An Information-Rich Automatic Calibration Concept

LARRY C. SOLLMAN

Abstract—Calibration appears to be an area in which automation can produce a significant improvement in productivity. However, the degree of success in automating the calibration function is highly dependent on the approach chosen. This approach must account for the workload characteristics commonly found in calibration operations, yet must be flexible enough to offer a solution for new and/or unique methods of managing calibration operations. ROLAIDS (Reconfigurable On-Line ATE Information Distribution System) is one such approach.

I. INTRODUCTION

ROLAIDS was originally targeted to weapon-system-oriented test and repair operations, in which the module workload is high variety and low volume. It deals with this workload by offering an approach emphasizing low cost and high versatility through very friendly, self-configuring, instrument-based, automatic test systems. Since many calibration operations experience the same kind of workload as does the weapon-system operation, ROLAIDS should be just as applicable, maybe even more so.

ROLAIDS has two basic parts. First is the initialization protocol that automatically develops communication links among the controller and the instruments in a test system, and, as a result, provides the controller with a list of FSCM, model, and serial numbers of those instruments (FSCM = Federal Supply Code for Manufacturers). Second is the Resource Description, which is essentially a standardized electronic data sheet comprised of ranges, error limits, programming codes, etc. Thus when the controller has completed self configuration and requested the instrument-resident Resource Descriptions (plus Fingerprints, to be described later), it has all the information necessary to execute user tasks, including calibration.

II. ROLAIDS INGREDIENTS UNVEiled

A. Initialization Protocol

In the IEEE-488 bus, as in the case with any bus, each connected resource (instrument, computer, etc.) has easy access to the bus lines. Then, to prevent miscommunication, each resource must be set at a unique address and that address must match the one dictated by the controller program. Also, whether test, repair, or calibration is being conducted, the user must account for the instruments utilized and usually does so by manufacturer, model number, and serial number.

Since the goal of ROLAIDS is to make automation friendly, the initialization protocol was designed to eliminate the administrative task and to provide information valuable to the user. As previously mentioned, this information, unique to each instrument, is the FSCM, model, and serial numbers. This protocol works as follows. In each resource, the unique identifier comprised of a FSCM, model, and serial number is present. It is divided into two sub-identifiers, the first of which contains the FSCM number in the first five locations and the model number in the remaining locations. The second sub-identifier is the serial number. The characters of the identifier are allowed to only have ASCII values of 32 to 126 (ASCII-American Standard Code for Information Interchange). Through a special three-byte sequence (decimal 18, ASCII "\"", ASCII "\") with the ATN line true, the initialization mode is invoked in each resource. The mode causes each resource to assume a known listen address, 15, (the ASCII "\") and to point to the first character of the identifier (first number of the FSCM). The controller sends address 15 to cause all resources to listen and searches for the lowest character to which a resource is pointing. It does this by sending characters in a binary search fashion and monitoring the SRQ line (Service Request) for resource responses. Each resource responds by asserting the SRQ line true (low) if the controller is higher than, or releasing it if equal to or lower than, the pointed-to character. The SRQ line is a "wired-or" configuration in which any resource can assert it true. All resources must allow it to become false (high) to cause the line to be false.

The character search commences with the transmission of decimal 127. Since the identifiers can only have characters with values of 32 to 126, any resource connected to the bus will assert the SRQ line true. Thus the controller knows that a search can begin if that line is true. It starts with decimal 64 (the "@" character) and an increment of 32. With each try, the increment is subtracted from the character if the SRQ is asserted true. Otherwise, it is added. Then, the increment is halved and the new character is sent. This process continues until the character that was computed with the increment of one is sent. At this point, if there is an SRQ true assertion, the controller character is one above the lowest pointed-to character. If the SRQ line is false, the controller character equals the lowest pointed-to character. As a result, the controller knows what the lowest character is and sends it with the DI08 line asserted true (0V). (ASCII characters require only seven of the eight data lines, and DI08 was set false up to this point.) Now, it is time for each resource to determine its future listening state. Any resource pointing to a character that equals the controller character (excluding the DI08 state) remains listening; all

Manuscript received April 11, 1984.

The author is with the Naval Weapons Support Center, Crane, IN.

U.S. Government work not protected by U.S. copyright.
transmitting identifier virtually another SRQ. An assertion occurs. This condition signals to the controller that the end of the first sub-identifier (FSCM-model number) of a resource (s) has been reached. The controller notes that and starts the character search with the transmission of another decimal 127. Eventually another SRQ TRUE assertion will not occur after the transmission of the 127 character, which indicates the end of the second subidentifier, the serial number. Because an entire unique identifier has been searched, all resources except one have ceased listening. At this time, the controller assigns an unoccupied address (other than 15) to the resource by transmitting it to that resource. The resource then stores the address and ceases listening.

To start the next resource selection, the controller sends address 15 again to cause all nonprocessed resources to respond. A new identifier is searched, and a new address is assigned. This procedure continues until address 15 is sent and the first character search does not produce an SRQ TRUE assertion with the transmission of decimal 127. At this point, the controller has acquired each identifier from, and assigned an address to, each resource.

Now, the links have been developed among the controller and the remaining resources, and additional information can be accessed. By sending the three-character sequence of decimal 18, ASCII "/" and ASCII "+" (with ATN TRUE), the controller can retrieve information, such as last date of calibration, options in each resource, and compensation factors, when a resource is addressed to talk. This resource-dependent information is called the fingerprint. The last reservoir of information the controller needs for self configuration is the Resource Description which the controller gets by sending decimal 18, ASCII "/" and ASCII "d" (ATN TRUE).

