MANAGING MULTI-VENDOR INSTRUMENTATION SYSTEMS WITH ABSTRACTION MODELS

Michael T. Lockard, James A. Garling, Jr.

EMC Corporation, Solutions Engineering Group
Irvine, CA

ABSTRACT

The quantity and types of measurements and measurement instrumentation required for a test are growing. This paper describes a methodology to define and program multi-vendor instrumentation using abstraction models in a database that allows new instrumentation to be defined rapidly. This allows users to support multiple vendors' systems while using a common user interface to define instrumentation networks, bus catalogs, measurements, pulse code modulated (PCM) formats, and data processing requirements.

Keywords: Airborne Instrumentation; Database Abstraction Models; ILIAD; TMATS; Telemetry

INTRODUCTION

Today's vendor instrumentation systems exhibit a surprisingly significant quantity of commonality in their architectures. This permitted the definitions of those architectures to be abstracted in a similar fashion and the corresponding derived objects and rules to be encapsulated into a single database. One goal in this endeavor has been to provide a common graphical interface to the user so that the operations to define all of the instrumentation components, describe the network interconnections, and define channel attributes would be identical, independent of the particular vendor's instrumentation system. Another goal in the design of an instrumentation support system has been to allow the reuse of the objects subsequently created. For example, a PCM format definition created for one instrumentation unit can be reused with another with similar capabilities. A third goal has been to provide an environment where it would be relatively easy to implement new unit functionality without having to change the programming of the graphical interface, or at least significantly reduce the amount of change required. This functionality has been implemented in the current Instrumentation Loading, Integration, Analysis, and Decommutation system, also known as ILIAD. ILIAD is a software suite developed jointly via a Cooperative Research And Development Agreement (CRADA) between the author's team and the U.S Air Force at Edwards AFB, California.

ILIAD'S INSTRUMENTATION SUPPORT

One of ILIAD's prime capabilities is supporting the definition of an instrumentation system configuration network, consisting of units, unit interconnections, the signal conditioning cards contained in each controller or remote unit, attribute setup definitions for each unit, and the channel connections and attributes. ILIAD also supports the definition of measurements for bus, analog, and digital units. This includes calibrations along with their time history usage. The user can also define PCM format layouts for the controller units which allows for unit programming as well as setup of a decommutation system from the same PCM format definition. Once the configuration is complete, the user can generate load modules for the instrumentation system and then perform the applicable load operation.

ABSTRACTION OF INSTRUMENTATION CATEGORIES

ILIAD maintains a central database repository for maintaining track of the instrumentation configuration. In addition, ILIAD has several support tables that are used to define the abstraction of multiple vendors' instrumentation systems. This information is referred to as 'support data', and contains all of the information that the programs rely on for the definition and rules of the various instrumentation. The support table at the top of the hierarchy is the one that defines the supported vendors. Following this, a definition follows for the instrumentation groups supported for each vendor. These instrumentation unit groups can be of various types. For example, a group could consist of Advanced Airborne Test Instrumentation System (AATIS) instrumentation, and Compatible Data Acquisition Unit (CDAU) controllers, Acroamatics® or Teletronics® decommutation systems, and a Heim Data Systems® Datarec® recorder. At this level of configuration management, a Unit Group includes an entire class of related instrumentation units such as controllers, remote units, and signal conditioning cards. Figure 1 shows examples of Unit Groups currently supported by ILIAD.

