Development of PC-based HIL Simulator "CRAMAS 2001"

●Takeshi Fukazawa ●Norio Yamada ●Takeshi Yasuda

●Naoya Kamiyama

Abstract

In recent years, the automobile industry has been actively promoting the development and evaluation of products in virtual space to reduce development time and high costs for repeated prototypes. The utilization of 3D-CAD and computer simulation is one facet of this development. In vehicle control system development, the demand has been increasing for Hardware-In-the-Loop Simulators (HIL Simulators), which simulate the behavior of a given system in real time.

In 1996, as automobile electronic control units (ECUs) became progressively higher in performance and highly complicated, a workstation-based HIL Simulator for an Antilock Brake System (ABS) ECU was developed at our company to reduce the number of design/evaluation man-hours and to improve product quality. Since then we have expanded the field of its application to engine control ECU and automatic transmission ECU, and have developed a personal-computer-based HIL Simulator to reduce size and cost. At the same time, HIL simulator usage has grown within the company, and some have been sold outside our company.

To create a support tool that can be used not only in the automotive field but in a broad range of other fields, a high-speed, multipurpose, expandable system named CRAMAS 2001 (ComputeR Aided Multi-Analysis System 2001), which is based on a standard PC and PCI bus, has been developed using the knowledge gained during the development of HIL Simulators, and will now be introduced in this paper.

1. Introduction

In recent years the automobile industry has been utilizing tools such as 3D-CAD and simulation systems to shorten development periods and reduce prototype costs. In the field of vehicle control development, the needs for a Hardware-in-the-Loop Simulation (HILS) are increasing to evaluate the behaviors of a given system virtually and in real-time. To improve quality and reduce the number of design and evaluation man-hours that accompany the development of high-performance, large-scale electronic control units (ECUs) for automobile electronic control equipment, our company developed a simulator, an ECU design support tool. Since then, the HILS has been used effectively in our ECU evaluation process.

With the aim of creating a support tool that can be used not only in the automotive sector but in a broad range of other fields, we utilized know-how that was acquired during the development of the aforementioned simulators to develop the CRAMAS 2001 (ComputeR Aided Multi-Analysis System 2001), a high-speed, multipurpose, expandable system that is based on a personal computer PCI bus.

2. Development history and aim

In fiscal year of 1996, our company developed a HILS system that utilized an engineering workstation (EWS) as the host machine which was equipped with I/O boards based on the Versa Module Europe (VME) bus standard. But since the system utilized an EWS as the host machine, it was expensive and never really came into standard use. So in FY 1997, we developed the CRAMAS, a HILS system that utilized a personal computer as the host machine and that was equipped with I/O boards based on the CompactPCI (Peripheral Component Interconnect) standard. By using a personal computer as the host machine and adopting a commercially available Intel CPU board for the instrument rack side, we were able to create a HILS system that had superior cost performance.

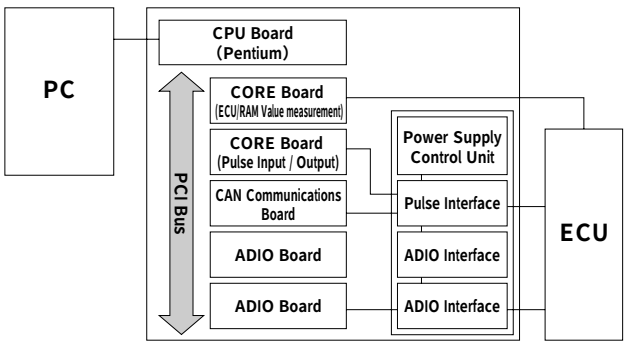

Fig.1 System configuration of CRAMAS 2001

To be able to adapt to expanding uses and changing needs, however, the CRAMAS required that a special I/O board be designed; and consequently, the lead time until a single new system was ready for start-up became longer.

Furthermore, several obstacles were in the way. For instance, a PC DOS (IBM) operating system was used for real-time processing during simulation; consequently, the full performance of 32-bit architecture could not be utilized. To eliminate such problems and construct a more timely HILS system, we have developed the PCI bus based CRAMAS 2001.

3. System configuration

3-1 Hardware overview

Figure 1 shows an example of the system configuration of a CRAMAS 2001 system that was made for a powertrain ECU. Broadly speaking, the system consists of two parts: a personal computer for setting up simulation conditions, taking measurement data, and monitoring data; and the CRAMAS main unit for performing simulation operations and processing signal input and output in real-time. These two parts are connected with an Ethernet cable, which enables real-time data measurement and monitoring.

A PCI bus is used for the bus system so that the system could be set up by selecting motherboards, CPU boards, and instrument racks that have a commercially available PCI bus (Figure 2). When this report was written, CPU boards having a clock speed of 1.5 GHz could be obtained. In the future it will be possible to improve performance at a lower cost as the clock speed of CPUs in commercially available personal computers continues to improve.