B. Resource Description

Figs. 1, 2, and 3 are examples of partial Resource Descriptions (RD’s). The HP 3325A and Autek 421 will be used to show the configurability/reconfigurability of ROLAIDS, and the Fluke 8600A will illustrate that ROLAIDS can accept manual resources as well as programmable ones. The programmable resources do not have to be ROLAIDS compatible. The controller merely requests the RD of these resources from the mass storage device (disk, tape, etc.) as is the case for the manual resource.

A user may have a test requirement to produce a 2.67-V dc signal for the unit being tested. The HP 3325A synthesizer, or the Autek 421 power supply when the synthesizer is not available, will be used for the source. The user connects the synthesizer to the controller and invokes the initialization protocol and RD transfer routines (assuming the HP 3325A is ROLAIDS compatible). To produce the 2.67-V dc signal, the controller executes a series of standard steps, called an algorithm. This algorithm varies with the user language (ATLASS, BASIC, PASCAL, etc.) and the user environment. For this example, the steps in Table I apply. For either instrument, the controller is able to take advantage of the standard input of the RD to extract the unique information of a particular instrument.

Several RD characteristics will now be explained. First of all, code to be sent to the bus is shown by keyword CODE. Anything in quotes is sent as is. Where RCODE appears, the controller uses the resource label chosen by the user. In the example, the A side of the Autek supply was chosen. Next, coding between parentheses is in expression form which must be evaluated by the controller. In the case of the HP 3325A, (VOLTAGE, D.DDD) means that the voltage 2.67 is to be converted to the D.DDD format or 2.670. The Autek 421 expression, on the other hand, requires more manipulation. (VOLTAGE X 100, +--ZZZZ) requires that the voltage (in volts) must be multiplied by 100. If the product does not yield four digits, leading zeros must be added. Also, a sign must be included.

If it is desired that the 2.67 V be measured by the Fluke 8600A manual multimeter, the steps in Table II are completed.
The keyword **TEXT** signals the controller to send the information following that keyword to a display, not the IEEE-488 bus. If the meter was programmable, the keyword **CODE** would have been present and the setup and execution of the measurement would have been automatic.

Now that the ROLAIDS concept has been described, its applicability to and benefit for calibration can be explained. Obviously, ROLAIDS, as do many other approaches, uses automation to increase productivity. However, it uniquely provides, in addition, the capability to generate generic calibration programs. That is, for a class of instruments such as dc supplies, the calibration facility chooses a scenario that meets its requirements for calibration. This scenario is translated into a computer program that will be executed by the applicable calibration system. The program contains “blanks” to be filled in by the RD of the specific instrument under calibration. Thus a dramatic reduction in calibration programming can be achieved.

Fig. 4 illustrates a representative ROLAIDS calibration system. After initialization, the ROLAIDS system has found the instrument to be calibrated through its FSCM, model, and serial numbers. Also, the system requests the RD and fills in the “blanks” of the generic program, conducts the calibration, and updates the instrument’s calibration date and possibly compensation factors via the IEEE-488 bus. As a specific example, the following scenario is chosen. For each range of a dc supply, the end limits and two points to be calculated equidistant between the end limits constitute a valid calibration program. The HP 3325A has end limits of -5 and 5 V, which causes the calculated equidistant voltages to be -1.67 and 1.67 V. For each voltage, the calibration program walks through the HP 3325A RD to obtain the correct programming codes, calculates the upper and lower limits from the error limits (e.g., for 5 V, the limits would be 5.02 and 4.98 V), sends the code to the synthesizer to obtain the voltage, walks through the ROLAIDS analyzer RD (dc voltmeter) to obtain code to measure the voltage and check for accuracy compatibility, sends code to the analyzer to take a reading, fetches the reading, and compares it with the limits. It should be noted that, when the synthesizer first assumes the dc mode, the delta settling time is required which is 1.5 s. Afterwards, the 0.2-s specification is used.

After the synthesizer is finished, the Autek 421 can be connected to the calibration system and the generic program can be rerun. However, several of the differences should be pointed out. One difference is that the program will find two ranges instead of one. Another is settling time. There is no delta settling time. Thus the RD solves one of the major problems of the generic program, which is the penalizing of all instruments for the characteristics of one. Without the RD approach, all instruments would have to wait 1.5 s for the first voltage.

Although a single instrument is illustrated, the concept extends easily to test-system calibration. The ROLAIDS calibration system merely finds more than one instrument and sends/receives signals through the test-system interface rather than an instrument input.

### III. Progress

Contracts have been completed that defined the RD syntax and provided a Global design of the ROLAIDS concept. A prototype ROLAIDS calibration system has been developed for dc supplies. In addition, a ROLAIDS-compatible ATLAS incremental compiler is being developed by Lexico Enterprises for the John Fluke Company, executable on the Fluke 1720 controller. This compiler is able to communicate with the RD and is a good candidate for proving out the ROLAIDS calibration concept in an operational environment. Lastly, a task force, P981, has been formed to make instrumentation systems easy to assemble and use. ROLAIDS is playing an active part in this effort.

### IV. Conclusions

ROLAIDS offers an information-rich concept that can radically improve the productivity of the calibration operation through automation and the minimization of calibration programming. The concept applies to the calibration of existing instruments, either manual or programmable, and the future ROLAIDS instruments. The ideal workload for the future will be ROLAIDS instruments only, regardless of whether or not the end user of the instrument needs a programmable instrument, so that calibration productivity can be maximized and instrument life-cycle cost minimized.