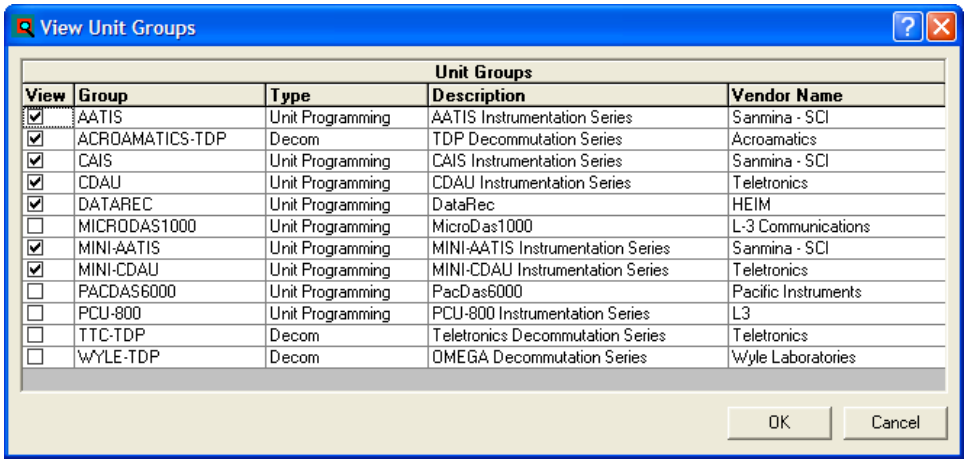

Figure 1. Instrumentation Unit Groups

DEFINITION OF UNIT TYPES

For each Unit Group, there exists a set of Unit Types that belong to the group. This represents all of the components that can be connected together or assembled together, such as control units, remote units, and signal conditioning cards. The Unit Type information includes the name and description of the Unit Type and the form factor of the unit, such as 'card' or 'box'. For each of these units, an attribute indicates whether or not samples (measurements) can be connected, and what the default sample size is (i.e., number of bits) for the measurement.

In the case of a 'box' type unit, an attribute indicates how many slots are available for signal conditioning cards. For signal conditioning cards, an attribute indicates how many channels are on the card.

Another attribute describes the type of loading to be used in programming the device, such as Common Airborne Instrumentation System (CAIS) bus, synchronous data link control (SDLC), Recommended Standard (RS) - 232, or No Load. For units that have programmable memory, the size of the memory is defined. For units that have paged memory, the size of the page is defined. For units that can have PCM formats loaded into memory, the number of PCM formats allowed is defined.

In another attribute, units that can have programmable attributes associated with them are identified. These are attributes that would be tied to a controller or to a signal conditioning card. The database also tracks the data sheet that applies to the unit so that a user can quickly access details of the selected unit. A reference to a small picture is also included in the database information for each unit.

For units that have a program generator requirement, the program name of the generator is defined in the database.

Figure 2 shows a list of units available that are members of the specified Instrumentation Group.

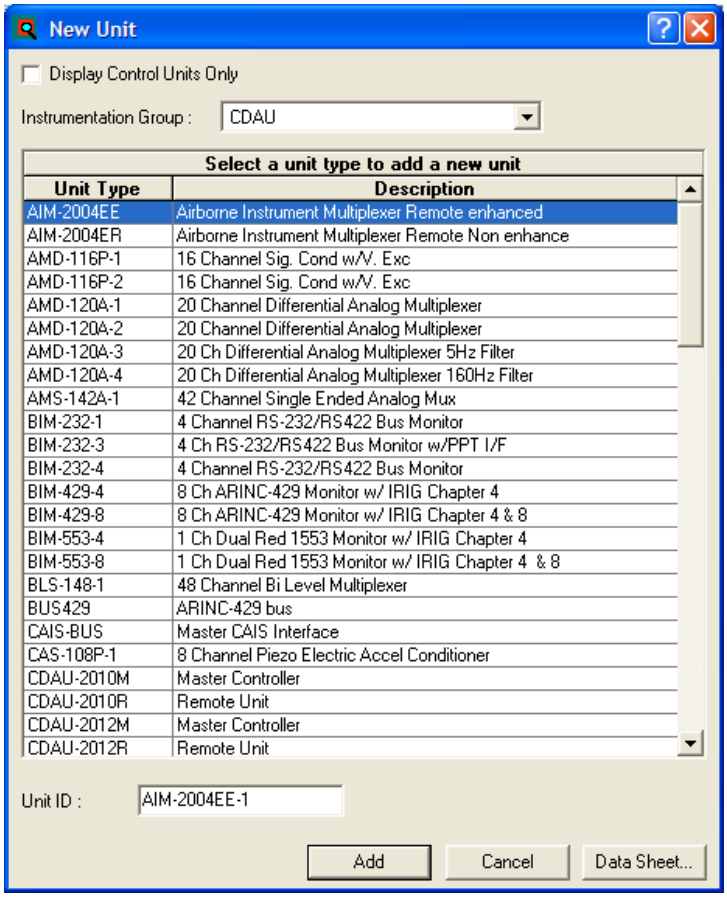

Figure 2. Units Available for the CDAU Instrumentation Group

UNIT CONNECTIONS

Once a definition of a unit has been placed in an Instrumentation Group, the rules on how the unit can be interconnected with other units must be identified. These relationships are characterized in the Connection Rules. The Connection Rules definition contains the parent to child mapping that define which units, such as signal conditioning cards, can be attached to which control units or remote units. When the user wants to add a new unit, a list of units that are eligible for selection at the chosen point in the network is displayed. Figure 3 is a sample of this database relationship table.