The instrument board rack that was adopted has a backplane based on the PCI Industrial Computer Manufacturers Group (PICMG) standard, making it possible to mount six PCI-bus-standard I/O boards (hereinafter referred to as "PCI instrument boards").

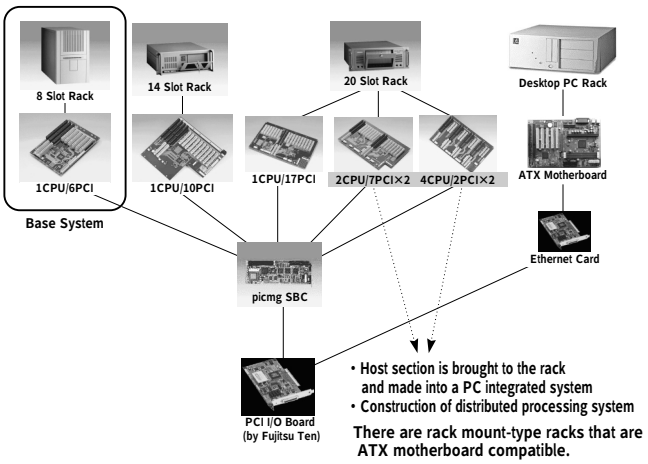

Fig.2 Utilization of commercially available products

One advantage of the PCI instrument board is that commercially available items can be easily used if they meet PCI bus standards. And for interface circuits that are electrically compatible with the ECU, we developed a compact interface rack that can be inserted into the 5-inch bay installation space of the PCI instrument rack, thus reducing size and conserving space.

3-1-1 CORE board

To be able to meet a variety of user needs within a short period of time, we developed a CORE board that can change the circuit configuration using a field programmable gate array (FPGA).

As shown in Figure 3, a CORE board consists of a digital signal processor and analog signal processor. CPU firmware and FPGA logic programs are downloaded from the host PC, making it possible to quickly develop and provide various types of I/O hardware for users.

The digital signal processor performs periodic signal measurement, periodic signal generation, and other special types of signal processing, using an FPGA and executing the peripheral functions of a microcomputer.

The analog signal processor utilizes a D/A converter and A/D converter, as well as an FPGA to achieve high-speed microcomputer input-output. And by outputting desired waveform data to the D/A converter, it is possible to achieve high-speed generation and output of various waveforms that fit the given purpose. A CORE board has a two-story configuration consisting of a digital signal processor and analog signal processor (Figure 4). If the analog signal processor is not needed, the second level can be disconnected, producing a configuration that consists only of the digital signal processor. Moreover, an FPGA ranging from 50,000 to 200,000 gates can be selected, making it possible to create a configuration that meets the given need.

Table 1 shows the hardware specifications of the CORE board.

Examples of special signal processing boards that utilize the CORE board are described as follows.

Table 1 CORE board specifications

Micro computer	Manufactured by Toshiba
	TMPR3927F
	Built in PCI Bus Interface Clock speed: 133MHz external bus 66MHz
FPGA	Manufactured by Altera Japan
	2 × EPF10K130E 130,000 gate 1 × EPF10K50S 50,000 gate ※For standard installation ※Expandable up to 200,000 gate FPGA
・Digital Input：Output: 56ch, 3.3V ・Analog output: ＊Fast CH 2ch,±5V Resolution 12Bit 5MHz ＊Other 6ch,±5V Resolution 12Bit 10KHz ・Analog input: 8ch,±5V Resolution 12Bit 10KHz ・External factor interrupt Input: 4ch, 3.3V ・Serial Communications: 2ch	

① Input-output signal for engine control

Engine control systems handle a large number of sensor and actuator signals that are difficult to process with ordinary A/D, D/A, and digital I/O. Typical signals include the following: engine revolution angle signals, vehicle speed signals, ignition signals, knock signals, and fuel injection signals.

The digital signal processor executes signal "generation" processing for engine revolution angle signals and vehicle speed signals, and "measurement" processing for ignition signals and fuel injection signals. Each processing is achieved through FPGA logic. As for the knock signals, "knock signal generation timing" is generated in the digital signal processor; then this signal is transmitted to the analog processor via the communications bus between the FPGAs, and sine wave data is output to the D/A converter.

② Internal data measurements of ECU microcomputers
The RAM values of microcomputers used for ECUs are measured by using the communications function of a Non-Break Debug module (NBD), a debugging unit with built-in microcomputer. The special protocol and high speed required for communications are accomplished by means of FPGA logic, and desired address values are automatically measured.

