" which essentially c

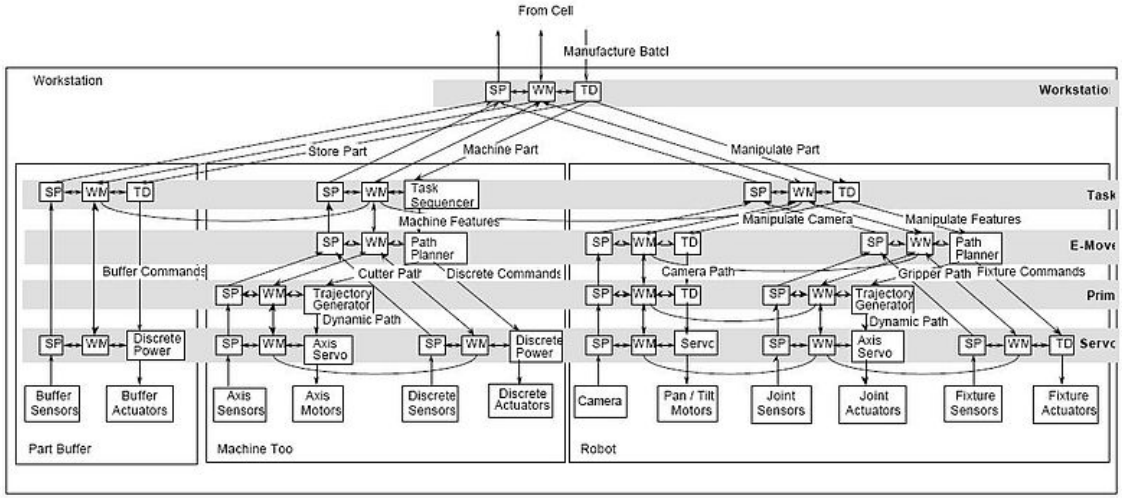
reates a "deadband" of activity. A technician adjusts this deadband so the valve moves only for a significant change in rate. This will in turn minimize the motion of the valve, and reduce its wear.

A real system might combine both approaches, using float switches and simple valves to prevent spills, and a rate sensor and rate valve to optimize refill rates and prevent water hammer. Backup and maintenance methods can make a real system very complicated.

Chapter 8

Real-Time Control System

Real-time Control System (RCS) is a Reference Model Architecture, suitable for many software-intensive, real-time control problem domains. RCS is a reference model architecture that defines the types of functions that are required in a real-time intelligent control system, and how these functions are related to each other.



Example of a RCS-3 application of a machining workstation containing a machine tool, part buffer, and robot with vision system. RCS-3 produces a layered graph of processing nodes, each of which contains a Task Decomposition (TD), World Modeling (WM), and Sensory Processing (SP) module. These modules are richly interconnected to each other by a communications system.

RCS is not a system design, nor is it a specification of how to implement specific systems. RCS prescribes a hierarchical control model based on a set of well-founded engineering principles to organize system complexity. All the control nodes at all levels share a generic node model.

Also RCS provides a comprehensive methodology for designing, engineering, integrating, and testing control systems. Architects iteratively partition system tasks and information into finer, finite subsets that are controllable and efficient. RCS focuses on intelligent control that adapts to uncertain and unstructured operating environments. The key concerns are sensing, perception, knowledge, costs, learning, planning, and execution.

Overview

A reference model architecture is a canonical form, not a system design specification. The RCS reference model architecture combines real-time motion planning and control with high level task planning, problem solving, world modeling, recursive state estimation, tactile and visual image processing, and acoustic signature analysis. In fact, the evolution of the RCS concept has been driven by an effort to include the best properties and capabilities of most, if not all, the intelligent control systems currently known in the literature, from subsumption to SOAR, from blackboards to object-oriented programming.

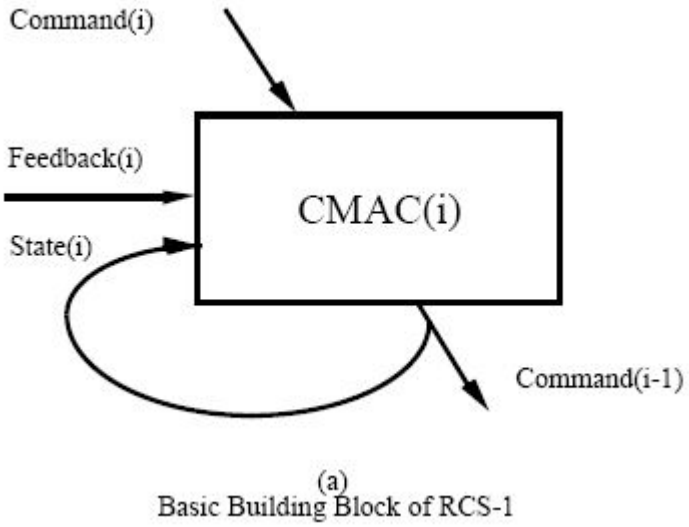
RCS (Real-time Control System) is developed into an intelligent agent architecture designed to enable any level of intelligent behavior, up to and including human levels of performance. RCS was inspired 30 years ago by a theoretical model of the cerebellum, the portion of the brain responsible for fine motor coordination and control of conscious motions. It was originally designed for sensory-interactive goal-directed control of laboratory manipulators. Over three decades, it has evolved into a real-time control architecture for intelligent machine tools, factory automation systems, and intelligent autonomous vehicles.

RCS applies to many problem domains including Manufacturing examples and Vehicle systems examples. Systems based on the RCS architecture have been designed and implemented to varying degrees for a wide variety of applications that include loading and unloading of parts and tools in machine tools, controlling machining workstations, performing robotic deburring and chamfering, and controlling space station telerobots, multiple autonomous undersea vehicles, unmanned land vehicles, coal mining automation systems, postal service mail handling systems, and submarine operational automation systems.

History

RCS has evolved through a variety of versions over a number of years as understanding of the complexity and sophistication of intelligent behavior has increased. The first implementation was designed for sensory-interactive robotics by Barbera in the mid 1970’s.

RCS-1



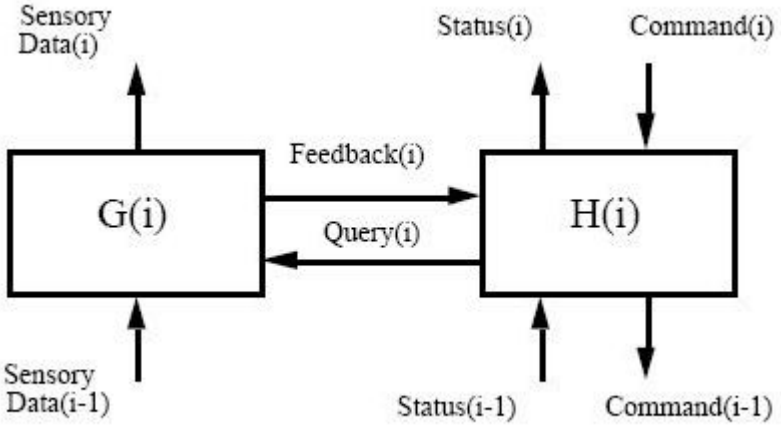
Basics of the RCS-1 control paradigm.

In RCS-1, the emphasis was on combining commands with sensory feedback so as to compute the proper response to every combination of goals and states. The application was to control a robot arm with a structured light vision system in visual pursuit tasks. RCS-1 was heavily influenced by biological models such as the Marr-Albus model, and the Cerebellar Model Arithmetic Computer (CMAC). of the cerebellum.

CMAC becomes a state machine when some of its outputs are fed directly back to the input, so RCS-1 was implemented as a set of state-machines arranged in a hierarchy of control levels. At each level, the input command effectively selects a behavior that is driven by feedback in stimulus-response fashion. CMAC thus became the reference model building block of RCS-1, as shown in the figure.

A hierarchy of these building blocks was used to implement a hierarchy of behaviors such as observed by Tinbergen and others. RCS-1 is similar in many respects to Brooks' subsumption architecture, except that RCS selects behaviors before the fact through goals expressed in commands, rather than after the fact through subsumption.

RCS-2



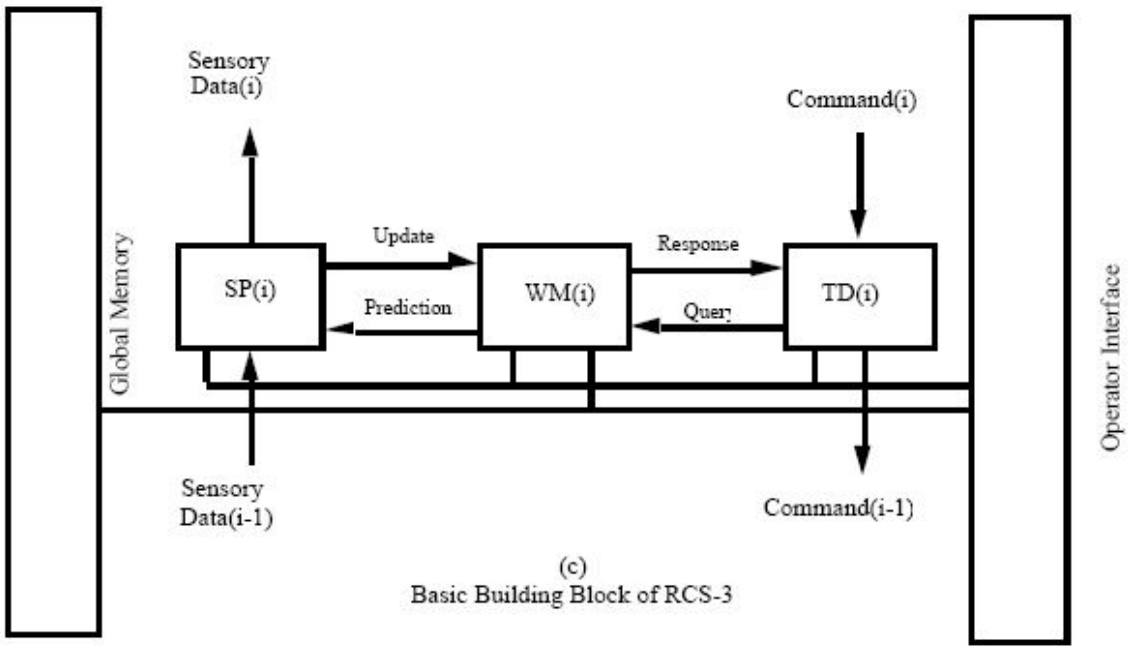
RCS-2 control paradigm.