ParentUnitType	ChildUnitType
CDAU-2016M	MCI-105-1
CDAU-2016M	GPS-101A-1
CDAU-2016M	FPD-104-2
CDAU-2016M	FPD-104-1
CDAU-2016M	IRG-101B-1
CDAU-2016M	FPD-104-3
CDAU-2016M	AMD-116P-2
CDAU-2016M	EUP-105-1
CDAU-2016M	PAO-100-2
CDAU-2016M	AMD-120A-1
CDAU-2016M	AMD-116P-1
CDAU-2016M	LRV-103-2
CDAU-2016M	AMD-120A-2
CDAU-2016M	RTD-120A-2
CDAU-2016M	LRV-103-1
CDAU-2016M	GRC-104A-1

Figure 3. Connection Rules Examples

THE UNIT CONFIGURATION

Each controller unit, remote unit, or card unit can have programmable attributes associated with it. In ILIAD, these attributes are termed the Unit Configuration. Figure 4 depicts an example of a Unit Configuration and a Network with unit connections as shown by the ILIAD graphical user interface. The Unit Configuration consists of attributes that are typically dependent at the unit level, rather than at the channel level. In some cases, these attributes apply globally to all channels of the card.

The database contains a mapping of which attributes belong to which unit. Each attribute belongs to a Category such that a set of related attributes can be presented together as a set. Each attribute is assigned an attribute type, which subsequently directs the graphical user interface on how to present the attribute to the user. The attribute types include:

1) Select from itemized list, where each list item is enumerated in the database;
2) Select from a calculated list, where the entries are calculated from a starting value, range, and increment;
3) Enter a decimal value within a given range;
4) Enter a hexadecimal value within a given range; and
5) Enter a string of characters.

Once the user has made an attribute value selection or has entered the value, the programming code value used to generate the program load module is stored in the database.