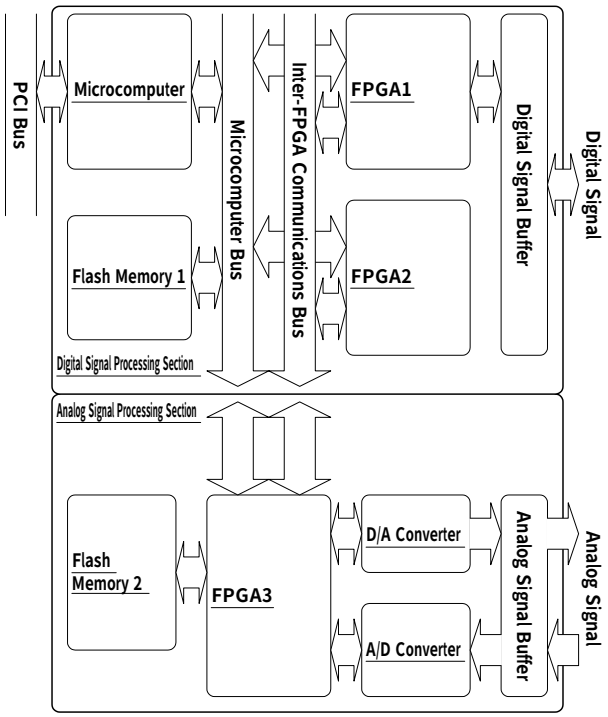

Fig.3 Block diagram of CORE board

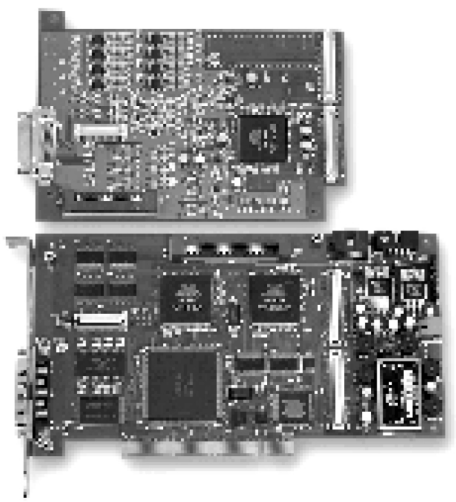

Fig.4 CORE board appearance

3-1-2 Interface

For the EWS-based CRAMAS and the initial CRAMAS interface circuit unit, a special rack (interface rack) on which multiple interface boards were mounted was connected by cable to the PCI instrument rack.

A disadvantage of this configuration was that a large number of cables were required to connect the PCI instrument rack to the interface rack, which made the overall system more complex; moreover, since two racks were needed, a large amount of space was required. Thus, for the CRAMAS 2001 system, we developed a new interface rack that integrated these racks, making the configuration more compact, space-saving, and simple for users.

The features of this new interface rack are described below.

Fig.5 Interface appearance

・Installation space for 5-inch bays

To integrate the PCI instrument rack and the interface rack, we developed a small interface rack that was capable of housing up to three interface boards in two 5-inch bay slots of the PCI instrument rack. Compared to the conventional configuration, the volume was reduced by approximately 87.5%, a substantial improvement in compactness. Furthermore, the number of I/O ports was increased by approximately 140%, and the convenience was improved due to the interface circuit automatic setup function and the interface code identification function (described later). Compactness and high performance was achieved from the adoption of a two-story construction/six-level circuit board per interface board and two-sided, high-density mounting.

To solve the problem of heat generation which occurs when high-density circuits are contained in narrow enclosed spaces, two high-output electric fans are installed at the back side of the interface rack, providing forced air cooling. Overheat detection and overheat protection functions are provided by the temperature-sensing IC. (When the inside temperature reaches 70°C, the overheat indicator lights up; and when the temperature reaches 80°C, the circuit power supply is forcibly turned off.)

・Interface circuit automatic setup function

In automobile ECU interface circuits, there are input interfaces where the signal is received by battery voltage pull-up, by 5V pull-up, and by GND pull-down. Thus, to enable usage for a variety of purposes, previous interfaces were configured such that mechanical switches could be switched for each I/O port. But since this method required manual setup, it took time to set up the interface circuit and it was necessary to adjust switches when changing an ECU for evaluation.

With the newly developed interface rack, the user can selectively change these circuit settings via the CRAMAS graphical user interface (GUI); moreover, there are functions that can automatically set up the circuits according to the nature of the condition settings. The burden on the user has thus been greatly reduced.

Signals that are output from the digital port for evaluation during normal measurements are used as signals for the interface circuit setup before measurement begins. This method makes it unnecessary to prepare separate signal lines for interface circuit setup, thereby reducing cost.

This system is configured with just two control lines for a standard digital output port. First, the S signal causes a switch from the normal signal path to the circuit setup latch circuit. Then the interface setup signal is output from the digital output port and the R signal becomes active, setting up the interface setup signal for the latch circuit. The interface circuit then switches according to this latched signal (Figure 6).

With this newly developed interface rack, the signal indicating whether the interface circuit settings are correctly latched is fed back to the normal digital input port; thus, the circuit settings are also checked.