The next generation, RCS-2, was developed by Barbera, Fitzgerald, Kent, and others for manufacturing control in the NIST Automated Manufacturing Research Facility (AMRF) during the early 1980's The basic building block of RCS-2 is shown in the figure.

The H function remained a finite state machine state-table executor. The new feature of RCS-2 was the inclusion of the G function consisting of a number of sensory processing algorithms including structured light and blob analysis algorithms. RCS-2 was used to define an eight level hierarchy consisting of Servo, Coordinate Transform, E-Move, Task, Workstation, Cell, Shop, and Facility levels of control.

Only the first six levels were actually built. Two of the AMRF workstations fully implemented five levels of RCS-2. The control system for the Army Field Material Handling Robot (FMR) was also implemented in RCS-2, as was the Army TMAP semi-autonomous land vehicle project.

RCS-3

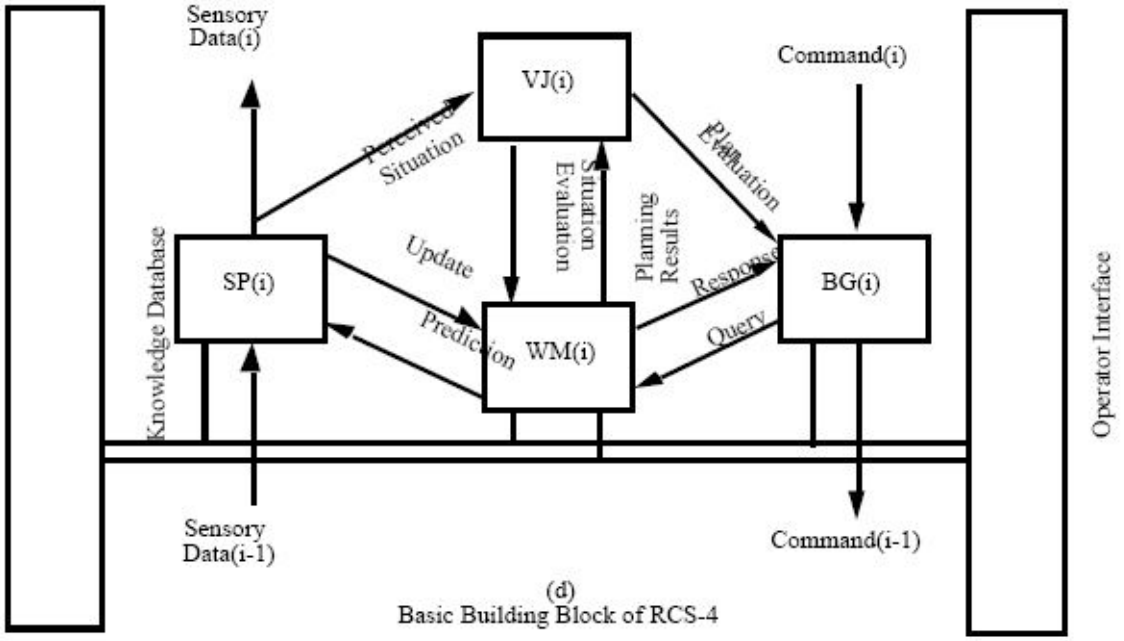


RCS-3 control paradigm.

RCS-3 was designed for the NBS/DARPA Multiple Autonomous Undersea Vehicle (MAUV) project and was adapted for the NASA/NBS Standard Reference Model Telerobot Control System Architecture (NASREM) developed for the space station Flight Telerobotic Servicer The basic building block of RCS-3 is shown in the figure.

The principal new features introduced in RCS-3 are the World Model and the operator interface. The inclusion of the World Model provides the basis for task planning and for model-based sensory processing. This led to refinement of the task decomposition (TD) modules so that each have a job assigner, and planner and executor for each of the subsystems assigned a job. This corresponds roughly to Saridis' three level control hierarchy.

RCS-4



RCS-4 control paradigm.

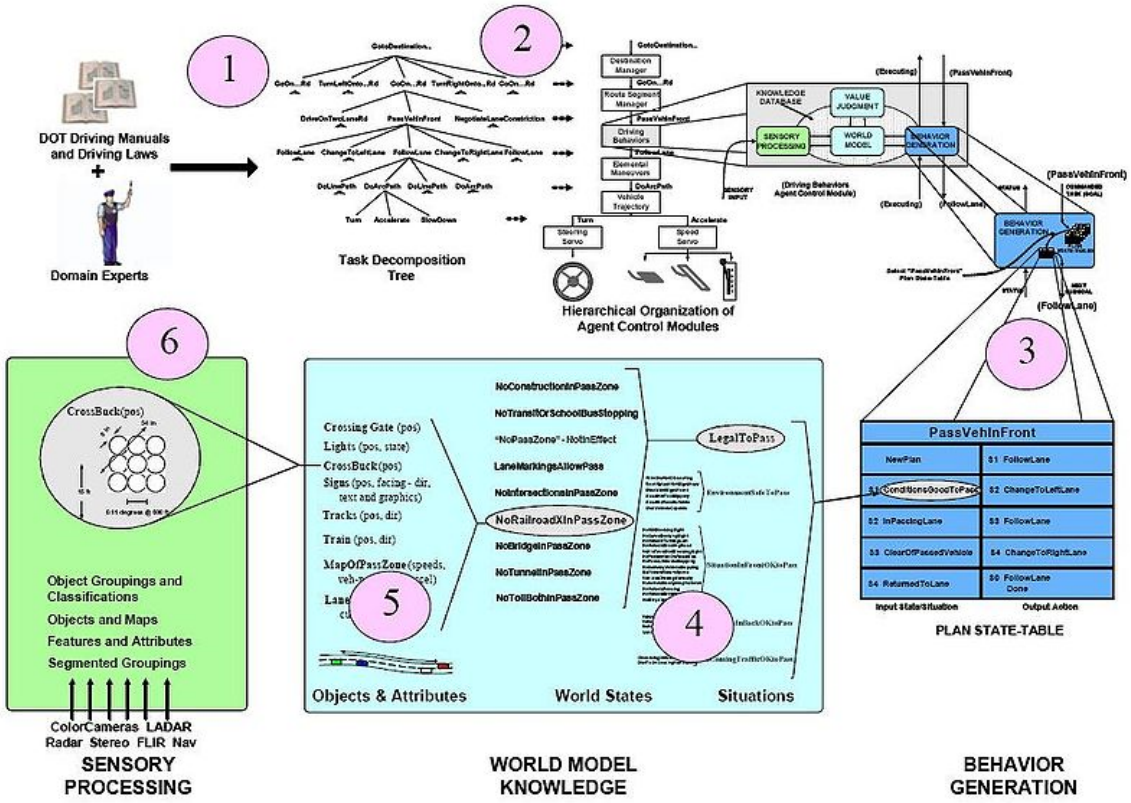
RCS-4 is developed since the 1990s by the NIST Robot Systems Division. The basic building block is shown in the figure). The principal new feature in RCS-4 is the explicit representation of the Value Judgment (VJ) system. VJ modules provide to the RCS-4 control system the type of functions provided to the biological brain by the limbic system. The VJ modules contain processes that compute cost, benefit, and risk of planned actions, and that place value on objects, materials, territory, situations, events, and outcomes. Value state-variables define what goals are important and what objects or regions should be attended to, attacked, defended, assisted, or otherwise acted upon. Value judgments, or evaluation functions, are an essential part of any form of planning or learning. The application of value judgments to intelligent control systems has been addressed by George Pugh. The structure and function of VJ modules are developed more completely developed in Albus (1991).

RCS-4 also uses the term behavior generation (BG) in place of the RCS-3 term task 5 decomposition (TD). The purpose of this change is to emphasize the degree of autonomous decision making. RCS-4 is designed to address highly autonomous applications in unstructured environments where high bandwidth communications are impossible, such as unmanned vehicles operating on the battlefield, deep undersea, or on distant planets. These applications require autonomous value judgments and sophisticated real-time perceptual capabilities. RCS-3 will continue to be used for less demanding applications, such as manufacturing, construction, or telerobotics for near-space, or shallow undersea operations, where environments are more structured and communication bandwidth to a human interface is less restricted. In these applications,

value judgments are often represented implicitly in task planning processes, or in human operator input.

RCS Methodology

In the figure, an example of the RCS methodology for designing a control system for autonomous onroad driving under everyday traffic conditions is summarized in six steps.



The six steps of the RCS methodology for knowledge acquisition and representation.

