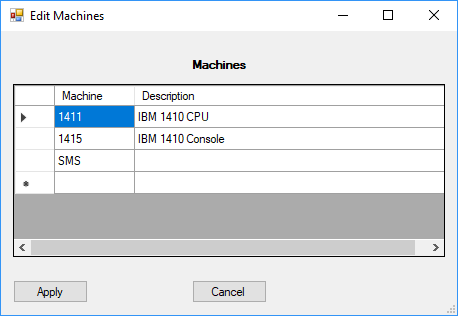
**Machines**



At the top of the heap, we have machines. A machine is not intended to be an entire computer. IBM subdivided computers (like the IBM 1410) into a series of machines (1411 – the CPU, 1415 – the console, 1414 – the I/O synchronizer, and so on). In addition, the application assumes a machine called SMS exists, to which it will attach SMS card ECOs.

The machine form, like many of the forms in this application, uses a .NET Data Grid View control. This control is not associated with the database directly. Instead, it is associated internally with a List or BindingList which absorbs any updates or deletes made in the rows displayed.

Like most of the forms in the application, any updates, including deletes, do not actually occur until you click the Apply button. (Attempts to delete rows, however, are checked for referential integrity. These checks are done in the application, not the database).

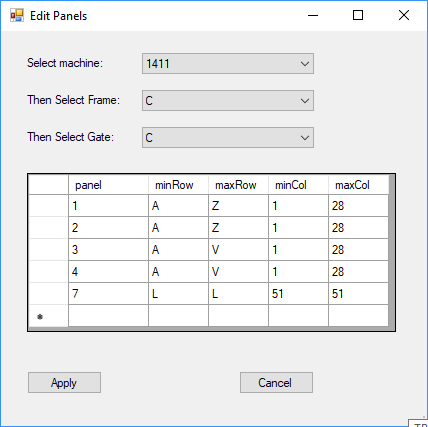
A new machine can be defined by entering the data in the new row that has “\*” in the left hand column.

NOTE: In some of the data grid views, one has to click F2 to edit a field. I have not yet looked into exactly why one sometimes has to do that.

This page is accessed via Edit => Machines

**Frames, (Machine) Gates and Panels**



In the SMS system, machines comprise one or more Frames, which comprise one or more Gates, which comprise one or more Panels, and panels are made of up card slots. Some machines (like the IBM 1410 machines) don’t actually use gates. In that case, in the application, one just creates a gate with the same name as the frame in the application.

Each panel entry defines the card slots available in that panel. In SMS, IBM used a coordinate system to define slots, where a letter was used for the row (skipping I and O), and a number for the card column.

These pages are accessed via:

* Edit => Frames, and selecting the appropriate machine
* Edit => Machine Gates, and selecting the appropriate machine and frame
* Edit => Panels, and selecting the appropriate machine, frame and gate

**Volume Sets**



Volume Sets are also at the top of the heap of the paper trail. A volume set is an associated set of diagram pages – typically associated with a given machine type and/or serial number, or, perhaps, a given source for the materials (which is the case in this example). The association in the application is completely arbitrary. Volume sets and volumes have no association with a “machine” as described above.

This screen is accessed via Edit => Volume Sets

**Volumes**



A volume is intended to represent a particular multi-ring volume binder within a given volume set. These would be associated with the classic dark blue IBM binders that could be found in a cart in the machine room or nearby, for CE’s to use when diagnosing problems with the computer. In the application, the volumes are associated with a given volume set. A volume can be associated with a given machine serial number – but this is just information – it is not used by the application. (The ID:###### displays the internal database key for the volume set, for informational purposes).

This screen is accessed via Edit => Volumes, and then selecting the appropriate Volume Set in the drop down list.

**Logic**

Logic levels and operations in the application are defined by the Logic section of the menu, which contains entries for editing logic families, logic voltage levels, IBM logic functions, and “standard” logic functions.