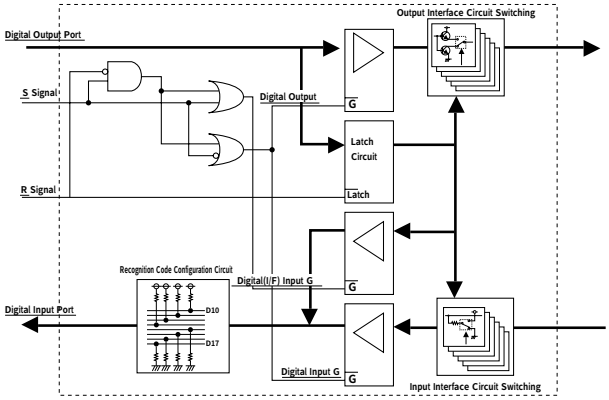

Fig.6 Block diagram of interface circuit switching unit

・Interface code identification function

Multiple types of the same digital input-output interface board can exist, depending on the ECU and needs of the customer. If the incorrect interface board is selected or if the incorrect settings are assigned, the system and/or ECU may receive burn damage. Thus, to prevent improper operation, we decided to add a function that identifies the interface code.

A gate circuit is installed for a normal digital input port, creating high impedance when off. When the digital port side is pulled up or down, the fixed data is input when the aforementioned gate is off. This input is then used as the interface identification signal. This setup makes it possible to prevent usage that will lead to the burn damage mentioned above. Specifically, at the stage prior to the aforementioned interface circuit setup sequence, an interface code reading sequence (S signal: H; R signal: L) such as that shown in the timing chart of Figure 7 has been added. The logic to shut off the aforementioned gate circuit during the sequence (Figure 6) is added in order to output the interface code.

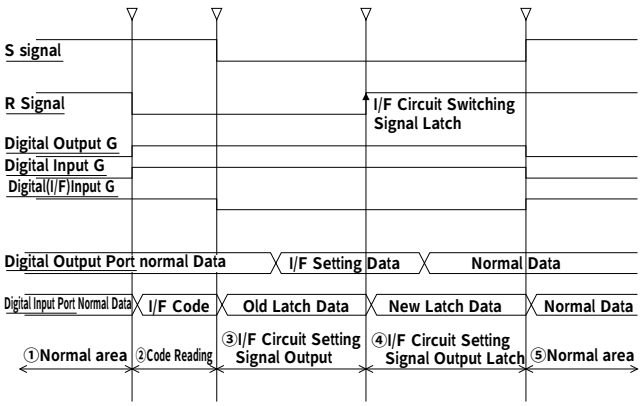

Fig.7 Interface circuit timing chart

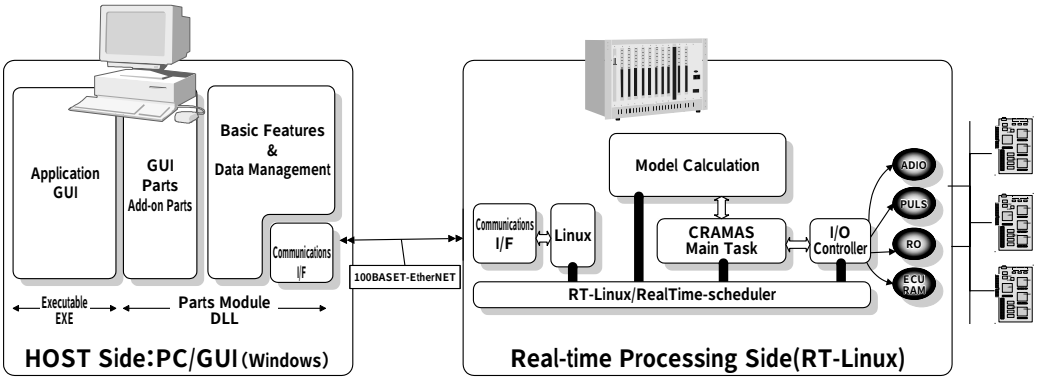

Fig.8 Software configuration

3-2 Software units

The CRAMAS software consists of the following three units (Figure 8):

(1) Graphical user interface (GUI)
(2) Dynamic link library (DLL)
(3) Real-time execution module

The GUI and DLL run on the Windows operating system of the personal computer (PC) that acts as the host machine. The user's condition settings and measurement data are expressed graphically on this host PC.

The real-time execution module runs on the RT-Linux operating system of the CRAMAS main unit and processes tasks in real-time at the specified cycle. The computed values or measurement data are sent to the host PC, allowing the user to monitor the measurement data in real-time. Commands can also be transmitted at the desired timing, enabling the user to control the status of simulation processing.

3.2.1 Application configuration of host PC

Applications of the CRAMAS host PC are distributed to the graphical user interface (GUI) and dynamic link library (DLL). The GUI is the part that allows direct dialogue between the user and the system, making it possible to perform complex system operations in accordance with the user's intuitive actions. Operations are also carried out on the Windows operating system, which users are acquainted with. In general, the CRAMAS system includes the following four applications:

(1) Operating environment control
(2) Pattern signal editing
(3) Real-time monitoring
(4) Measurement data analysis

The CRAMAS 2001 strengthens the condition set up and the measurement data analysis features that are used in detailed debugging operations for the control and measurement-targeted systems in particular. It is also equipped with functions that enable the user to customize the windows on the host PC for a variety of uses. (CRAMAS functions are explained in the next chapter.)

