Automation and Robotics in Construction X
G.H. Watson, R.L. Tucker and J.K. Walters (Editors)

GENERIC CRANE AND HOIST AUTOMATION WITH THE APPLICATION OF ADVANCED CONTROL ARCHITECTURES

N.A. Armstrong, P.R. Moore and R.H. Weston
Modular Systems Research Group
Department of Manufacturing Engineering
Loughborough University of Technology
Loughborough, Leicestershire
LE11 3TU, United Kingdom

Synopsis

Crane and hoist systems can potentially benefit from contemporary advances made in electronics based enabling technologies and control systems engineering. This paper describes the design and implementation of a modular distributed control system for crane and hoist automation in manufacturing and construction. The distributed control platform has been realised using the *FIP-Fieldbus* and the open machine control integration platform *UMC (Universal Machine Control)*. The technology is evaluated on a gantry crane 'demonstrator' which embodies control features such as; anti-swing, condition monitoring, tele-operation, automatic load coupling/decoupling, and automatic cycling. The project has successfully proven the applicability of modular distributed control systems for industrial automation and has substantiated the benefits of designing, implementing and maintaining control systems using an open machine control platform.

1.0 Introduction

A three year research programme entitled *MACHINE (Monitoring And Control of Hoists in INdustrial Environments)* started in September 1990 with the objective of applying innovative and cost effective improvements in the control, monitoring and mechanical design aspects of hoist and crane technologies. The desired outcome being semi-autonomous mechanical handling systems for diverse industrial sectors which demonstrate enhanced functional performance and improved efficiency and working conditions of operators. This collaborative research venture between six European partners was sponsored under the BRITE-EURAM initiative to a value of some 2 million ECU [1,2].

Principal aims of the research programme were to produce representative mathematical models of typical crane and hoist dynamics for subsequent use in designing improved crane control schemes; advance the performance of crane drive systems and hoist mechanisms; to establish a method of automatic or semi-automatic coupling of crane end-effectors to specified loads, and to demonstrate a reliability of coupling of up to 98%; to demonstrate guaranteed vertical lifting so as to eliminate swing at pick up by at least 90%, and to reduce the over swing of the load during traversing motions; develop a prototype performance record and condition monitoring system for cranes - useful for the production and maintenance planning; demonstrate operation with reduced personnel; provide real-time position data for operators, and demonstrate the efficient integration of the various sub-systems.

A control architecture was required which would integrate the control of crane axis drives and input/output hardware and support the collection of crane monitoring and collision avoidance data, provided a platform for swing control regimes, operator interface functions and partial

autonomous operation. The control architecture needed to be sufficiently flexibility to enable such features to be developed independently and added, modified or removed; and allow the addition of, as yet unspecified, features which may be required in the future. For these reasons a distributed implementation of a control system based on the Universal Machine Control (UMC) architecture was chosen [3,4].

2.0 A Reference Architecture for Machine Control

The specification and design of Reference Architectures for machine and process control has been the subject of the ongoing research of the Modular Systems Group at Loughborough University of Technology. This research has produced the Universal Machine Control (UMC) system, which is an *open systems* methodology and software suite from which custom control systems can be designed and built from largely standard software modules; implementation-specific software is used to customise the application, but this is a significantly reduced amount compared with a specifically written software solution [5,6]. The current UMC run-time system is implemented on the Motorola 68XXX family of processors using the Microware OS9 real-time operating system. Run-time UMC machines typically use networked, diskless, rack based, single board computers with VME and/or G64 parallel backplanes. A machine's control system based on the UMC Reference Architecture (see Figure 1.) can be built quickly, simply and cost effectively from a range of re-usable software modules. Such machines can be simple or complex determined specifically by the application. The complexity of the control system reflects this and can be easily modified to accommodate updated hardware/software or for the system to be reconfigured or expanded to provide additional functionality as and when required.

Configuration tools allow the machine's structure to be defined in terms of the device hardware/software used and the user specific software which customises the control system for a particular application. Machine configuration data can also be used by associated modelling and simulation tools for offline development and testing of application code; these are useful for evaluating and emulating machine performance before it is used for a 'real-world' control system. An interface to higher level factory automation systems is also available through the CIM-BIOSYS *open systems* integration platform [7].