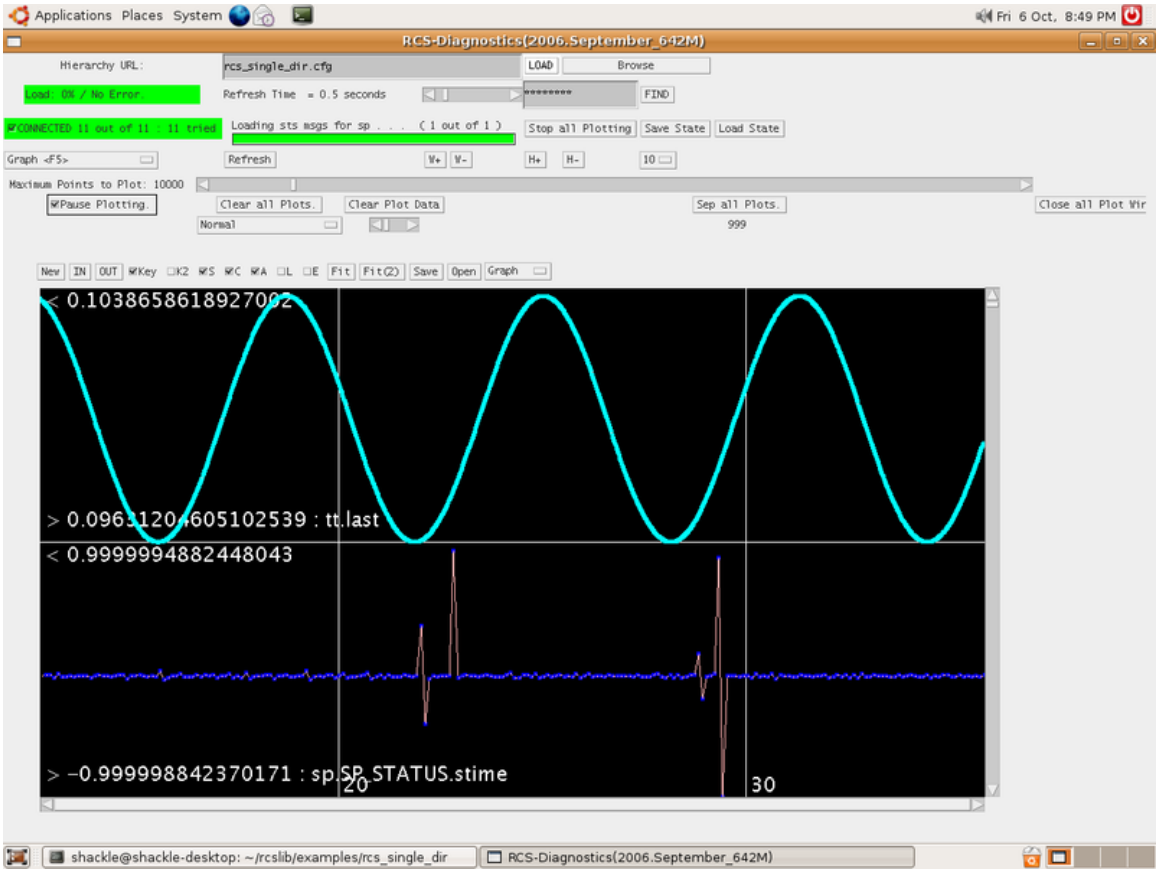
- Step 1 consists of an intensive analysis of domain knowledge from training manuals and subject matter experts. Scenarios are developed and analyzed for each task and subtask. The result of this step is a structuring of procedural knowledge into a task decomposition tree with simpler and simpler tasks at each echelon. At each echelon, a vocabulary of commands (action verbs with goal states, parameters, and constraints) is defined to evoke task behavior at each echelon.
- Step 2 defines a hierarchical structure of organizational units that will execute the commands defined in step 1. For each unit, its duties and responsibilities in response to each command are specified. This is analogous to establishing a work breakdown structure for a development project, or defining an organizational chart for a business or military operation.

- Step 3 specifies the processing that is triggered within each unit upon receipt of an input command. For each input command, a state-graph (or statetable or extended finite state automaton) is defined that provides a plan (or procedure for making a plan) for accomplishing the commanded task. The input command selects (or causes to be generated) an appropriate state-table, the execution of which generates a series of output commands to units at the next lower echelon. The library of state-tables contains a set of statesensitive procedural rules that identify all the task branching conditions and specify the corresponding state transition and output command parameters.

The result of step 3 is that each organizational unit has for each input command a state-table of ordered production rules, each suitable for execution by an extended finite state automaton (FSA). The sequence of output subcommands required to accomplish the input command is generated by situations (i.e., branching conditions) that cause the FSA to transition from one output subcommand to the next.

- In step 4, each of the situations that are defined in step 3 are analyzed to reveal their dependencies on world and task states. This step identifies the detailed relationships between entities, events, and states of the world that cause a particular situation to be true.
- In step 5, we identify and name all of the objects and entities together with their particular features and attributes that are relevant to detecting the above world states and situations.
- In step 6, we use the context of the particular task activities to establish the distances and, therefore, the resolutions at which the relevant objects and entities must be measured and recognized by the sensory processing component. This establishes a set of requirements and/or specifications for the sensor system to support each subtask activity.

Real-time Control System Software



Real-Time Control Systems Software.

Based on the RCS Reference Model Architecture the NIST has developed a Real-time Control System Software Library. This is an archive of free C++, Java and Ada code, scripts, tools, makefiles, and documentation developed to aid programmers of software to be used in real-time control systems, especially those using the Reference Model Architecture for Intelligent Systems Design.

Applications

- The ISAM Framework is an RCS application to the Manufacturing Domain.
- The 4D-RCS Reference Model Architecture is the RCS application to the Vehicle Domain, and
- The NASA/NBS Standard Reference Model for Telerobot Control Systems Architecture (NASREM) is an application to the Space Domain.

Chapter 9

Industrial Ethernet & ORiN

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			
	Versatile **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.)

ORiN

ORiN (Open Robot/Resource interface for the Network) is a standard network interface for FA (factory automation) systems. Japan Robot Association proposed ORiN in 2002, and ORiN Forum develops and maintenances the ORiN standard.

Backgraund

The installation of PC (Personal Computer) applications in the factory is increasing recently. Various types of application software systems, such as production management system, process management system, operation monitoring system and failure analysis system, are operating in the factory. These software systems are becoming indispensable for the manufacturing system.

However, most of these software systems are only compatible with specific models or specific manufacturers of the FA system. This is because the software system is “custom made” depending on the specific special network or protocol. Once this type of application is installed in a factory and if there are no resident software engineers for the system, the improvement of the system will stop, the cost-effectiveness of the system will be worsen, and the total value of the system will deteriorate.

Another recent problem in production is the rapid increase of the product demand at the initial stage of the product release. The manufactures will lose the chance of possible profit if they cannot meet the demand. To cope with the problem, manufacturing industry is trying to achieve the vertical startup of the production, and high reusability of both hardware and software is the key for the goal.

To solve these problems, ORiN was developed as a standard PC application platform.

Outline

ORiN was originally developed as a standard platform for robot applications. Nowadays, ORiN became a manufacturing application program platform for handling wider range of resources including robots and other FA devices like programmable logic controllers (PLC) and numerical control (NC) systems, or more generic resources like databases and local file systems. ORiN specifications are on software only and are independent from hardware. Therefore, ORiN can be smoothly integrated with other existing technologies only by developing software. By using ORiN, development of manufacture-independent and model-independent application becomes easy.

By utilizing ORiN, various application software development and active multi-vender system construction by third-party companies are expected. In addition, on economy side, increase of manufacturing competitiveness, expansion of FA market, advancement of software industry in FA, and creation of FA engineering industry are also expected.

Features

ORiN is independent from hardware, and all ORiN specifications are for software. ORiN (Version 2) is composed of the following three key technology specifications.

1. CAO (Controller Access Object), standard program interface specifications : Specifications to facilitate generalization of application software
2. CRD (Controller Resource Definition), standard data schema specifications : Specifications to facilitate data exchange between application software
3. CAP (Controller Access Protocol), standard communication protocol : Protocol for communication between FA devices and applications Three types of CAP are defined: CAP (SOAP), e-CAP (HTTP), b-CAP (TCP/UDP).

With these three key standard technologies, ORiN provides following features.

- Unified accessing model and data representation
- Variable and file based access to the resources in the device
- Applicable to various devices in the factory
- No device modification is required for ORiN connection
- XML data representation to cooperate with other systems
- Easy device access over Internet with simple parameter setup
- Configurable application interface

History

- ORiN project started as a part of standardization activities in Japan Robot Association (JARA).
- With the support from New Energy and Industrial Technology Development Organization (NEDO), ORiN system was developed from 1999 (3years activity).
- Member companies of JARA participated field robot connection test at International Robot Exhibition (IREX) in 1999 and 2001. ORiN Version 1.0 was formally created in 2002.
- ORiN Forum was created in 2002 to promote and advance the ORiN standard. (Key members of the Forum: FA device manufacture, software development company, system integrator company, etc.)
- ORiN Forum member companies participated various ORiN field application tests for three years. The test results were reflected to the ORiN Version 2.0 (created in 2005). ORiN2 SDK was released as a supported software product in 2005.
- ORiN application was proposed as an annex of ISO20242-Part4. The DIS of the standard was approved in July 2010.

ORiN realted terms

ORiN SDK

ORiN SDK is a software development kit for ORiN Version 1.0, including RAO, standard providers and development tools. The SDK is used to develop original RAO providers and ORiN applications, and the SDK is also used as ORiN

execution environment. ORiN Forum distributes the SDK, but the Forum plans to stop the distribution and technical support of the SDK at the end of March 2011.

ORiN2 SDK

ORiN 2 SDK is a software development kit for ORiN Version 2.0. The SDK provides standard interface specifications for applications and devices, standard data schema, and standard communication protocol. Development of provider module (extension module) based on the specifications is also possible with the SDK. DENSO WAVE INCORPORATED distributes and supports the SDK.

Chapter 10

SERCOS III

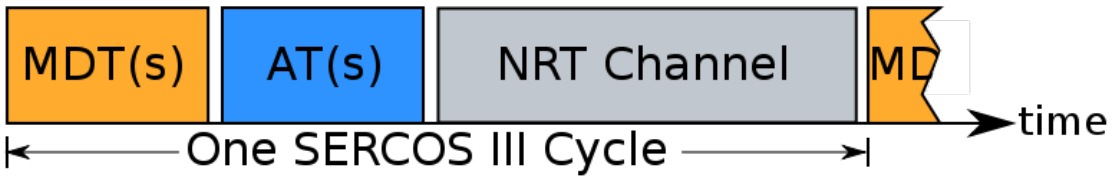
SERCOS III is the third generation of the SERCOS interface, a globally standardized open digital interface for the communication between industrial controls, motion devices, and input/output devices (I/O). SERCOS III merges the hard real-time aspects of the SERCOS interface with Ethernet. It is based 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

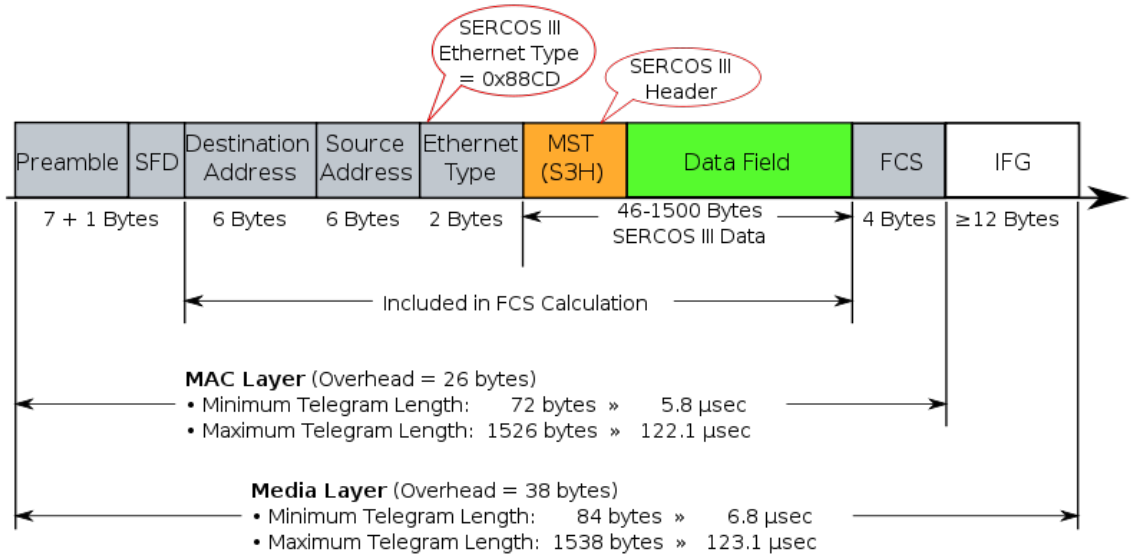


The basic SERCOS III cycle

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 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.

