iCC 1996

3rd international CAN Conference

in Paris (France)

Sponsored by

Motorola Semiconductor
National Semiconductor
Philips Semiconductors

Organized by

CAN in Automation (CiA)
international users and manufacturers group
Am Weichselgarten 26
D-91058 Erlangen
Phone +49-9131-69086-0
Fax +49-9131-69086-79
Email:headquarters@can-cia.de
URL: http://www.can-cia.de

Kingdom Founder -
A Tool for Building CAN Systems

Daniel Berglund, KVASER AB

The design of a CAN based system can be a complex task. Several issues need to be addressed: choosing the right modules, verifying that the chosen modules really are able to communicate, assignment of bus identifiers, verifying that the real time requirements of the system are met, selecting correct bus parameters, and so on.

Kingdom Founder is a graphical design tool which aids the system designer in this process. Fetched from a database or defined on the fly, the modules and the CAN bus are drawn on a worksheet. The information structure of the system, as well as several module and system parameters are then defined. The system designer can then use the program to check for errors or potential problem sources, produce documentation and generate system startup information and code skeletons for the modules.

Introduction

Building distributed CAN based systems using custom-made or off-the-shelf modules can be a tedious and time-consuming task. The system builder must address several issues:

- The system must be electrically "well-connected", i.e. a suitable cable must be chosen, the bit rate must be set to a correct value, the bus must not be overloaded, etc.
- The modules shall be assigned a range of identifiers to use
- The real-time performance of the system must be analyzed and verified. This is closely connected to the task of choosing identifiers, as the identifiers usually governs the access to the bus.
- The system must be thoroughly and clearly documented, and outdated information must not appear in the documentation
- Any custom-made modules in the system must be constructed using accurate information about the other modules in the system.

The system builder needs two things in order to succeed:

- A decent higher level protocol, with support for designing modules separately from the system, and with support for configuring such modules in order to adapt them for a specific system, and
- A tool that aids in the process of choosing suitable modules, analyzing the system with respect to electrical, real-time, and data-flow parameters, and generating most of the system documentation.

The higher layer protocol best suited to fulfill the above needs is CAN Kingdom, which is now supported by a tool, Kingdom Founder. Other protocols compatible with CAN Kingdom, like DeviceNet and J1939, are supported and such modules can be integrated into the systems.

Kingdom Founder comes in two versions:

- *Kingdom Founder for Modules*: Each module is defined with respect to hardware parameters and what pieces of information ("variables") it can receive and/or transmit. Each variable is assigned a name, a data size, a data representation and several other attributes. Also, any pre-defined CAN Kingdom structure, such as forms, documents, or folders, are defined here. Documentation and a "C" code skeleton for the module may be generated directly. When the definition of the module is complete, it is stored in a database that is later used by...

- *Kingdom Founder for Systems*. Here, the work is typically divided into three separate phases.
    1. The system builder defines a number of "dummy modules" each one representing a module in the real system. Then, the variables in each dummy module is defined but as opposed to a "real" module, no CAN Kingdom structures are defined; a dummy module contains only variables. The transmit dummy variables are then connected to receive variables in other modules. For each connection a "period" and a "deadline" is given; these are the base for a later analysis of the real-

time performance. The modules are also connected electrically by choosing cables from a cable library and setting the length of the different cable segments.

2. The "dummy" modules are substituted by suitable "real" modules from a database (constructed by Kingdom Founder for Modules). The variables in the dummy modules are mapped onto variables in the real modules and the system is analyzed again.
3. The system is "built", that is, the appropriate CAN Kingdom configuration is generated, either manually or automatically by the program. When the system is built, it can be analyzed from different points of view and several types of reports can be generated. For example, a bit-timing analysis calculates the maximum usable bit rate for the system, and also calculates the appropriate CAN controller register settings for each module. A real-time analysis reveals if there are any bottlenecks in the system and ensures that each variable can be guaranteed to arrive within its specified deadline. The information gained in this analysis is used when choosing the correct identifiers.

It is also possible to define different phases of the system, each phase having its own set of variables and variable connections. This can be used for defining and analyzing, for example, a system with a startup, a normal, a maintenance, and a low-power phase. The system configuration during startup may be analyzed without interference from the normal run-time configuration.

Kingdom Founder supports compressed envelopes and letters[1], allowing for support of J1939.

Several kinds of reports can be generated by the built-in report generator. The reports can be printed directly or be saved in RTF format, enabling the user to import them to WordPerfect, Microsoft Word. For each module, reports containing variable lists, form lists, document lists, detailed list of forms, and other can be generated. System-level reports include real-time performance, bit timing analysis, producer/consumer relationships, King's List, a system consistency check, a list of used CAN identifiers, action/reaction relationships, etc.

Computer oriented output are currently a variable database in canDB format (for use by the CANanyzer by Vector Informatik), and "C" and Pascal (dearly wanted by those who are writing e.g. service tools in Delphi) header files defining the variables. A code skeleton for the module can also be generated (see below).