Run-time data associated with the machine and its hardware is stored in global data-areas which permits excellent data visibility for machine monitoring, diagnostics and error recovery strategies. Software modules (called handlers) provide a virtual interface through which the control system can access a range of devices. As a result of this uniformity, devices from different vendors can be seamlessly integrated in a single solution. The user's application software calls a set of 'standard' UMC functions which provide the interface, via handlers, to the machine hardware; as such the author of application software is freed from writing this often complex and hardware specific code and can concentrate on the task's primary purpose, namely, the user requirements. UMC architecture is hierarchically structured thus:

2.1 Machine Level

The machine level defines the overall format and structure of the control system. Utilities are provided for creating and modifying machine data (edit files which describe the machine in terms of number of axes, ports, tasks, events, device specific data etc) and location files. At run time a machine configuration utility creates the actual machine from these data files. Utilities exist for monitoring and dynamically modifying the functioning of a UMC machine during execution.

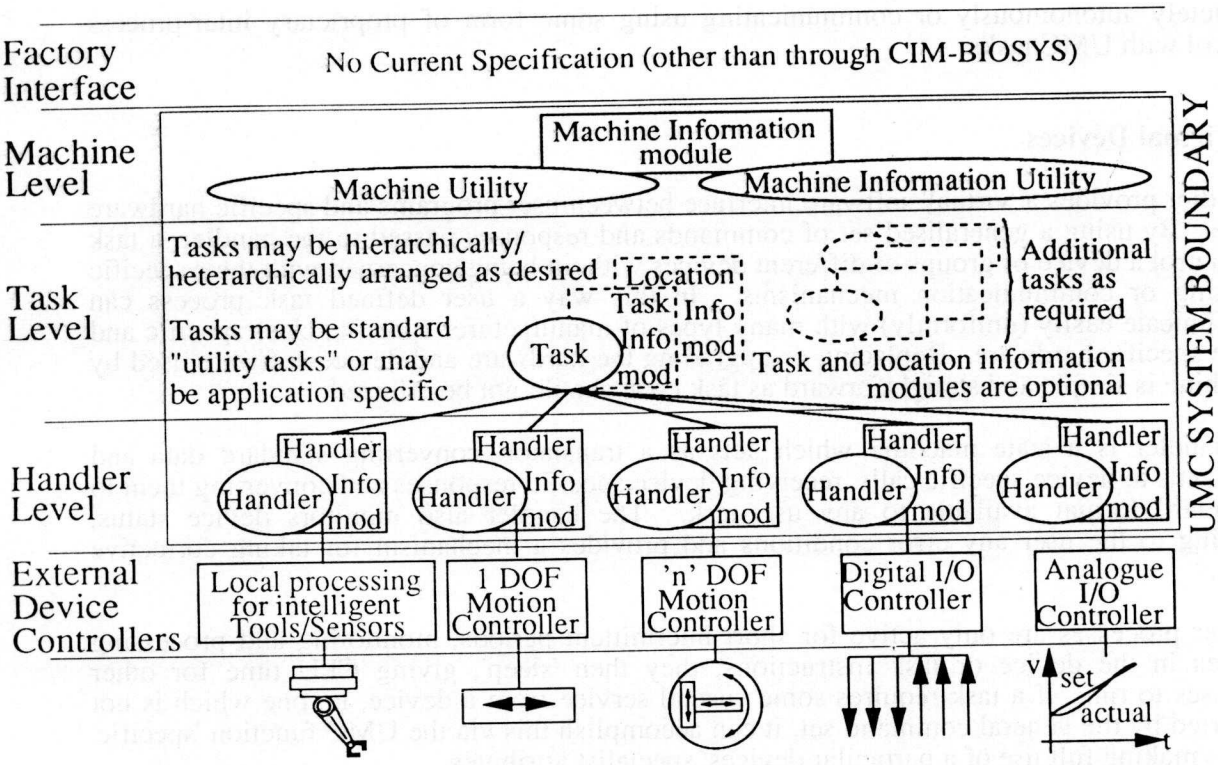

Figure 1. Universal Machine Control Hierarchy

2.2 User Tasks

Application specific programs (referred to as user tasks) are currently written using the C language. A library of 'standard' functions provides all the necessary links to the UMC control system. Users are then free to write whatever code they need, specific to their requirements, with access to all the UMC machine hardware and data that the task may require. Library functions include calls for reading input and writing output channels, single and multi-axis constant velocity and trapezoidal velocity moves, mapping and profiling of multiple axes, velocity control, teaching positions and moving to predefined locations. Library functions are also available for reading and setting parameters, general error recovery and status reporting, inter-task synchronisation and communication [8].