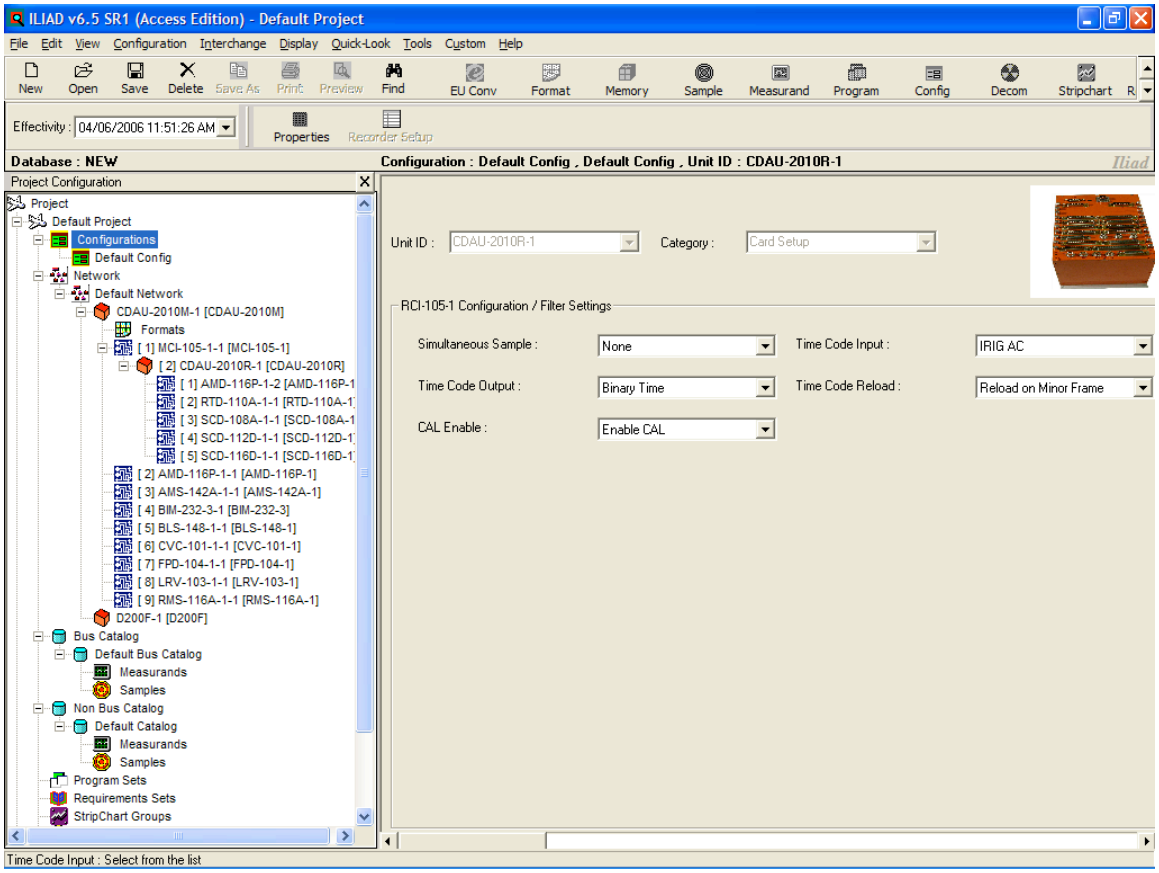

Figure 4. Instrumentation Network – Unit Connections and Configuration

Another consideration in dealing with the Unit Configuration attributes is a concept termed the ‘dependent attribute’. This denotes that the selections or values associated with an attribute are dependent on another attribute. Once the value of a particular attribute is selected, the choices for the dependent attribute change.

CHANNEL ATTRIBUTE ABSTRACTION

The next type of abstracted object is the channel attribute, also termed a ‘sample’. Samples are also used for bus words, such as in a MIL-STD-1553 bus message. The sample is considered the raw data returned when a channel is accessed for information, or when a bus word of a message is retrieved. Each sample that is created is assigned a Sample Type, which gives it a general category. This can be Analog, Digital, or Bus; or it can be a Special Sample such as Sync, Frame ID, or Fixed Word. Another category is Reserved, which is used to define samples directly attached to units instead of channels.

When it comes to the programming attributes for samples, each sample is assigned an Extended Type, which is the way the sample is categorized for assignment of attributes. Each Extended Type has its own specific set of attributes assigned to it, and each of these

attributes has an Attribute Type, which includes some of the same types as the attributes of Unit Configurations. For samples, this includes:

1) Select from itemized list, where each list item is enumerated in the database;
2) Select from a calculated list, where the entries are calculated from a starting value, range, and increment;
3) Enter a decimal value within a given range; and
4) Enter a hexadecimal value within a given range.

For example, a given box type unit such as a controller or remote unit can have several Extended Types, meaning that samples defined for the unit have attributes that differ depending on the sample type chosen. Some signal conditioning cards also have more than one sample type.

Figure 5 depicts an example of the sample attributes and value selection for a Teletronics® SCD-108A card. In this example, the sample attributes are: Data Output Format, Primary Gain, Secondary Gain, User Specified Offset, and Trim Counts.