Computer requirements are 486-class processor, 8MB RAM (more is recommended, as usual), 10MB available disk space, and Windows 3.1 or later (Windows 95, Windows NT.)

Code generation

Mainly intended for development purposes, the code generator in Kingdom Founder generates a code framework into which the module designer can hook his application-dependent code. The application programmer sees a coding model resembling the event-driven paradigm usually found in today's user interfaces. For example, the arrival of a CAN message will trigger a sequence of "Receive Variable" event handlers, each one written by the application programmer. This procedure, commonly referred to as *message-cracking*, can be controlled in great detail from Kingdom Founder, enabling the module designer to provide a short execution path to time-critical message handlers.
The system builder can use Kingdom Founder to generate the required module adaptation code, which, when CAN Kingdom is used, is placed in a special module in the system. When the system starts this module (the King) initializes the other modules, thus connecting the information producers to the information consumers in the system. The King is also used to check that the system is intact, i.e. that all modules expected to be present in the system really are alive and well. This configuration may, at the system designer's option, be extended into the running phase of the system, so as to handle module hot-swapping (replacement of faulty modules with off-the-shelf spare parts), on-line and off-line diagnostics, data conversion to make apparently incompatible modules fit into the same system, data and error logging, and several other tasks.

The code generated by Kingdom Founder is linked with a special code library which is written specifically for the HLP used. Switching HLP at design time may be as easy as just linking with another library and writing a few special routines to handle with the peculiarities of the new HLP (like "duplicate MAC ID check" in DeviceNet).
Code can also be generated for a CAN Kingdom King. The generated code contains all required information to setup the system at run time.

[1] This is CAN Kingdom terminology for packing data into the CAN arbitration field. This is used by e.g.

Support for other higher layer protocols

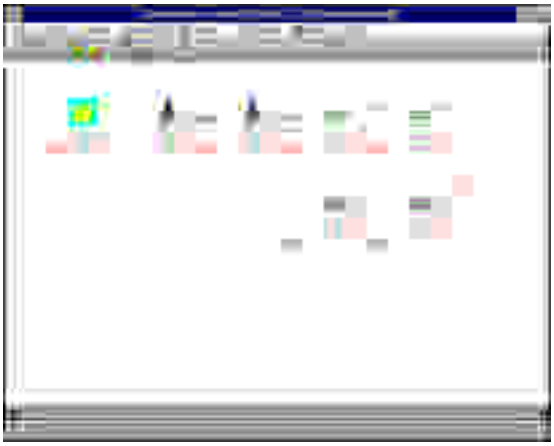
Today, there are several higher layer protocols on the market: DeviceNet, SDS, J1939 just to mention a few. During our work with Kingdom Founder we have realized that the basic model we have chosen for describing modules and systems lends itself very well to systems using other higher layer protocols than CAN Kingdom. Of course, a system builder using a less flexible and powerful HLP may not utilize the full power of CAN but it is nevertheless possible to integrate modules using different HLPs into the same system, provided the HLPs used are compatible.

Kingdom Founder facilitates the building of such heterogeneous systems by allowing the module constructor to loosely map features found in CAN Kingdom onto the features used by other HLPs. For example, attributes in DeviceNet can be interpreted as CAN Kingdom variables. A profile (DeviceNet, J1939 etc.) can be viewed as a set of specialized CAN Kingdom Documents. The system designer's chief task, namely defining the flow of information in the system, is independent of the HLP used. Code and report generation and error checking are written specifically for each HLP.

User Interface Generation

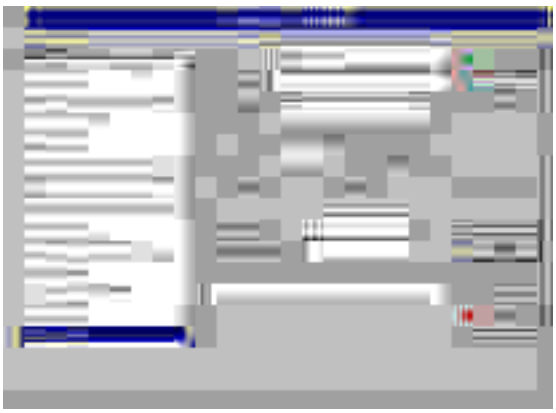
During the development of a module or a system, there is generally a need for interactively testing the newly produced software. Using the module descriptions, Kingdom Founder can generate e.g. dialog boxes for setting parameters in a module, graphical interfaces to make small, well-defined changes in the system configuration and so on. These user interface elements are generated in a special script language and fed into another (Windows based) tool which monitors the CAN bus. In effect, simple service tools can be generated in this way.