A machine will typically be made up of groups of devices, ie, the associated axes constituting a robot arm. Axis positions can be defined off-line using a dedicated editor or on-line using a specially written task. These locations are stored, at run-time, in a data module and can be saved as edit files (on disk) for re-use each time the machine is run. A user task might specifically control one group of axes with different tasks supervising other groups or performing other duties of the machine. A natural decomposition of the application is easily supported. Occasions may arise where a device is associated with two or more groups or user tasks, ie, an axis passing parts between work stations, in which case both tasks can share access of a single device. User processes may monitor the status of a handler at any time, without being the controlling task, by reading data from the device's global data module.

The UMC architecture imposes no constraints on the content or structure of the application code, linking to the library is necessary only if access is required to UMC functions. Proprietary (non-UMC) software modules will run alongside UMC processes, functioning

completely autonomously or communicating using some form of proprietary inter-process protocol with UMC tasks.

2.3 Virtual Devices

A handler provides a 'virtual' software interface between user programs and specific hardware devices. By using a generalised set of commands and responses passed to the handler, a task can control a device or groups of different devices without being concerned with their specific operating or communication mechanisms. In this way a user defined task process can communicate easily (uniformly) with many types of manufacturer specific, class specific and device specific hardware. Replacing or upgrading the hardware and device-software used by a machine is simple and straightforward as task code should not be effected.

The handler is a state machine which acts as a translator, converting standard data and commands to device specific calls, receiving device specific responses and converting them to a standard format available to any user-task. The handler also monitors device status, reporting to the user any error conditions and provides a mechanism for taking corrective action.

Handler processes are only active for short intermittent periods, monitoring and processing changes in the device or task instructions; they then 'sleep', giving CPU time for other processes to run. If a task requires some special service from a device, ie, one which is not supported by the general command set, it can accomplish this via the UMC function 'specific' call, so making full use of a particular devices' specialist attributes.

3.0 Industrial Distributed Control Networks

Around 1984, standards bodies and industrial collaborative groups, in America and Europe began work on the creation of open standard digital communications networks for low level industrial devices (Fieldbus). This resulted in three distinct variations: the Instrument Society of America (ISA) Fieldbus, FIP from France and Profibus from Germany. The International Electrotechnical Commission (IEC) decided that the ISA fieldbus would form the basis for the international standard, although elements from FIP, Profibus and other factions will also have been included in the final standard.

In an attempt to hasten the realisation of an open standard, groups such as the International Fieldbus Consortium (IFC) and Inter-operable Systems Project (ISP) have formed to provide early 'Fieldbus' solutions, however, it is unlikely that devices with IFC or ISP fieldbus will be available before 1994. Meanwhile, the FIP and Profibus groups have continued to form national standards, vendor and user bases; and now see a broad, and increasing, use of their respective 'fieldbuses'. Network systems from different industrial sectors have also been proposed as suitable for the requirements of process, manufacturing and construction industry automation, Eg. MIL-STD-1553B a military standard and ARINC 629 an aviation standard; SERCOS from the machine-tool industry and CANBUS from the automotive industry. Proprietary networks from major PLC, motor-drive and instrumentation manufacturers and specialist networking companies have also become prominent, including Intel's Bitbus, Rosemount's Hart, Echelon Lonworks, and Pheonix contact's Interbus-S. These lists are by no means comprehensive, but serve to indicate the interest shown and potential market for such systems.

From the criteria for a simple digital replacement of an analogue standard, the draft Fieldbus standard now contains many (sometimes disparate) options in an attempt to provide a solution satisfactory for use in many diverse industrial application domains. It is hardly surprising

therefore, that there would be a certain amount of procrastination and, at times, acrimony between the major players and that the formation of the IEC standard has been overdue. It is expected that the Physical-layer standard will be finalised very soon, and that the Datalink-layer is expected to become a draft standard by the middle of the year. There seems to be little progress towards an Application-layer specification which is acceptable to all the members of the associated working group [9]. At various times a (real-time) subset of Manufacturing Automation Protocol (MAP)'s Manufacturing Message Specification (MMS) services [10], global-object services similar to those offered by FIP and an application layer based on Profibus services have been suggested.