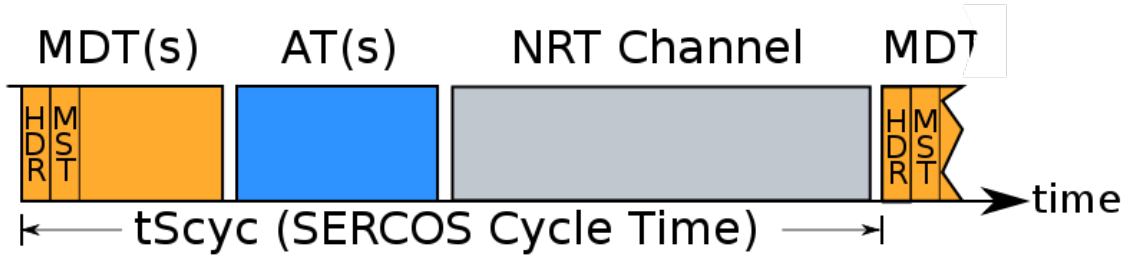
Ethernet type
: A unique EtherType value has been assigned via the IEEE EtherType Field Registration Authority for SERCOS III (0x88CD).

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

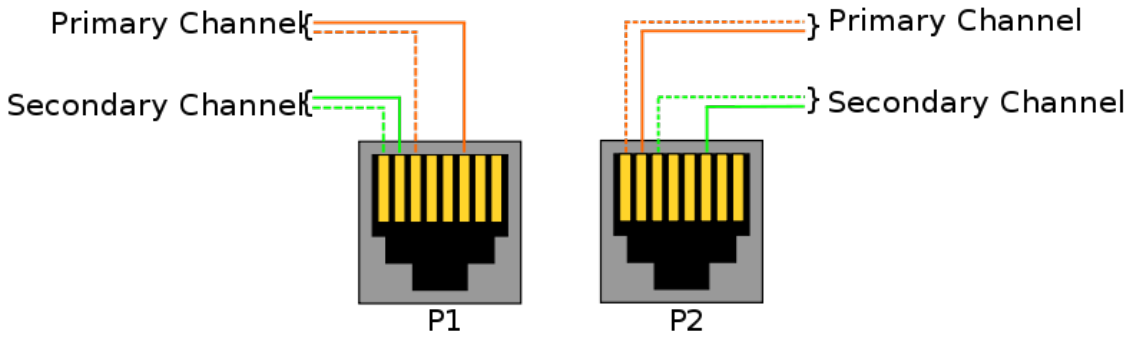
Two main types of telegrams are used within the SERCOS III Cycle. The Master Data Telegram (MDT), and the Acknowledge telegram (AT). Both telegram types are issued by the Master (control). The MDT contains information provided by the Master to Slaves. It is filled by the Master, and read by Slaves. The 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.

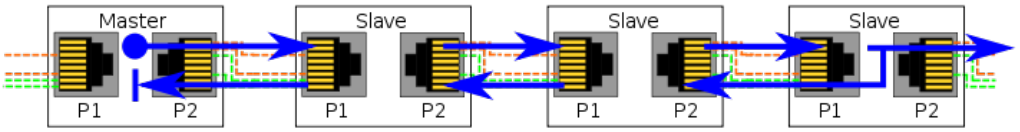
SERCOS III address
: Each SERCOS III device contains a numeric address used by other devices on the SERCOS III network to exchange data. The address may be any whole integer from 1 to 511.

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

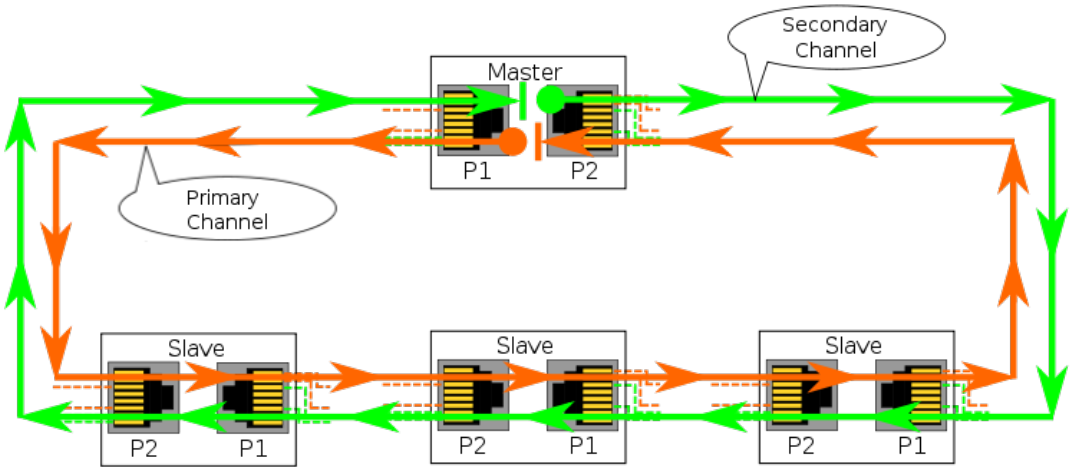
The SERCOS III specification defines two possible network topologies; Ring and Line. To those familiar with other networks, they may appear to both be configured as a Ring. All telegrams begin and end at the Master. The Full Duplex feature of the physical layer is used to achieve this.

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 (see below), 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 (see below).

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.

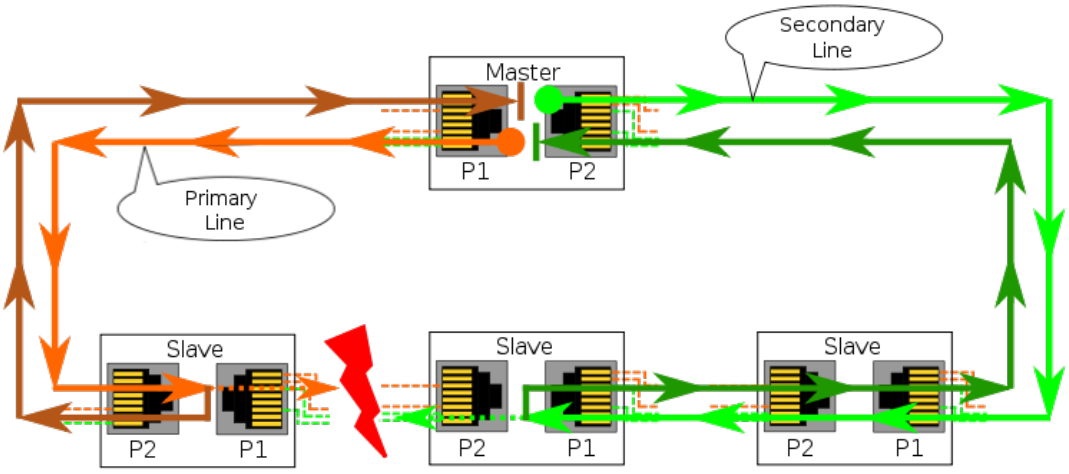
Infrastructure hardware

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.

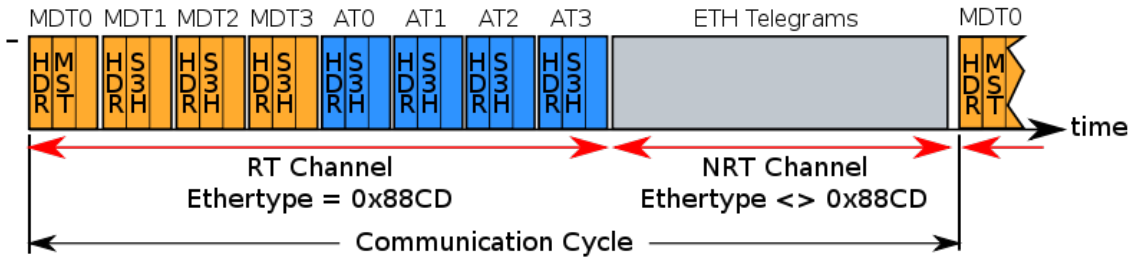
Non-real-time (NRT) channel

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.

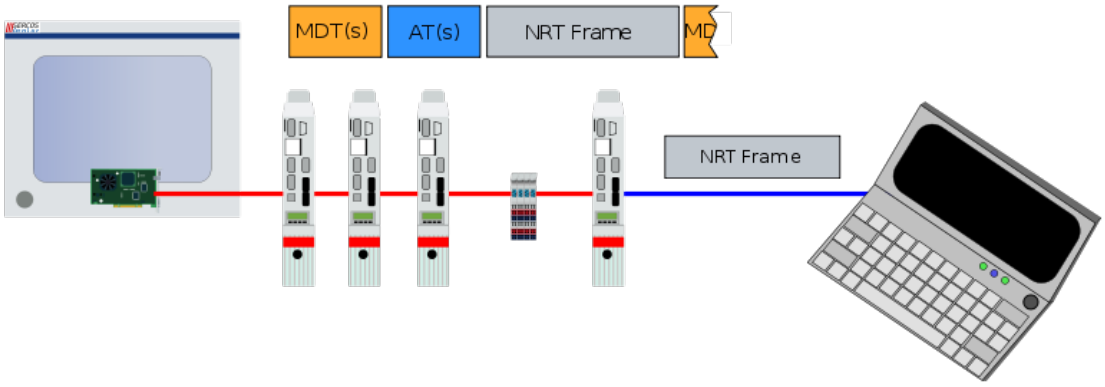
A SERCOS III network will always pass NRT frames, even when cyclic operation has not been initialized. This means that devices always have access to the network for NRT messages, as long as the ports are powered.