Some screen snapshots from Kingdom Founder, version 0.51:

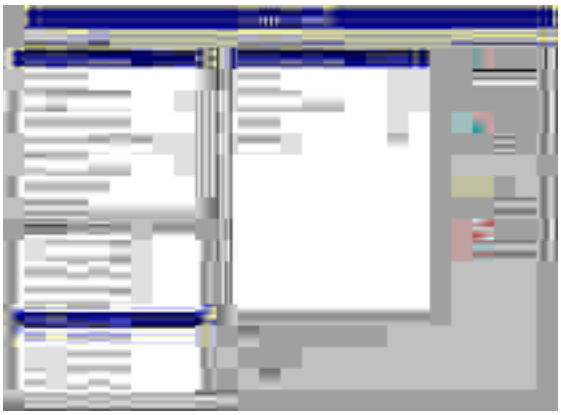


The system view.

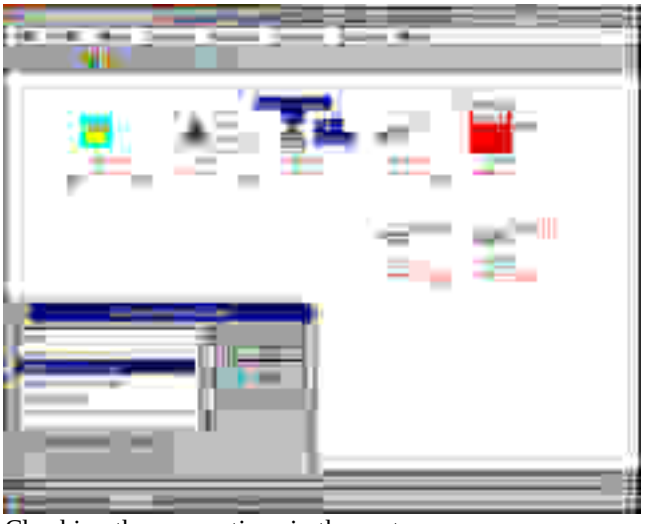
Defining a module.



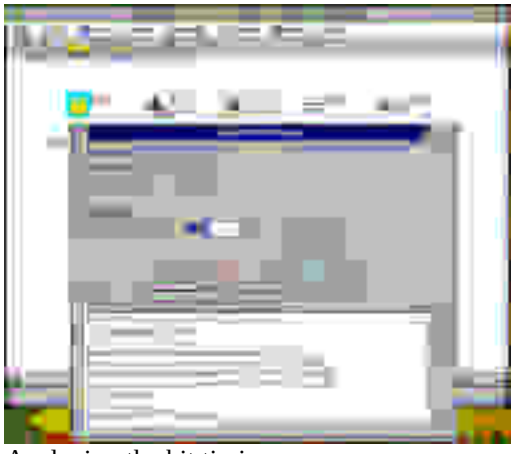
Configuring a variable.



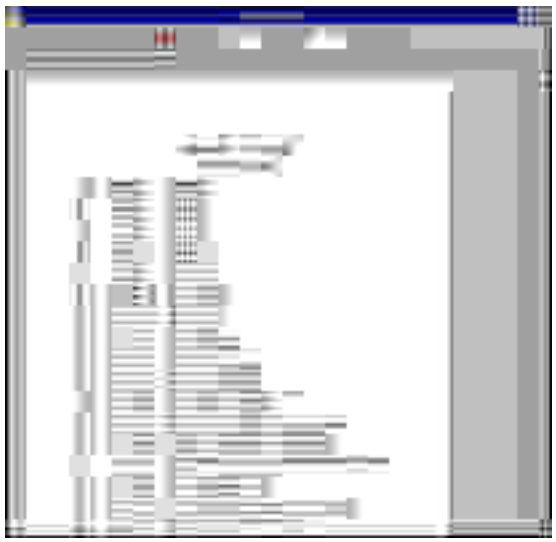
Defining the layout of a Form.



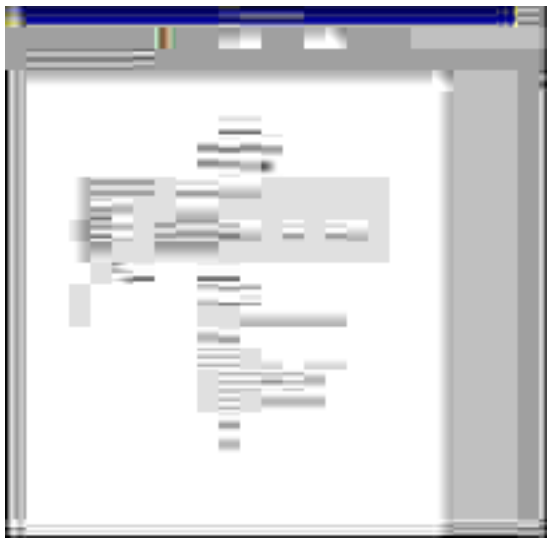
Checking the connections in the system.



Analyzing the bit timing.



Generating a list of Forms.



A detailed list of the Forms.