The DLLs are components of the CRAMAS system's fundamental functions, including the control functions of the user setting data that is operated on the GUI, as well as communications functions with the real-time execution module. This structure makes it easier to expand and customize the system. GUI applications utilize the basic functions of the DLL via the application programming interface (API), giving practical effect to the software functions of the CRAMAS.

3-2-2 Real-time execution module configuration

Broadly speaking, the software of the CRAMAS system's real-time execution module includes that of the following (Figure 9):

(1) PCI device drivers
(2) I/O value conversion and delivery module
(3) Model calculation module
(4) Main control module

Items 1 and 2 above are fundamental functions of the CRAMAS system, while item 3 is a program consisting of commercially available modeling tools.

The CRAMAS corresponds to MATLAB/Simulink (by MathWorks) as a modeling tool for simulation. With this model, the target model is constructed as a graphical block diagram and we can easily produce the C code program from it on the host PC.

In order to use this in the CRAMAS, the execution file that is made by the compiling of the C code program is downloaded to the real-time execution module, and it is linked with the CRAMAS modules of items 1 and 2. Some simulator systems of other companies utilize techniques that confuse the model and the hardware I/O processing. The CRAMAS, however, does not use such techniques; rather, it separates physical model and hardware I/O processing to enable system expansion, wide-range usage, and portability.

Item 4 is the module that controls all modules. It receives measurement condition setting data from the host PC, analyzes and processes real-time commands during simulation, and transmits measurement data to the host PC.

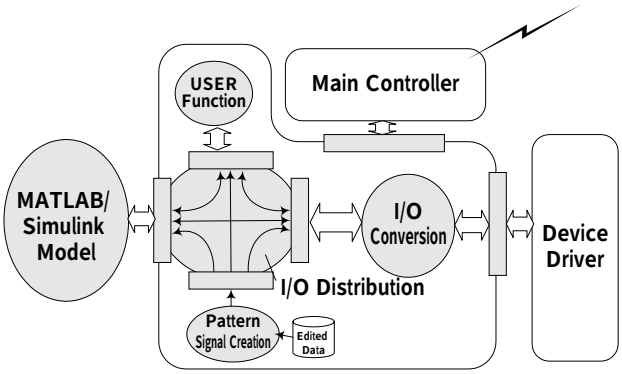

Fig.9 Real-time execution module configuration

<<Introduction of RT-Linux>>

When the PC-based system was being developed, PC DOS (by IBM) was adopted as the real-time processor's operating system (OS) in the original CRAMAS. It was adopted because it had several advantages over a real-time operating system (RT-OS): it was less expensive, easier to use, easier to develop, and could be used with various past tools and development environments.

But as the CRAMAS system started being used in various ways, and requests for support for complex, large-scale systems and high-speed processing began to emerge, it became difficult to meet such needs with the conventional 16-bit, single-task DOS operating system.

With the CRAMAS 2001, we considered introducing a RT-OS for the real-time processor, but since a commercially available RT-OS is much more expensive when a development environment is included, the price of the CRAMAS system itself would be high.

Thus, we turned our focus to RT-Linux, which adds real-time performance to the original Linux, which is rapidly becoming popular in the area of server machines.

Table 2 DOS / RT-Linux benchmark results

Item	RT-Linux	DOS
①Model Calculation Time	280 μ sec	560 μ sec
②Minimum Task Cycle	35 μ sec	200 μ sec

※Using CPU equivalent to Pentium II 266MHz

RT-Linux makes extremely high-speed processing possible without sacrificing any of the original Linux functions. Furthermore, since all source codes are open to the public and since modification and redistribution are not restricted, it has gained attention as a RT-OS for installation of various systems.

The software of DOS-based CRAMAS was ported to that of RT-Linux-based CRAMAS, and a benchmark test of DOS and RT-Linux was conducted. The results are shown in Table 2. The results verify that RT-Linux, in comparison to DOS, achieves much higher performance for both the operation period and the minimum task cycle.

Before RT-Linux could be introduced, however, another problem needed to be solved.

As previously mentioned, the CRAMAS configuration is a cross environment that is between two different operating systems that are divided into the host PC (Windows) and measurement system (RT-Linux). With the CRAMAS system, a compiler is needed to construct RT-Linux program codes on the Windows operating system since model compiling is done on the host PC. Using the source code of the GNU-gcc compiler, which is also open source, we were able to create a Windows-Linux cross-compiler.

4. Application functions

This chapter will introduce the major functions of each application of the CRAMAS 2001.