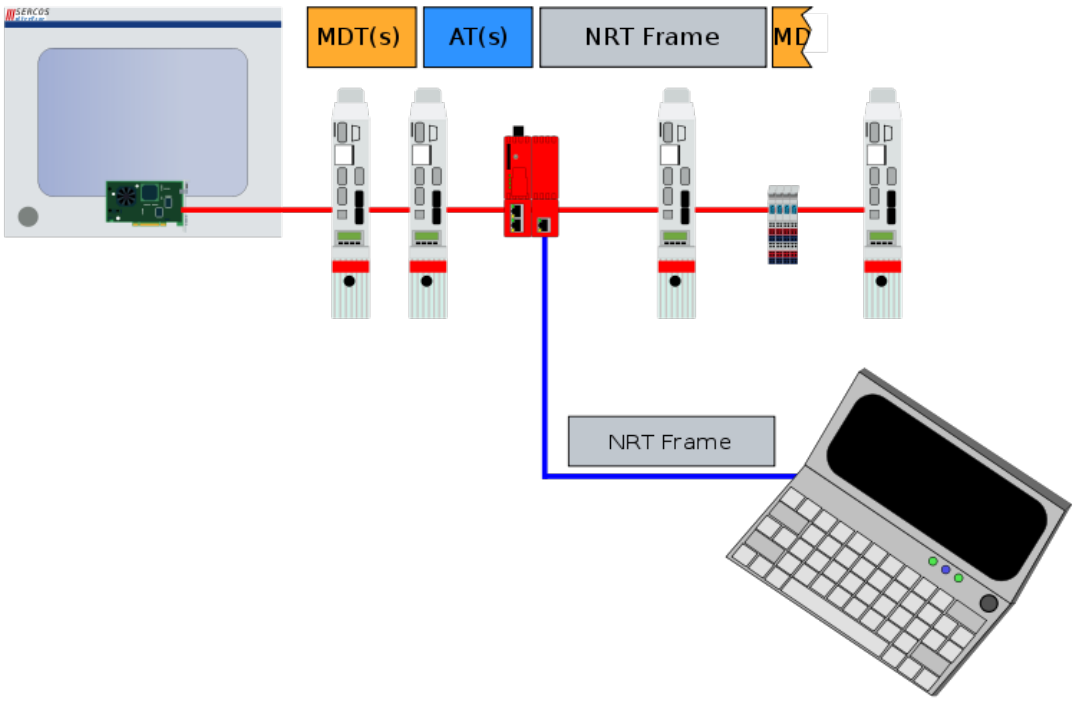
NRT Channel

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.

Additional resources

Development tools

A number of different manufacturers provide tools to aid in the design of SERCOS III compliant network nodes. Since SERCOS III uses standard Ethernet frames, most off-the-shelf network tools can be used on SERCOS III networks. For example, a driver is available for the free Wireshark network protocol analyzer.

Open source driver

On April 21, 2009, SERCOS International and SERCOS North America announced a cooperation with the Open Source Automation Development Lab eG (OSADL) to provide an open source software driver library for SERCOS III. The library simplifies the development of a SERCOS Master node.

Chapter 11

Profibus

Profibus electrical connector

PROFIBUS (Process Field Bus) is a standard for field bus communication in automation technology and was first promoted (1989) by BMBF (German department of education

and research). It should not be confused with the PROFINET standard for Industrial Ethernet.

Origin

The history of PROFIBUS goes back to a publicly promoted plan for an association started in Germany in 1987 and for which 21 companies and institutes devised a master project plan called "field bus". The goal was to implement and spread the use of a bit-serial field bus based on the basic requirements of the field device interfaces. For this purpose, member companies agreed to support a common technical concept for production (i.e discrete or factory automation) and process automation. First, the complex communication protocol Profibus FMS (Field bus Message Specification), which was tailored for demanding communication tasks, was specified. Subsequently in 1993, the specification for the simpler and thus considerably faster protocol PROFIBUS DP (Decentralized Peripherals) was completed. Profibus FMS is used for (non deterministic) communication of data between Profibus Masters. Profibus DP is a protocol made for (deterministic) communication between Profibus masters and their remote I/O slaves.

There are two variations of PROFIBUS in use today; the most commonly used PROFIBUS DP, and the lesser used, application specific, PROFIBUS PA:

- **PROFIBUS DP** (Decentralized Peripherals) is used to operate sensors and actuators via a centralized controller in production (factory) automation applications. The many standard diagnostic options, in particular, are focused on here.
- **PROFIBUS PA** (Process Automation) is used to monitor measuring equipment via a process control system in process automation applications. This variant is designed for use in explosion/hazardous areas (Ex-zone 0 and 1). The Physical Layer (i.e. the cable) conforms to IEC 61158-2, which allows power to be delivered over the bus to field instruments, while limiting current flows so that explosive conditions are not created, even if a malfunction occurs. The number of devices attached to a PA segment is limited by this feature. PA has a data transmission rate of 31.25 kbit/s. However, PA uses the same protocol as DP, and can be linked to a DP network using a coupler device. The much faster DP acts as a backbone network for transmitting process signals to the controller. This means that DP and PA can work tightly together, especially in hybrid applications where process and factory automation networks operate side by side.

In excess of 30 million PROFIBUS nodes were installed by the end of 2009. 5 million of these are in the process industries.

Technology

PROFIBUS Protocol (OSI reference model)

OSI-Layer		PROFIBUS			
7	Application	DPV0	DPV1	DPV2	Management
6	Presentation	--			
5	Session				
4	Transport				
3	Network				
2	Data Link	FDL			
1	Physical	EIA-485	Optical	MBP	

Application layer

To utilize these functions, various service levels of the DP protocol were defined:

- DP-V0 for cyclic exchange of data and diagnosis
- DP-V1 for acyclic and cyclic data exchange and alarm handling
- DP-V2 for isochronous mode and data exchange broadcast (slave-to-slave communication)

Security layer

The security layer **FDL** (Field bus Data Link) works with a hybrid access method that combines token passing with a master-slave method. In a PROFIBUS DP network, the controllers or process control systems are the masters and the sensors and actuators are the slaves.

Various telegram types are used. They can be differentiated by their start delimiter (SD):

No data: SD1 = 0x10

SD1	DA	SA	FC	FCS	ED

Variable length data:

SD2 = 0x68

SD2	LE	LEr	SD2	DA	SA	FC	DSAP	SSAP	PDU	FCS	ED

Fixed length data:

SD3 = 0xA2

SD3	DA	SA	FC	PDU	FCS	ED

Token:

SD4 = 0xDC

SD4	DA	SA	ED

Brief acknowledgement:

SC = 0xE5

SC

SD: Start Delimiter

LE: Length of protocol data unit, (incl. DA,SA,FC,DSAP,SSAP)

LEr: Repetition of protocol data unit, (Hamming distance = 4)

FC: Function Code

DA: Destination Address

SA: Source Address

DSAP: Destination Service Access Point

SSAP: Source Service Access Point

SAP (Decimal)	**SERVICE**
Default 0	Cyclical Data Exchange (Write_Read_Data)
54	Master-to-Master SAP (M-M Communication)
55	Change Station Address (Set_Slave_Add)
56	Read Inputs (Rd_Inp)
57	Read Outputs (Rd_Outp)
58	Control Commands to a DP Slave (Global_Control)
59	Read Configuration Data (Get_Cfg)

60	Read Diagnostic Data (Slave_Diagnosis)
61	Send Parameterization Data (Set_Prm)
62	Check Configuration Data (Chk_Cfg)

Note: SAP55 is optional and may be disabled if the slave doesn't provide non-volatile storage memory for the station address.

PDU: Protocol Data Unit (protocol data)

FCS: Frame Checking Sequence

ED: End Delimiter (= 0x16 !)

The FCS is calculated by simply adding up the bytes within the specified length. An overflow is ignored here. Each byte is saved with an even parity and transferred asynchronously with a start and stop bit. There may not be a pause between a stop bit and the following start bit when the bytes of a telegram are transmitted. The master signals the start of a new telegram with a SYN pause of at least 33 bits (logical "1" = bus idle).

Bit-transmission layer

Three different methods are specified for the bit-transmission layer:

- With electrical transmission pursuant to EIA-485, twisted pair cables with impedances of 150 ohms are used in a bus topology. Bit rates from 9.6 kbit/s to 12 Mbit/s can be used. The cable length between two repeaters is limited from 100 to 1200 m, depending on the bit rate used. This transmission method is primarily used with PROFIBUS DP.
- With optical transmission via fiber optics, star-, bus- and ring-topologies are used. The distance between the repeaters can be up to 15 km. The ring topology can also be executed redundantly.
- With MBP (Manchester Bus Powered) transmission technology, data and field bus power are fed through the same cable. The power can be reduced in such a way that use in explosion-hazardous environments is possible. The bus topology can be up to 1900 m long and permits branching to field devices (max. 60 m branches). The bit rate here is a fixed 31.25 kbit/s. This technology was specially established for use in process automation for PROFIBUS PA.

For data transfer via sliding contacts for mobile devices or optical or radio data transmission in open spaces, products from various manufacturers can be obtained, however they do not conform to any standard.

Profiles

Profiles are pre-defined configurations of the functions and features available from PROFIBUS 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 also encourages competition that drives vendors towards enhanced performance and lower costs.

There are PROFIBUS profiles for Encoders, Laboratory instruments, Intelligent Pumps, Robots and Numerically Controlled machines, for example. Profiles also exist for applications such as using HART and wireless with PROFIBUS, and process automation devices via PROFIBUS PA. Other profiles have been specified for Motion Control (PROFIdrive) and Functional Safety (PROFIsafe).

Standardization

PROFIBUS was defined in 1991/1993 in DIN 19245, was then included in EN 50170 in 1996 and, since 1999, established in IEC 61158/IEC 61784.

Organization

The *PROFIBUS Nutzerorganisation e.V.* (PROFIBUS User Organization, or PNO) was created in 1989. This group was composed mainly of manufacturers and users from Europe. In 1992, the first regional PROFIBUS organization was founded (PROFIBUS Schweiz in Switzerland). In the following years, additional Regional PROFIBUS & PROFINET Associations (RPAs) were added.

In 1995, all the RPAs joined together under the international umbrella association PROFIBUS & PROFINET International (PI). Today, PROFIBUS is represented by 25 RPAs around the world (including PNO) with over 1400 members, including most if not all major automation vendors and service suppliers, along with many end users.

Chapter 12

OpenSCADA

OpenSCADA is an open SCADA system constructed on principles of modularity, cross-platform and scalability.The OpenSCADA system is intended for: acquisition, archiving, visualization of the information, delivery of operating influences, and also for other related operations, which are characteristic for full-function SCADA systems.