**Logic Families**

IBM defined several logic families for use in SMS. The IBM 1410, for which I built this application, uses only a subset of them. Logic families are used when defining SMS card types (described later) – a given card is presumed to use only of the logic families. This screen shot of the Logic Family editor, accessed by Logic => IBM Logic Families, show some of them. (Not sure why the blank logic family is in there, presumably an artifact of importing a card type that did not have a family specified). The logic families are for documentation only, and not used for anything else (in particular, they are not used for HDL generation).



**Logic Voltage Levels**

The various IBM logic families used (usually different) specified voltage levels to represent 0 or1. These can be different for an IBM SMS Automated Logic Diagram logic block input and output, and are represented most often by a single letter, or left blank in the logic block. For example, the most typical level in the IBM 1410 is “S” (logic family SDTRL), where +S is ideally 0v and –S is ideally -5 v. The values in the logic level table, accessed via Logic => Logic Voltage Levels *are specified in tenths of a volt.* As with logic families, the voltage levels are not really used in the application. Most of the levels in the screen shot below were obtained from the manual IBM Customer Engineering, Instruction Reference, 1410 System Fundamentals, S223-2589, page 77. I used levels where +S, for example, is presumed to be logic ‘1’ and –S is presumed to be logic ‘0’.



**IBM Logic Functions**

At the top of each ALD logic block, there is a logic function symbol, for example +A or –O or IP, and so on. These represent logic functions *as IBM defined them*, representing “positive logic” and “negative logic”. But, care is needed when interpreting the diagrams, because it seems that built into many (but not all) of these names *is an assumption that the output of the gate is inverted*. For example, SDTRL gates are often NAND in terms of –S == 0 and +S == 1. But they show up in the ALD diagrams as +A (or, equivalently, -O). As with families and voltage levels the application does *not* use these IBM logic functions. They are for documentation only. One thing worthy of note: if the IBM logic function has an extra “A” or “O” at the end, one will usually find that the output of the gate goes to a “DOT Function”. Whether a DOT function performs an actual AND or an OR depends on how the circuits are constructed, and may differ from the logic implied by that trailing letter. For example, SDTRL “DOT Functions” operate as “OR” gates, using the logic values of +S == 0V and –S == -12V, because SDTRL uses PNP transistors, and when they are on, they pull the voltage to 0. So turning any one transistor on in the gates feeding the DOT function will pull the voltage to 0 – a logical OR. There are also lots of special IBM logic functions: D (driver), DLY (delay), R (resistor), CAP (capacitor), IP (power inverter), L (load – usually a diode clamp) and so on. NOTE: Currently this table is NOT used to validate the logic function entered on a logic diagram – it is actually a free-form text field.



**Standard Logic Functions**

Because at some point the plan is to generate a representation of a machine from the ALD diagrams, I defined another logic function that I call the “Standard Logic Function”. These are the more familiar kinds of logic: NAND, NOR, NOT as well as a plethora of special ones (DELAY, EQUAL (for driver circuits that don’t invert), Resistor, Capacitor, Trigger, Sense (amplifier), OSC (oscillator), ONE (always logic ‘1’), Zero (always logic ‘0’), Switch, and so on. These ARE used by the application. When an ALD logic block is defined to the application, the application requires the selection of a “Gate” (a circuit card gate, not to be confused with the machine gate, described earlier) which is associated with the card type and the particular pins used for that logic block by IBM. Each Gate, in turn, has associated with it one of the Standard Logic Functions (which is visible when the gate is selected in the application). The circuit card gate definition (described later) requires selection of one of these Standard Logic Functions when it is edited. The Standard Logic Function for a particular circuit card gate is determined by an analysis of the circuit in the card, for each gate on the card.



**Pages**

SMS Volumes handled by this application are comprised of Card Location Pages and Logic Diagram Pages. A card location page is a chart which identifies which modules are in which slots of a panel(in a gate, frame and machine). A logic diagram chart is where the machine’s logic was printed out – the Automated Logic Diagram.