Potentially of greatest interest to a user of fieldbus is the 'User-layer'; a proposed additional tier in the hierarchy, although not part of the original ISO model [11]. The User-layer essentially 'hides' all networking functionality from the user by providing a set of virtual-device services which appear to directly interface the user's software with the remote device. For example, virtual-device function blocks would exist for instigating one of a standard set of PID loops on a remote device, what the fieldbus actually does to achieved this is of no concern to the user, other than wether is has been successful or not. As with the other layers, various classes of conformance, from completely defined (genuinely Open) to manufacturer/device specific have had to be included. NB: the UMC user-task library is analogous to the proposed IEC User-layer in providing a set of functions by which a user can control a specific device in a virtual manner.

3.1 Selection of A Distributed Control Network

An international standard fieldbus would have been desirable for this project, but would not have been available within it's life span. However, a system which offers features close to those likely in the IEC standard is appealing as it would then offer an easy migration path for future developments or applications arising from the project.

A 'distributed real-time database' was recognised to suit the global data-sharing aspect of a distributed version of the UMC architecture. The only *Open System* currently offering guaranteed real-time distributed-object services, for which multi-vendor hardware and software is available is the FIP fieldbus contender [12]. The implementation of FIP used also offers an acyclic messaging (and file transfer) facility; multiple physical media, speed and intrinsic safety options; (potentially) some MMS like services; ready availability of hardware from various sources; and an Open (French) national standard.

4.0 Distributed Control Implementation of UMC

The approach adopted for a modular distributed control system employs many of the concepts and attributes of the 'centralised' form of UMC. The users application code is defined within task programs which access the system via a library of standard functions. To convert a task written for a centralised UMC machine to one suitable for a distributed application should require little more than relinking it with the appropriate library. Task and Handler processes communicate in a consistent manner, whether they are situated on the same or different processors. Machine and device global data is accessible to any process that requires it, via the 'distributed data-base' maintained by the network. Distributed processes use a software interrupt mechanism for processing network service requests, this gives a fast response to aperiodic requests and allows the normal program-flow of tasks and handlers to be maintained.

Where centralised UMC uses a system of OS9 events, signals and shared data areas, the distributed form has network functions which globally mimic or locally reproduce these

interprocess communication mechanisms. Attempts have been made to optimise the performance of distributed-system calls in-so-much as frequently used data is stored locally rather than repetitively requesting it from the network. However, some of the 'internal' functionality is not directly transferable between the two approaches, Eg. storing device data module addresses (pointers) in the machine data module of a centralised system is simple and efficient but not applicable in a distributed environment; in this instance the data module addresses will be different at each node and are therefore stored locally by each process.

The distributed implementation of UMC uses both the FIP synchronous object-variable 'data-base' and asynchronous message transmission services. Fip variables are used for maintaining a global copy of the real-time machine and device data. Messaging is used for the remote setting of OS9 events and signals and consumer to producer variable update requests. A combination of FIP variables and messaging is used for a global event mechanism.

Special distributed control utilities have been written to support local configuration of the data-modules (which define the access of networked global data), loading of task location modules, creation of application tasks, handler and network processes, and monitoring and resetting of global event and data-values.

The FIP implementation of distributed Universal Machine Control was first publicly demonstrated on a five axis modular robotic machine at a United Kingdom Department of Trade and Industry 'Fieldbus Open Day' at Loughborough in November 1992.

5.0 Implementation of the Demonstrator Crane Control System

5.1 Hardware Architecture

The demonstrator uses a suitably designed overhead-travelling (gantry) crane. The control-hardware architecture uses three FIP network nodes linked by a shielded twisted pair cable operating at a raw data-rate of 1Mbit/s. One node is mounted on the trolley (crab), this controls two variable-speed vector drives (for the cross-travel and hoist motors) via closed-loop motion controllers, plus an I/O module which supports digital and analogue channels. One node is mounted on the bridge, this controls the long-travel via a motion controller and vector drives and an associated I/O module; it also provides a 'gateway' between the FIP and proprietary-radio networks. The radio link is used to connect a data-logging node (for condition-monitoring), tele-operator and a collision avoidance system. The third node is mounted on the crane grab and provides the various I/O functions. Figure 2 depicts the control system hardware architecture.

5.2 Software Architecture

UMC is used to provide a control architecture for the fundamental crane control system and set of basic networking functions for communication between this and the proprietary, ie, non-UMC applications - data-collection, tele-operator and grab-control software (Figure 3). A UMC single-axis handler is used to control each drive, along with a UMC binary-analogue handler for each block of I/O.