Project targets

The basic purposes, which are pursued with the project, are:

- openness;
- reliability;
- flexibility;
- scalability;
- security;
- financial availability;
- providing the convenient control interface.

Scopes

The OpenSCADA system is intended for performance as SCADA systems of usual functionality, and for use in adjacent areas of information technologies.

The OpenSCADA system can be used:

- on industrial targets as full-function SCADA system;
- in built in systems, as the execution environment (including PLC);
- for construction of various models (technological, chemical, physical, electrical processes);

- on personal computers, servers and clusters for acquisition, processing, representation and archiving of the information about system and its environment.

Architecture

Heart of system is the modular kernel.

Depending on what modules are connected, the system can carry out both functions of various servers, and functions of clients of client–server architecture. Actually, the architecture of system allows to realize the distributed client-server systems of any complexity.

For achievement of high speed due to reduction of communications time, the architecture allows to unite functions of the distributed systems in one program.

Architecturally, the system OpenSCADA consists of following subsystems:

- The security subsystem. Contains lists of users and groups of users, provides check of the rights of access to system elements, etc.
- The modular DB subsystem. Provides access to databases. It includes the following modules:

Subversion

  -
    - DBF;
    - MySQL;
    - SQLite;
    - FireBird;
    - PostgreSQL.
- The modular transport subsystem. Provides the communications with an environment by means of various communication interfaces. It includes the following modules:
    - Sockets;
    - SSL;
    - Serial - Serial interface.
- The modular transport's protocol subsystem. It is closely connected with a subsystem of transports and provides support of various reports of an exchange with external systems. It includes the following modules:
    - HTTP - HTTP-realisation;
    - SelfSystem - Self system OpenSCADA protocol;
    - Modbus;
    - UserProtocol;
    - OPC_UA.
- The modular DAQ (data acquisition) subsystem. Provides data acquisition from external sources: controllers, sensors, etc. Except for it the subsystem can give

environment for a writing the data generators (models, regulators...). It includes the following modules:
    - BlockCalc - Block based calculator;
    - JavaLikeCalc - Java-like based calculator;
    - DiamondBoards - Diamond DA (data acquisition) boards;
    - System - System DA;
    - LogicLev - Logic level;
    - SNMP - SNMP client;
    - Modbus;
    - DCON - DCON client;
    - ICP_DAS - ICP DAS equipment;
    - DAQGate - Data sources gate;
    - SoundCard;
    - OPC_UA;
    - BFN - Support of the BFN modules for Viper CT/BAS and others from "Big Dutchman".
- The modular archive subsystem. Contains archives of two types: archives of messages and archives of values. An archiving way is defined by algorithm which is incorporated in the archiver's module. It includes the following modules:
    - FSArch - File system archiver;
    - DBArch - To DB archiver.
- The modular user interfaces subsystem. Contains functions of the user interfaces. It includes the following modules:
    - QTStarter - QT GUI starter;
    - QTCfg - System configurator (QT);
    - WebCfg - System configurator (Web);
    - WebCfgD - Dynamic WEB configurator;
    - VCAEngine - Visual control area engine;
    - Vision - Operation user interface (QT);
    - WebVision - Operation user interface (WEB);
    - WebUser - Web-interface from the user.
- The control modules subsystem. Provides the control over modules.
- The modular special subsystem. Contains functions not entered in other subsystems. It includes the following modules:
    - FLibComplex1 - Complex1 function's library;
    - FLibMath - Math functions' library;
    - FLibSYS - System API functions;
    - SystemTests - OpenSCADA system's tests;

Proceeding from a modules principle, the modular subsystems, which are specified above, can expand the functionality by connection of corresponding type of the modules.

The modular kernel of OpenSCADA system is designed in the form of static and shared libraries. It allows to build in functions of system existing programs, and also to create new programs on the basis of a modular kernel of OpenSCADA system.

However, the modular kernel is self-sufficient and can be used by means of the simple starting program.

Modules of OpenSCADA system are stored in dynamic libraries. Each dynamic library can contain set of modules of various type. Filling of dynamic libraries by modules is defined by functional connectivity of modules. Dynamic libraries suppose hot replacement that allows to make updating of modules during work. The method of storage of a code of modules in dynamic libraries is the core for OpenSCADA system as it is supported practically by all modern OS. It does not exclude an opportunity of development of other storage modules code methods.

Release history

Name/Version	Date
OpenSCADA 0.2.6	07/25/2004
OpenSCADA 0.3.0	11/07/2004
OpenSCADA 0.3.1	11/22/2004
OpenSCADA 0.4.0	09/07/2005
OpenSCADA 0.4.1	10/30/2005
OpenSCADA 0.5.0	07/27/2006
OpenSCADA 0.6.0	12/07/2007
OpenSCADA 0.6.1	03/23/2008
OpenSCADA 0.6.2	09/10/2008
OpenSCADA 0.6.3	01/12/2009
OpenSCADA 0.6.4	10/12/2009
OpenSCADA 0.6.4 (beta 2)	01/25/2010
OpenSCADA 0.6.4 (beta 3)	05/25/2010
OpenSCADA 0.7.0 - First production release	10/24/2010
OpenSCADA 0.7.0.1	12/19/2010
OpenSCADA 0.7.0.2	03/01/2011

Chapter 13

Stuxnet

Stuxnet is a Windows computer worm discovered in July 2010 that targets industrial software and equipment. While it is not the first time that hackers have targeted industrial systems, it is the first discovered malware that spies on and subverts industrial systems, and the first to include a programmable logic controller (PLC) rootkit.

The worm initially spreads indiscriminately, but includes a highly specialized malware payload that is designed to target only Siemens Supervisory Control And Data Acquisition (SCADA) systems that are configured to control and monitor specific industrial processes. Stuxnet infects PLCs by subverting the Step-7 software application that is used to reprogram these devices.

Different variants of Stuxnet targeted five Iranian organisations, with the probable target widely suspected to be uranium enrichment infrastructure in Iran; Symantec noted in August 2010 that 60% of the infected computers worldwide were in Iran. Siemens stated on November 29 that the worm has not caused any damage to its customers, but the Iran nuclear program, which uses embargoed Siemens equipment procured clandestinely, has been damaged by Stuxnet. Kaspersky Labs concluded that the sophisticated attack could only have been conducted "with nation-state support" and it has been speculated that Israel may have been involved.

History

Experts at first believed that Stuxnet started spreading around March or April 2010, but the first variant of the worm appeared in June 2009. A second, with substantial improvements, appeared in March 2010, apparently because its authors believed that Stuxnet was not spreading fast enough; a third, with minor improvements, appeared in April 2010. The worm contains a component with a build time stamp from 3 February 2010. In the United Kingdom on 25 November 2010, Sky News reported that it had received information from an anonymous source at an unidentified IT security

organization that Stuxnet, or a variation of the worm, had been traded on the black market.

The worm was first reported by the security company VirusBlokAda in mid-June 2010. Journalist Brian Krebs's 15 July 2010 blog posting was the first widely read report on the worm. Its name is derived from some keywords discovered in the software.

Affected countries

A study of the spread of Stuxnet by Symantec showed that the main affected countries in the early days of the infection were Iran, Indonesia and India:

Country	Infected computers
Iran	58.85%
Indonesia	18.22%
India	8.31%
Azerbaijan	2.57%
United States	1.56%
Pakistan	1.28%
Others	9.2%

Operation

Unlike most malware, Stuxnet does little harm to computers and networks that do not meet specific configuration requirements; "The attackers took great care to make sure that only their designated targets were hit...It was a marksman's job." While the worm is promiscuous, it makes itself inert if Siemens software is not found on infected computers, contains safeguards to prevent each infected computer from spreading the worm to more than three others, and to erase itself on 24 June 2012.

For its targets, Stuxnet contains, among other things, code for a man-in-the-middle attack that fakes industrial process control sensor signals so an infected system does not shut down due to abnormal behavior. Such complexity is very unusual for malware. The worm consists of a layered attack against three different systems:

1. The Windows operating system,
2. Siemens PCS 7, WinCC and STEP7 industrial software applications that run on Windows and
3. One or more Siemens S7 PLCs.

Windows infection

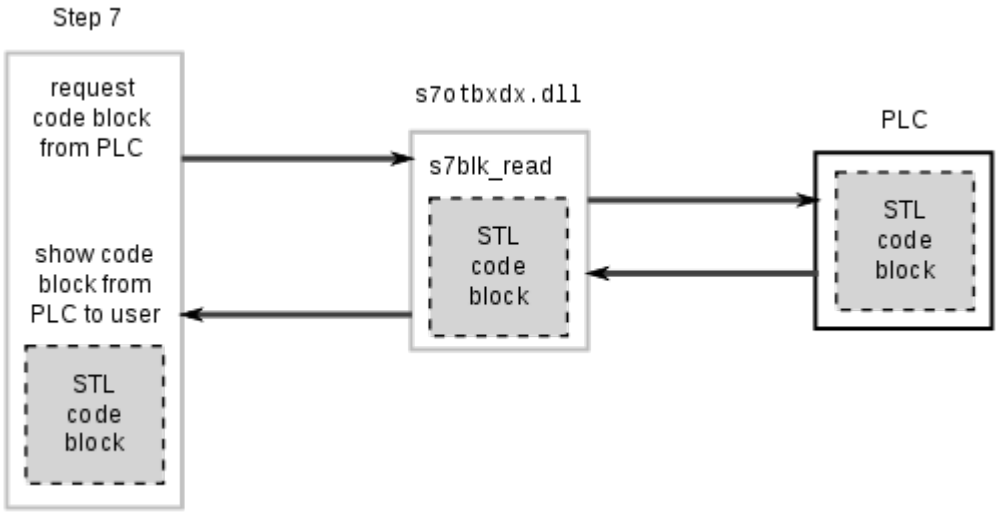
Stuxnet attacked Windows systems using an unprecedented four zero-day attacks (plus the CPLINK vulnerability and a vulnerability used by the Conficker worm) It is initially

spread using infected removable drives such as USB flash drives, and then uses other exploits and techniques such as peer-to-peer RPC to infect and update other computers inside private networks that are not directly connected to the Internet. The number of zero-day Windows exploits used is unusual, as zero-day Windows exploits are valued, and hackers do not normally waste the use of four different ones in the same worm. Stuxnet is unusually large at half a megabyte in size, and written in different programming languages (including C and C++) which is also irregular for malware. The Windows component of the malware is promiscuous in that it spreads relatively quickly and indiscriminately.