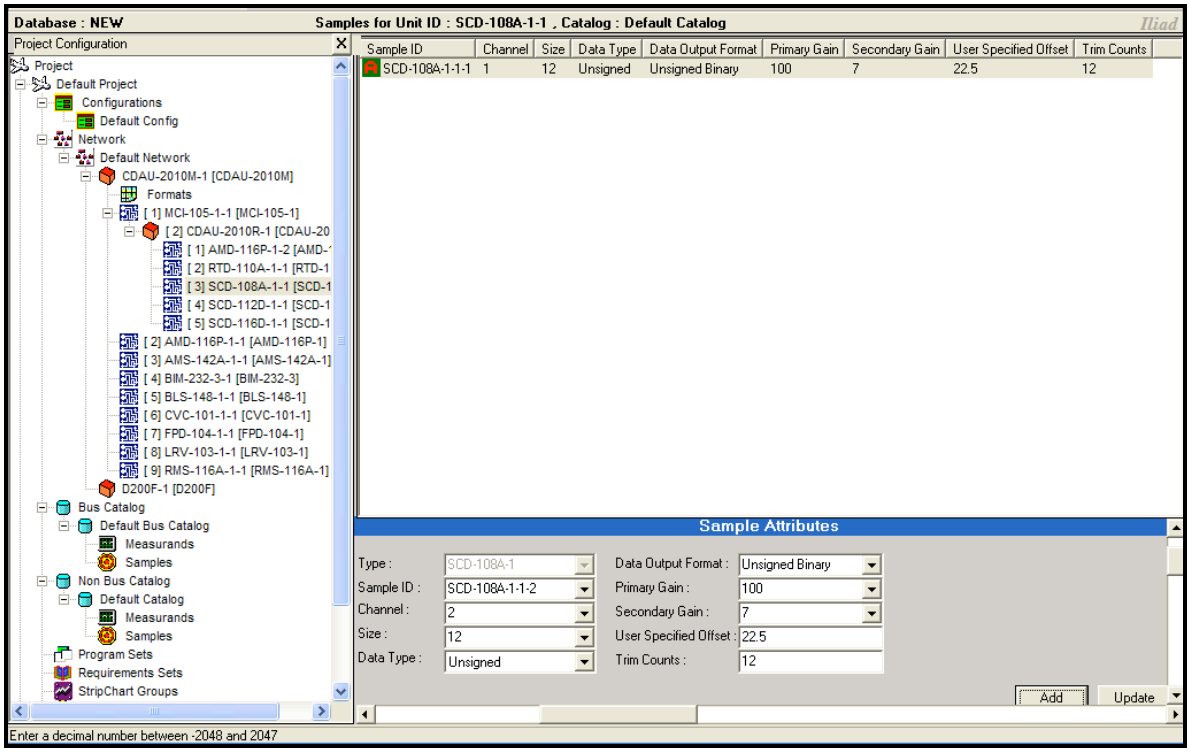

Figure 5. Sample Attribute Value Definition for a Signal Conditioning Card

PCM UNIT PROGRAMMABLE ATTRIBUTES

A PCM Format defines a stream of measurements that a controller generates from its internal program. One of the attributes of the controller is the number of PCM Formats that

its unit memory can contain. However, there are other attributes of a PCM Format that must be maintained for each unit. Some of these are generic and apply universally. For example, typical universal attributes include the number of words in a frame, number of frames in a major frame, and the common word size. Other attributes are more closely related to the unit, such as bit rate, which can have different allowable rates among different controllers. These Unit Type PCM Format attributes are tracked in the database for each unit type.

Another capability of various controllers is the ability to generate more than one PCM stream at a time, using what is called programmable tracks. This allows a subset of the PCM stream to be sent to another output on the controller. Figure 6 shows a typical set of PCM Format attributes, some of which are generic, and some of which are very specific to the controller.

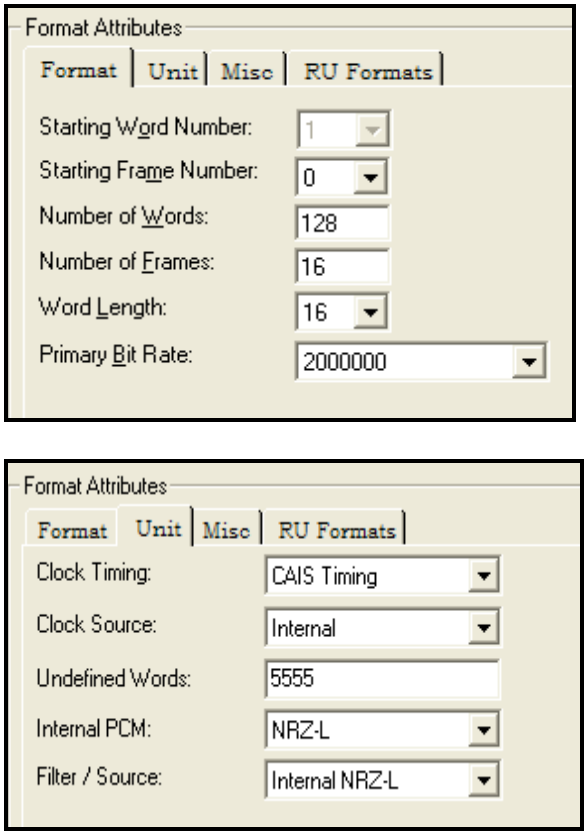

Figure 6. PCM Format Attributes for a Controller

UNIT PROGRAM GENERATION

To produce the appropriate programming for various units, the user undergoes a process of defining instances of the units of interest, connecting them in the proper configuration, defining the Unit Configuration attributes, defining the samples on the units and signal conditioning cards, setting the sample attributes, and defining the PCM Formats. This is

followed by defining the Program Set, which represents the load modules for all of the programmable units. At this point, the PCM Formats are specified for the controllers, and the load modules are generated. The program generators extract the codes from the database that represent the values the user chose or entered, and produce either a binary or hexadecimal load module depending on the class of instrumentation. The load module is then ready to be loaded into the target controller and remote units using the load type specified in the database.

SUMMARY

The authors have demonstrated a methodology for supporting instrumentation from multiple vendors by abstracting the programmable characteristics of the various devices into a common schema definition of unit types, capacities, connection rules, sample types, sample attributes, PCM configurable attributes, and loading characteristics. This allows additional units to be easily added, utilizing the same graphical user interface to define instrumentation configurations, PCM Formats, and generate instrumentation load modules. This eases the job of the end user, simplifies user training requirements, and minimizes overall operations and maintenance costs for blended unit type environments.