The application code is split into three tasks. The first task (task1) provides the interface between the control system 'core' and the peripheral systems. It passes crane position data to the radio nodes and provides the man-machine-interface - importing commands via keyboard and tele-operator; these are processed and executed directly for grab I/O commands or exported as gross crane demand-position data / move-commands to the other tasks. These tasks use the gross commands and data to drive the crane hardware via UMC function calls to

the handlers. The second task (task2) uses an anti-swing control algorithm to process the 'move' commands, and drives the handler/hardware in a low-level-UMC velocity control mode. The third task (task3) processes the same commands using high-level-UMC, multi-axis trapezoidal move functions. By switching between task2 and task3, the same gross command can be executed with or without the anti-swing control algorithm.

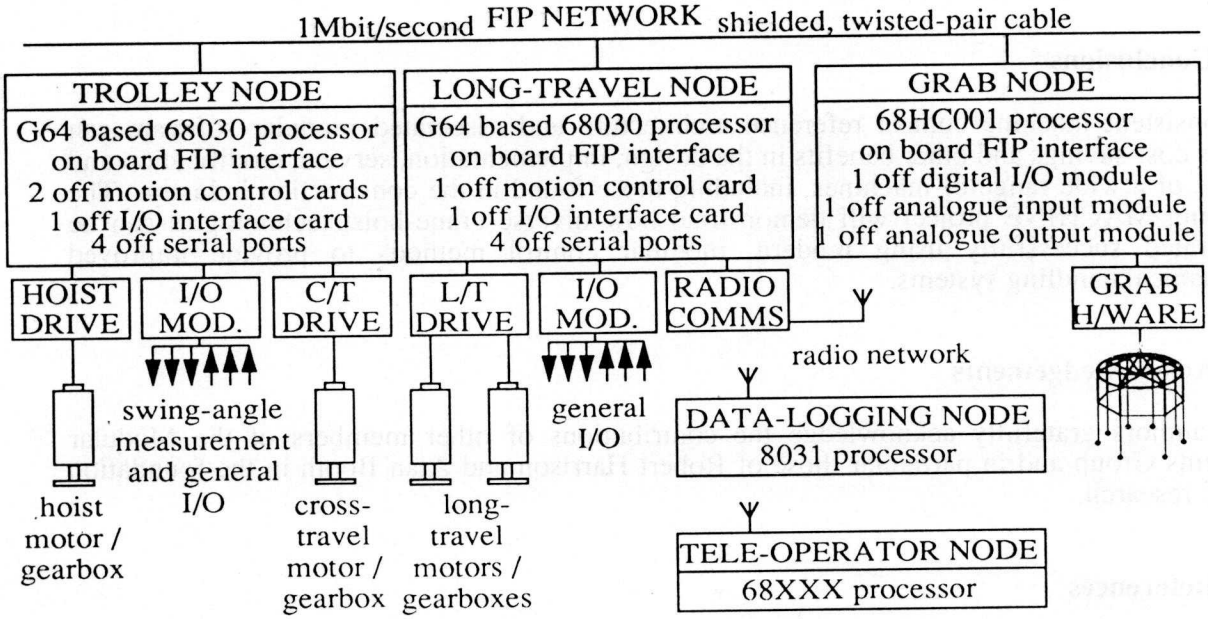

Figure 2. Demonstrator Hardware Architecture

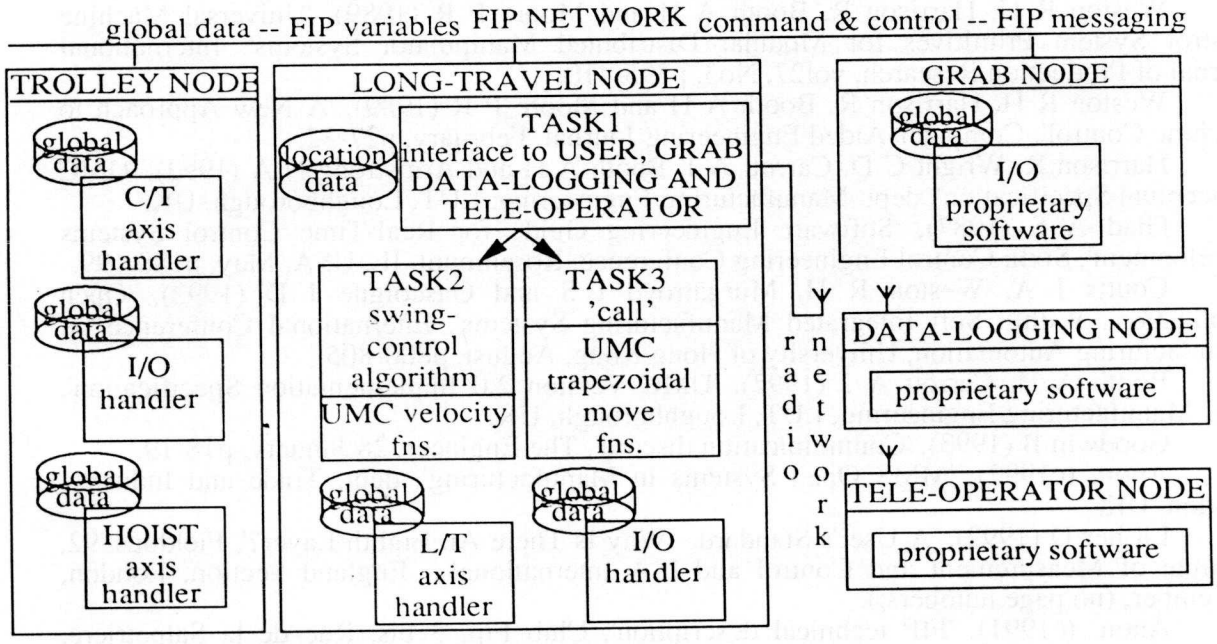

Figure 3. Demonstrator Software Architecture

By splitting the control requirements into discrete elements, each part can be written, debugged and tested in isolation. Furthermore, piecemeal changes can be made without forcing a major re-write of the software. ie, changing the drive hardware might require a different handler but the rest of the software would remain unchanged. Likewise, changing the anti-swing control algorithm would require modification of task2, but nothing else. This approach also applies to the integration of proprietary systems which can be installed, modified or removed in a similar fashion.

6.0 Conclusions

A consistent machine control reference architecture and associated modular software can offers cost-savings and other benefits in the design, implementation, service, modification and re-use of a wide range of machines, including those found in the construction industry. The ongoing MACHINE project will demonstrate how diverse crane-hoist technologies can be integrated successfully using modern, modular control methods to provide improved mechanical handling systems.