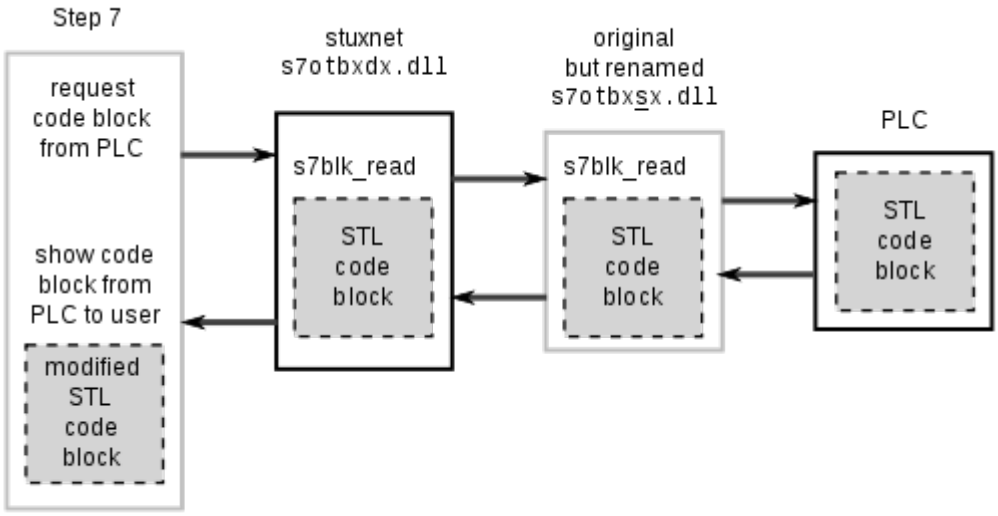
The malware has both user-mode and kernel-mode rootkit capability under Windows, and its device drivers have been digitally signed with the private keys of two certificates that were stolen from separate companies, JMicron and Realtek, that are both located at Hsinchu Science Park in Taiwan. The driver signing helped it install kernel-mode rootkit drivers successfully and therefore remain undetected for a relatively long period of time. Both compromised certificates have been revoked by VeriSign.

Two websites in Denmark and Malaysia were configured as command and control servers for the malware, allowing it to be updated, and for industrial espionage to be conducted by uploading information. Both of these websites have subsequently been taken down as part of a global effort to disable the malware.

Step 7 software infection



Overview of normal communications between Step 7 and a Siemens PLC

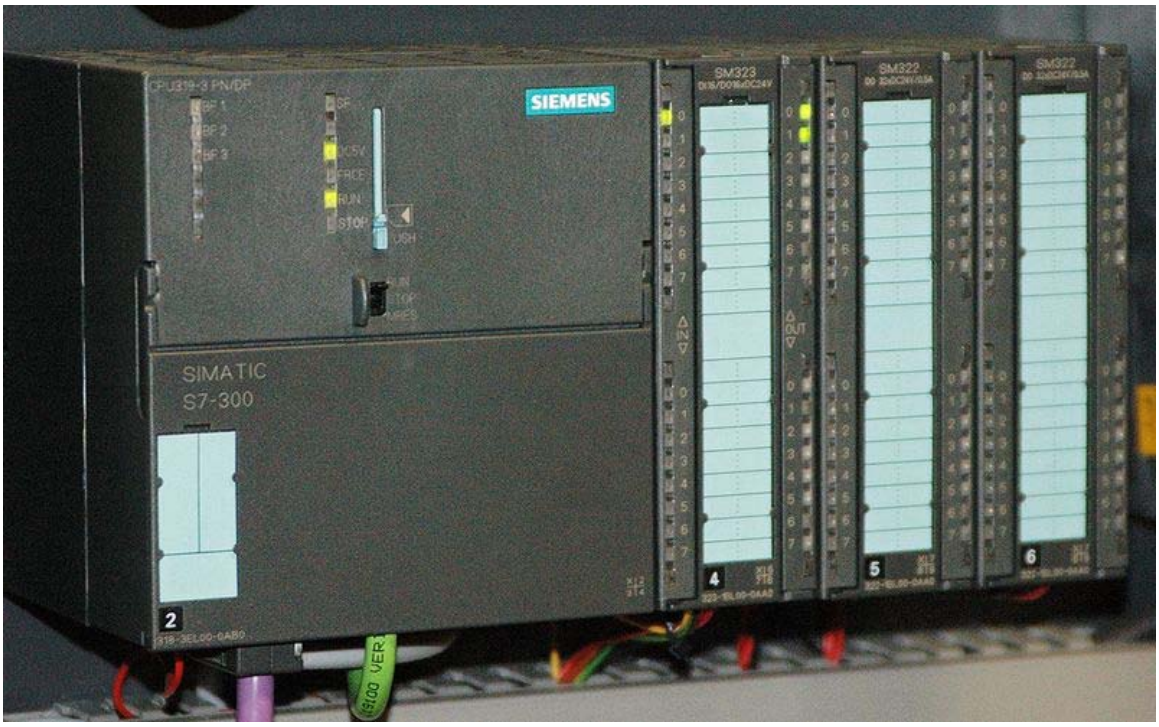


Overview of Stuxnet hijacking communication between Step 7 software and a Siemens PLC

According to German researcher Ralph Langner, once installed on a Windows system Stuxnet infects project files belonging to Siemens' WinCC/PCS 7 SCADA control software (Step 7), and subverts a key communication library of WinCC called `s7otbxdx.dll`. Doing so intercepts communications between the WinCC software running under Windows and the target Siemens PLC devices that the software is able to configure and program when the two are connected via a data cable. In this way, the malware is able to install itself on PLC devices unnoticed, and subsequently to mask its presence from WinCC if the control software attempts to read an infected block of memory from the PLC system.

The malware furthermore used a zero-day exploit in the WinCC/SCADA database software in the form of a hard-coded database password.

PLC infection



Siemens Simatic S7-300 PLC CPU with three I/O modules attached

The entirety of the Stuxnet code has not yet been understood, but its payload targets only those SCADA configurations that meet criteria that it is programmed to identify. Stuxnet requires specific slave variable-frequency drives (frequency converter drives) to be attached to the targeted Siemens S7-300 system and its associated modules. It only attacks those PLC systems with variable-frequency drives from two specific vendors: Vacon based in Finland and Fararo Paya based in Iran. Furthermore, it monitors the frequency of the attached motors, and only attacks systems that spin between 807 Hz and 1210 Hz. The industrial applications of motors with these parameters are diverse, and may include pumps or gas centrifuges.

Stuxnet installs malware into memory block DB890 of the PLC that monitors the Profibus messaging bus of the system. When certain criteria are met, it periodically modifies the frequency to 1410 Hz and then to 2 Hz and then to 1064 Hz, and thus affects the operation of the connected motors by changing their rotational speed. It also installs a rootkit—the first such documented case on this platform—that hides the malware on the system and masks the changes in rotational speed from monitoring systems.

Removal

Siemens has released a detection and removal tool for Stuxnet. Siemens recommends contacting customer support if an infection is detected and advises installing Microsoft

patches for security vulnerabilities and prohibiting the use of third-party USB flash drives. Siemens also advises immediately upgrading password access codes.

The worm's ability to reprogram external PLCs may complicate the removal procedure. Symantec's Liam O'Murchu warns that fixing Windows systems may not completely solve the infection; a thorough audit of PLCs may be necessary. Despite speculation that incorrect removal of the worm could cause damage, Siemens reports that in the first four months since discovery, the malware was successfully removed from the systems of twenty-two customers without any adverse impact.

Control system security

Prevention of control system security incidents, such as from viral infections like Stuxnet, is a topic that is being addressed in both the public and the private sector.

The U.S. Department of Homeland Security National Cyber Security Division operates the Control System Security Program (CSSP). The program operates a specialized Computer Emergency Response Team (ICS-CERT), conducts a biannual conference (ICSJWG), provides training, publishes recommended practices, and provides a self-assessment tool. As part of a Department of Homeland Security plan to improve American computer security, in 2008 it and the Idaho National Laboratory (INL) worked with Siemens to identify security holes in the company's widely used Process Control System 7 (PCS 7) and its software Step 7. In July 2008 INL and Siemens publicly announced flaws in the control system at a Chicago conference; Stuxnet exploited these holes in 2009.

Several industry organizations and professional societies have published standards and best practice guidelines providing direction and guidance for control system end-users on how to establish a Control System Security management program. The basic premise that all of these documents share is that prevention requires a multi-layered approach, often referred to as "defense-in-depth". The layers include policies & procedures, awareness & training, network segmentation, access control measures, physical security measures, system hardening, e.g., patch management, and system monitoring, anti-virus and IPS. The standards and best practices also all recommend starting with a risk analysis and a control system security assessment. The purpose is to assess the current level of risk and the size of the gap between that risk and what is tolerable. The other purpose of an assessment is to identify the vulnerabilities and develop a prioritized program to eliminate or minimize them.

In response to these concerns, cyber security standards and certifications programs such as ISA 99 and SASecure have been developed to evaluate and certify the security of industrial automation products.

Automation, SCADA and control system developers often use off-the-shelf equipment, software and protocols, integrating and configuring these in different ways for a variety of applications. This "common" approach can make it easier for malware to bring down

some vulnerable systems. However, proprietary automation, SCADA and control system developers are able to provide a completely bespoke solution, using new protocols and hardware/software/firmware solutions yet unknown to developers of malware.

Speculations about the target and origin

Experts believe that Stuxnet required the largest and costliest development effort in malware history. Its many capabilities would have required a team of people to program, in-depth knowledge of industrial processes, and an interest in attacking industrial infrastructure. Eric Byres, who has years of experience maintaining and troubleshooting Siemens systems, told *Wired* that writing the code would have taken many man-months, if not years. Symantec estimates that the group developing Stuxnet would have consisted of anywhere from five to thirty people, and would have taken six months to prepare. *The Guardian*, the BBC and *The New York Times* all reported that experts studying Stuxnet considered that the complexity of the code indicates that only a nation state would have the capabilities to produce it. The self-destruct and other safeguards within the code imply that a Western government was responsible, with lawyers evaluating the worm's ramifications. Bruce Schneier condemned the news coverage of Stuxnet as hype, however, stating that it is almost entirely based on speculation.