4-1 Operating environment control (Figure 10)

This basic tool of the CRAMAS system starts up related applications and sets up, registers, and controls various types of data, including measurement and simula-

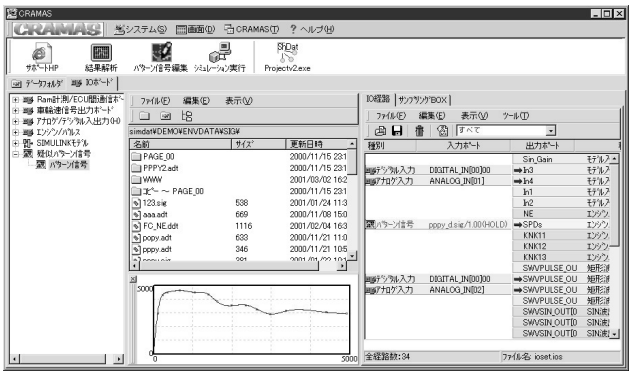

Fig.10 Operating environment control

tion conditions. The set up and registered data can be saved as test pattern data files, and they are called at the required moment to enable reproduction of simulation.

Furthermore, multiple test patterns can be registered as batch data for continuous automatic operation. These data groups can also be managed using folders according to the measurement-targeted system type.

This application is used to set up and verify the I/O board of the real-time execution module and to download CORE board firmware.

4-2 Pattern signal editing (Figure 11)

Through simple mouse-executed plotting, this tool can be used to edit time series data in order to create signal waveforms artificially in the real-time execution module. Linear interpolation, spline interpolation, digital waveform enlargement/reduction, editing while making comparisons to existing waveforms, and numeric input through PC key operation are also possible. CSV-type data point sequences can be freely imported. With Microsoft Excel, for instance, data point sequences can be created, imported, and even attached in reverse.

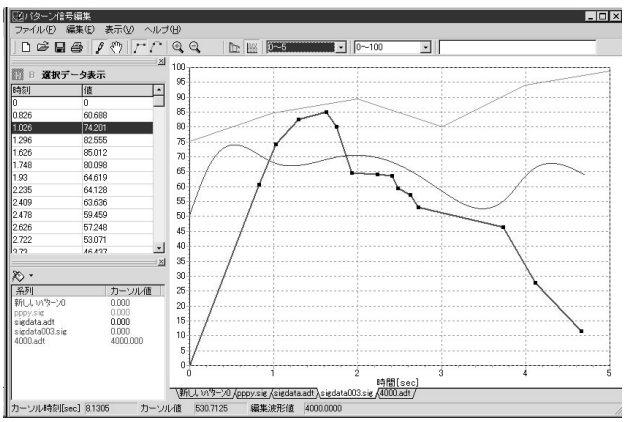

Fig.11 Pattern signal editing

Furthermore, measurement data that is imported using other measuring instruments can be converted to CRAMAS pattern data and utilized.

In the real-time processor, 20 megabytes of memory space (standard) is reserved for pattern signal generation. With spline waveforms, a maximum of four minutes' worth of waveforms (for registration of ten signals and generation period of 1 msec) can be generated to expand in the memory for all time value data during start-up process. For linear interpolation and digital signals, it is possible to generate waveforms equivalent to up to 35,790 minutes and 87,300 modified plot points (under same conditions).

4-3 Real-time monitoring (Figure 12)

During simulation this tool makes it possible to display real-time charts of measurement data transmitted from the real-time execution module and to handle executable parameters for simulation.

It primarily has the function-based control windows described below.

① Real-time chart window

Multiple registered data are displayed in real-time on the same chart. Items such as auto scale mode and physical/logic value mode can be specified for each registered data display.

② Virtual instruments (control panel)

The user can freely arrange analog meters, sliders, and other control parts on the panel, and freely construct measurement and control windows to fit their purpose.

With CRAMAS 2001, commercially available ActiveX parts as well as conventional parts can be incorporated. This system makes it possible to provide control parts that meet users' needs and provide such parts in a short time.

③ LED checker

This tool can display RAM values of the ECU microcomputer as bits in LED style or physical quantity values.

With the development of CRAMAS 2001, each of the control windows is a separate module that is linked during execution. This makes it possible to freely and quickly change the tool configuration, expand functions for the

future, and customize to meet the user's needs. We are now planning to develop links with 3D-CAD and other data analysis tools since there are strong potential needs for measurement data visualization and customization.

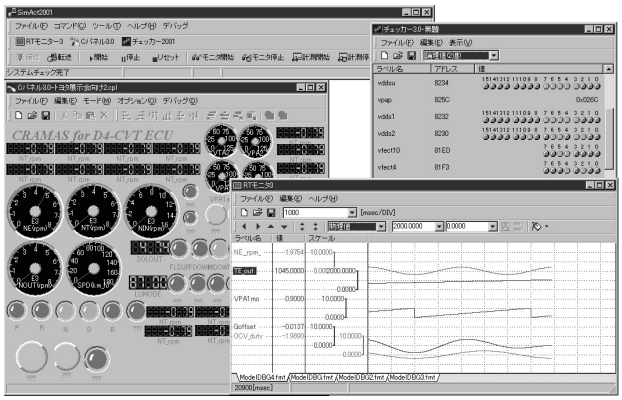

Fig.12 Real-time monitoring

4-4 Measurement data analysis (Figure 13)

This tool is prepared to analyze measurement data in detail.