7.0 Acknowledgements

The authors gratefully acknowledge the contributions of other members of the Modular Systems Group and in particular those of Robert Harrison and Alan Booth in the foundation UMC research.

8.0 References

1 Anon. (1991), 'MACHINE: Monitoring and Control of Hoist Technology in Industrial Environments', BRITE EURAM programme synopses of current projects 1990-91 second edition, CEC. p238.

2 Dodman K (1991), 'Crane Automation', Cranes Today, March, p41-44.

3 Weston R H, Harrison R, Booth A H and Moore P R (1989), 'Universal Machine Control System Primitives for Modular Distributed Manipulator Systems'. International Journal of Production Research, vol27, No3, p395-410.

4 Weston R H, Harrison R, Booth A H and Moore P R (1989), 'A New Approach to Machine Control'. Computer-Aided Engineering Journal, February, p27-32.

5 Harrison R, Wright C D, Carrott A J, Booth A H and Armstrong N A (1992), 'UMC Conceptual Specification', dept. Manufacturing Engineering, LUT, Loughborough, UK.

6. Glad A S (1987), 'Software Engineering Guide for Real-Time Control Systems Development', Sixth Control Engineering Conference, Rosemount, IL. USA, May, p188-199.

7 Coutts I A, Weston R H, Murgatroyd I S and Gascoigne J D (1992), 'Open Applications Within Soft Integrated Manufacturing Systems'. International Conference on Manufacturing Automation, University of Hong Kong, August, p800-805.

8 Booth A H, Carrott A J (1992), 'UMC Version 2.0 Implementation Specification', dept. Manufacturing Engineering, LUT, Loughborough, UK.

9 Goodwin B (1993), 'Communication discord', The Engineer, 28 January, p18-19.

10 Anon. (c1992), 'MMS Open Systems in Manufacturing', dept. Trade and Industry, London, UK.

11 Lasher D (1992), 'A User's Standard - Why Is There An Eighth Layer?', Fieldbus '92, Institute of Measurement and Control and ISA International - England Section, London, November, (no page numbers.).

12 Anon. (c1991), 'FIP technical description', Club Fip, 3 bis, Rue de la Salpetriere, Nancy, France.