Iran as target

Ralph Langner, the researcher who identified that Stuxnet infected PLCs, first speculated publicly in September 2010 that the malware was of Israeli origin, and that it targeted Iranian nuclear facilities. Kevin Hogan, Senior Director of Security Response at Symantec, reported that the majority of infected systems were in Iran (about 60%), which has led to speculation that it may have been deliberately targeting "high-value infrastructure" in Iran including either the Bushehr Nuclear Power Plant or the Natanz nuclear facility. Langner called the malware "a one-shot weapon" and said that the intended target was probably hit, although he admitted this was speculation. Another German researcher, Frank Rieger, was the first to speculate that Natanz was the target.

Natanz nuclear facilities

Iranian President Mahmoud Ahmadinejad looking at gas centrifuge cascades at the Natanz nuclear facility in Iran, April 2008

According to the Israeli newspaper Haaretz, experts on Iran and computer security specialists are increasingly convinced that Stuxnet was meant "to sabotage the uranium enrichment facility at Natanz – where the centrifuge operational capacity has dropped over the past year by 30 percent." On 23 November 2010 it was announced that uranium enrichment at Natanz had ceased several times because of a series of major technical problems. A "serious nuclear accident" occurred at the site in the first half of 2009, which is speculated to have forced the head of Iran's Atomic Energy Organization Gholam Reza Aghazadeh to resign. Statistics published by the Federation of American Scientists (FAS) show that the number of enriched centrifuges operational in Iran mysteriously declined from about 4,700 to about 3,900 beginning around the time the nuclear incident WikiLeaks mentioned would have occurred. The Institute for Science and International Security (ISIS) suggests in a report published in December 2010 that Stuxnet is "a reasonable explanation for the apparent damage" at Natanz and may have destroyed up to 1000 centrifuges (10 percent) sometime between November 2009 and late January 2010. The authors conclude:

"The attacks seem designed to force a change in the centrifuge's rotor speed, first raising the speed and then lowering it, likely with the intention of inducing excessive vibrations or distortions that would destroy the centrifuge. If its goal was to quickly destroy all the centrifuges in the FEP, Stuxnet failed. But if the goal was to destroy a more limited number of centrifuges and set back Iran's progress in operating the FEP, while making detection difficult, it may have succeeded, at least temporarily."

The ISIS report further notes that Iranian authorities have attempted to conceal the breakdown by installing new centrifuges on a large scale.

According to the *Washington Post*, International Atomic Energy Agency (IAEA) cameras installed in the Natanz facility recorded the sudden dismantling and removal of approximately 900-1000 centrifuges during the time the Stuxnet worm was reportedly active at the plant. Iranian technicians, however, were able to quickly replace the centrifuges and the report concluded that uranium enrichment was likely only briefly disrupted.

An analysis by FAS experts demonstrate that Iran's enrichment capacity grew during 2010. The study indicates that Iran's centrifuges appear to be performing 60 percent better than in the previous year, which would significantly reduce Tehran's time to produce bomb-grade uranium. The new FAS report was reviewed by an official with the IAEA who affirmed the study.

Iranian reaction

The Associated Press reported that the semi-official Iranian Students News Agency released a statement on 24 September 2010 stating that experts from the Atomic Energy Organization of Iran met in the previous week to discuss how Stuxnet could be removed from their systems. According to analysts, such as David Albright, Western intelligence agencies have been attempting to sabotage the Iranian nuclear program for some time.

The head of the Bushehr Nuclear Power Plant told Reuters that only the personal computers of staff at the plant had been infected by Stuxnet and the state-run newspaper *Iran Daily* quoted Reza Taghipour, Iran's telecommunications minister, as saying that it had not caused "serious damage to government systems". The Director of Information Technology Council at the Iranian Ministry of Industries and Mines, Mahmud Liaii, has said that: "An electronic war has been launched against Iran... This computer worm is designed to transfer data about production lines from our industrial plants to locations outside Iran."

In response to the infection, Iran has assembled a team to combat it. With more than 30,000 IP addresses affected in Iran, an official has said that the infection is fast spreading in Iran and the problem has been compounded by the ability of Stuxnet to mutate. Iran has set up its own systems to clean up infections and has advised against using the Siemens SCADA antivirus since it is suspected that the antivirus is actually embedded with codes which update Stuxnet instead of eradicating it.

According to Hamid Alipour, deputy head of Iran's government Information Technology Company, "The attack is still ongoing and new versions of this virus are spreading." He reports that his company had begun the cleanup process at Iran's "sensitive centres and organizations." "We had anticipated that we could root out the virus within one to two months, but the virus is not stable, and since we started the cleanup process three new versions of it have been spreading," he told the Islamic Republic News Agency on 27 September 2010.

On 29 November 2010, Iranian president Mahmoud Ahmadinejad stated for the first time that a computer virus had caused problems with the controller handling the centrifuges at its Natanz facilities. According to Reuters he told reporters at a news conference in Tehran, "They succeeded in creating problems for a limited number of our centrifuges with the software they had installed in electronic parts."

On the same day two Iranian nuclear scientists were targeted in separate, but nearly simultaneous car bomb attacks near Shahid Beheshti University in Tehran. Majid Shahriari, head of the Iranian nuclear program was killed. Fereydoon Abbasi, a high-ranking official at the Ministry of Defense was seriously wounded. *Wired* speculated that the assassinations could indicate that whoever was behind Stuxnet felt that it was not sufficient to stop the nuclear program. In January 2010, another Iranian nuclear scientist, a physics professor at Tehran University, had been killed in a similar bomb explosion.

Given the growth in Iranian enrichment capability in 2010, the country may have intentionally put out disinformation to cause Stuxnet's creators to believe that the worm was more successful in disabling the Iranian nuclear program than it actually was.

Possible origin

Both Israel and the United States or other Western nations, working separately or together, have been named as possible creators of Stuxnet.

Israel

Israel, perhaps through Unit 8200, has been speculated to be the country behind Stuxnet in many media reports and by experts such as Richard Falkenrath, former Senior Director for Policy and Plans within the U.S. Office of Homeland Security. Yossi Melman, who covers intelligence for the Israeli daily newspaper *Haaretz* and is writing a book about Israeli intelligence, also suspected that Israel was involved, noting that Meir Dagan, the former (2011) head of the national intelligence agency Mossad, had his term extended in 2009 because he was said to be involved in important projects. Additionally, Israel now expects that Iran will have a nuclear weapon in 2014 or 2015—at least three years later than earlier estimates—without the need for an Israeli military attack on Iranian nuclear facilities; "They seem to know something, that they have more time than originally thought", he added. Israel has not publicly commented on the Stuxnet attack but confirmed that cyberwarfare is now among the pillars of its defense doctrine, with a military intelligence unit set up to pursue both defensive and offensive options. When questioned whether Israel was behind the virus in the fall of 2010, some Israeli officials broke into "wide smiles", fueling speculation that the government of Israel was involved with its genesis. American presidential advisor Gary Samore also smiled when Stuxnet was mentioned, although American officials have indicated that the virus originated abroad. According to *The Telegraph*, Israeli newspaper *Haaretz* reported that a video celebrating operational successes of Gabi Ashkenazi, retiring IDF Chief of Staff, was shown at his retirement party and included references to Stuxnet, thus strengthening claims that Israel's security forces were responsible.

In 2009, a year before Stuxnet was discovered, Scott Borg of the United States Cyber-Consequences Unit (US-CCU) suggested that Israel might prefer to mount a cyber-attack rather than a military strike on Iran's nuclear facilities. According to Borg this kind of attack could involve disrupting sensitive equipment such as centrifuges using malware introduced via infected memory sticks: "Since the autumn of 2002, I have regularly predicted that this sort of cyber-attack tool would eventually be developed ... Israel certainly has the ability to create Stuxnet and there is little downside to such an attack, because it would be virtually impossible to prove who did it. So a tool like Stuxnet is Israel's obvious weapon of choice." Iran uses P-1 centrifuges at Natanz, the design for which A. Q. Khan stole in 1976 and took to Pakistan. His black market nuclear-proliferation network sold P-1s to, among other customers, Iran. Experts believe that Israel also somehow acquired P-1s and tested Stuxnet on the centrifuges, installed at the Dimona facility that is part of its own nuclear program. The equipment may be from the United States, which received P-1s from Libya's former nuclear program.

Some have also referred to several clues in the code such as a concealed reference to the word "MYRTUS", believed to refer to the Myrtle tree, or Hadassah in Hebrew. Hadassah was the birth name of the former Jewish queen of Persia, Queen Esther. However, it may be that the "MYRTUS" reference is simply a misinterpreted reference to SCADA components known as *RTUs* (Remote Terminal Units) and that this reference is actually "My RTUs"–a management feature of SCADA. Also, the number 19790509 appears once in the code and might refer to the date "1979 May 09", the day Habib Elghanian, a

Persian Jew, was executed in Tehran. Another date that appears in the code is "24 September 2007", the day that Iran's president Mahmoud Ahmadinejad spoke at Columbia University and made comments questioning the validity of the Holocaust. Such data is not conclusive, since, as written by Symantec, "Attackers would have the natural desire to implicate another party" with a false flag.

United States

There has also been speculation on the involvement of the United States, with one report stating that "there is vanishingly little doubt that [it] played a role in creating the worm." It has been reported that the United States, under one of its most secret programs, initiated by the Bush administration and accelerated by the Obama administration, has sought to destroy Iran's nuclear program by novel methods such as undermining Iranian computer systems. A diplomatic cable obtained by WikiLeaks showed how the United States was advised to target Iran's nuclear capabilities through 'covert sabotage'. A Wired article claimed that Stuxnet "is believed to have been created by the United States". The CIA may have caused a large Siberian pipeline explosion in 1982 by sabotaging SCADA software the Soviets stole.

Joint effort and other nations

According to *Vanity Fair*, Rieger stated that three European countries' intelligence agencies agreed that Stuxnet was a joint United States-Israel effort. The code for the Windows injector and the PLC payload differ in style, likely implying the participation of two nations. Other experts believe that a US-Israel cooperation is unlikely because "the level of trust between the two countries' intelligence and military establishments is not high." Jordan and France are other possibilities, and Siemens may have also participated. Langner also speculated that the infection may have spread from USB drives belonging to Russian contractors. On 15 July 2010, the day the worm's existence became widely known, a distributed denial-of-service attack—almost certainly from Russia and likely related to Stuxnet—made on the servers for two leading mailing lists on industrial-systems security disabled one of the lists.