It includes various analysis functions, including the capability to enlarge specified areas, display cursor values (display values on a table at desired times) by means of mouse operation, finding and displaying the time and value difference between two points, and conduct searches for measurement data to pick out the identical condition date.

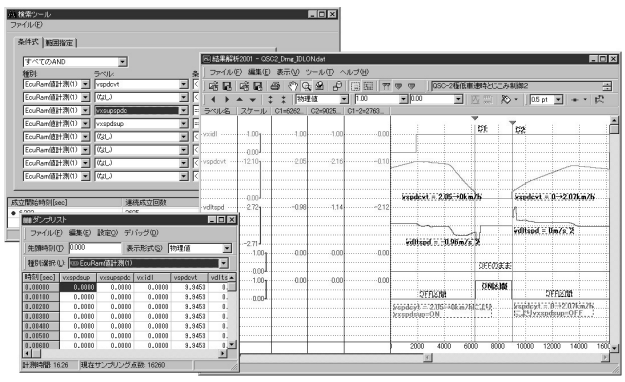

Fig.13 Measurement data analysis

Furthermore, the control panel has functions such as "Undo/Redo" of enlargement/reduction operations, display of scale maximum/minimum values based on the display mode (physical values/hexadecimal, decimal, and binary logic values), and joint use of scale format with the real-time monitoring window.

The functions as a document creation support tool were also strengthened.

Designation of line widths, display of measured point plots, designation of measured point-to-point graphing, affixing of desired text labels (comments) to charts, and printing functions have been improved greatly. The addition of these functions has made it easier to create debugging reports and other working level documents.

5 Application examples

This chapter will describe system examples utilizing CRAMAS.

5-1 CVT system (Figure 14)

At the development and design stage of the "ECU for direct-injection 4-cylinder engines, electronic throttles, and continuously variable transmission (CVT)", it is extremely difficult to evaluate the ECU by using only conventional testing and evaluating equipments, because these equipments can not produce complex and continuously variable signals corresponding to the behavior of the CVT.

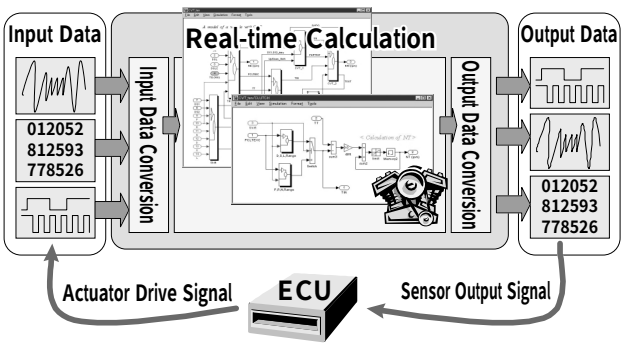

Fig.14 CVT system

Thus, by modeling the behavior of an actual vehicle (focusing on the CVT portion) under physical laws and simulating in real-time using the CRAMAS, it has become possible to conduct various evaluations of the ECU under an environment closely resembling, an actual vehicle test. Furthermore, since changes in vehicle behavior and ECU's internal RAM values, as well as changes in ECU's control data and I/O signals, can be simultaneously measured and then analyzed on the same window, our company has been able to reduce by up to

60% the number of man-hours needed for designing and evaluating ECU's control programs.

5-2 Inspection system in the ECU manufacturing line

By adding "inspections using CRAMAS" to our company's ECU manufacturing inspection process, it has become possible to conduct both dynamic and static inspections under simulated vehicle environments rather than conduct conventional partial static inspections. In this way, the ECU is inspected while the input-output signals of a vehicle model and the ECU are mutually exchanged and while the conditions of an actual vehicle are successively simulated, including the following conditions: "Engine off condition → Stator on → Rise in engine revolution and vehicle speed as accelerator is pressed."

Incidentally, this system is configured to use the CRAMAS-API from the WFLEX line control system that is utilized on our company's manufacturing line.

5-3 Power supply voltage fluctuation testing system (Figure 15)

This system uses the CRAMAS pattern signal editor or Excel to create and register power supply voltage fluctuation test waveform patterns that meet ECU test standards; and automatically operates (unattended day and night), determines acceptability, and saves data involving multiple test items. It is equipped with a special interface unit having the capability of high-speed (1msec) power supply shut off by gas-filled relay. It is also equipped with ISO 9141 communications functions that enable ECU's diagnosis codes reading and deletion. This system reduces conventional testing man-hours by up to 80%.

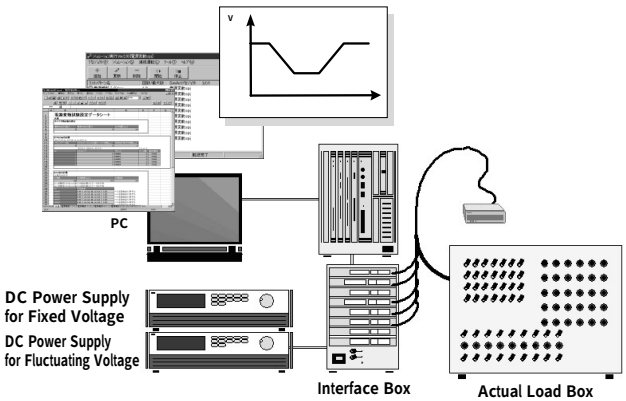

Fig.15 Power supply voltage fluctuation testing system

Incidentally, this function was adapted from the "continuous automatic operation" function that sets up, registers, and sequentially executes routine test patterns ahead of time.

5-4 FMEA diagnostic debugging (Figure 16)

This is an example in which an open ECU terminal, GND short, and battery voltage short are controlled by CRAMAS pattern signals, and the behavior of the ECU is verified. ECU's diagnostic functions such as sensor signal disconnections, solenoid failures, relay failures, and other specifications can be debugged while ECU's control is active.

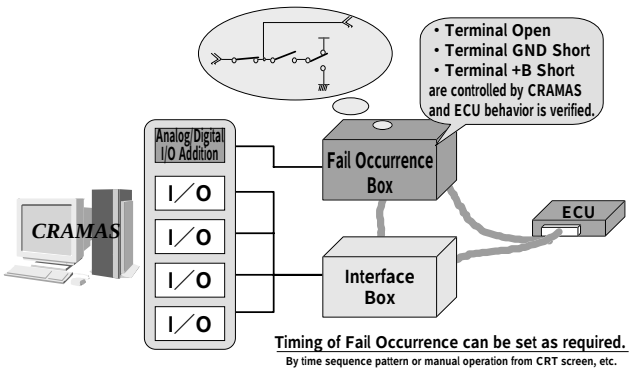

Fig.16 FMEA diagnostic debugging

5-5 CAN communications (Figure 17)

The Controller Area Network (CAN) is a communications protocol that is widely used in the automotive sector. A PCI-CAN communications board was developed for the CRAMAS 2001, making it possible to trans-

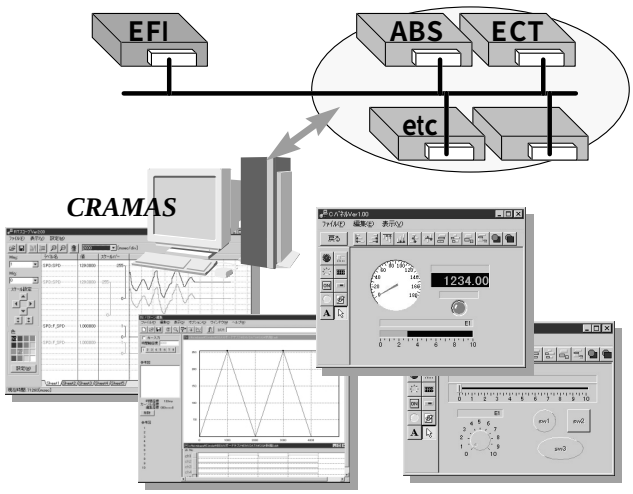

Fig.17 CAN communications

mit data to the CAN bus (node simulation) and to monitor data on the CAN bus.

6. Conclusion

Recent development work has resulted in the creation of the CRAMAS 2001, a platform system that has high speed, flexibility, and expandability, and that can be adapted to meet future needs for large-scale simulation and use with diversified systems.

It is clear that simulation technology will be widely used not only in the automotive industry but in other industries as well.

We will continue to develop more advanced functions for the CRAMAS 2001 as well as applications for a variety of industries, and to expand sales to the automotive industry and new markets.

References

1) Kaneko: Open Design No. 7: PCI Bus Details and Steps toward Application, CQ Publishing (1997).
2) Funaki: Linux Real-Time Measurement/Control Development Guidebook, Shuwa System Co., Ltd. (1999).
3) Tonou et al: Advanced Software Checker for Engine Control ECUs, Fujitsu Ten Technical Report, Vol. 15, No. 2 (1997).

Profiles of Writers

Takeshi Fukazawa

Entered Fujitsu Ten in 1980, since then working with electronic engine control devices, marketing technology. From 2000, has been involved in the development of development support tools. Currently in the SE Engineering Section Manager of CRAMAS Business Department at Vehicle Electronics Products Group.

Takeshi Yasuda

Entered Fujitsu Ten in 1983, since then has worked with the development of ECT, EFI, ABS and other vehicle control systems and development support tools. Currently in the SE Engineering Section of CRAMAS Business Department at Vehicle Electronics Products Group.

Norio Yamada

Entered Fujitsu Ten in 1976, since then has worked in the development of automotive electronic control devices and design support tools. Currently in the IP Engineering Section of CRAMAS Business Department at Vehicle lectronics Products Group.

Naoya Kamiyama

Entered Fujitsu Ten in 1992, since then has worked in the development of ABS systems and other vehicle control systems, as well as support tools. Currently in the IP Engineering Section of CRAMAS Business Department at Vehicle Electronics Products Group.