

IBM SMS Data Application - General

There is quite a lot going on with the SMS data gathering application, though in reality it isn't all that complicated – but there is a LOT of detail. The end goal is to be able to synthesize HDL for use in programming a Field Programmable Gate Array (FPGA) to implement the machine's logic.

The application was developed in C# under Visual Studio 2017, using MySQL. It ought not to be *too* difficult to change to a different database, though I did not generalize it in its current release. Why C#? The toolset was freely available, in wide use and I was interested in learning C#. Also, I did not expect there to be all that many people (if any) who might actually be interested in actually using the application. I considered a web application say under PHP. Given the relatively simple structure of the application, that ought to be do-able. If I am still around in another decade, who knows. ;)

The C# code is, well, unprofessional (and as someone who did right some code professionally, I get to say that.) There are lots of places where code could be factored – where I chose the expediency of copy and paste. This helps make the code easier to understand, but sometimes changes that perhaps could have been made in just one place had the code been factored end up having to be made in more than one place.

The application is available on github at <http://github.com/cube1us/IBM1410SMS>

It will probably help to access a set of drawings, particularly from the IBM 1410, at least at first, as an aid in following this documentation. Recommended are:

- The SMS circuit card diagram volume at http://bitsavers.org/pdf/ibm/1410/drawings/1410_SMS_VOL_1.pdf
- Volume II containing the feature list, most of the card location charts and some ALD logic diagrams for the IBM 1410 at http://bitsavers.org/pdf/ibm/1410/drawings/1410_SYSTEM_VOL_II.pdf

For the database structure, the root directory for the project has some things worth looking at – especially the data model for the database. Also, one can use the MySQL workbench to look at the documentation for individual data elements. There is a very simple database isolation layer, found in sub-project MySQLFramework. The actual application is in sub-project IBM1410SMS. I wrote this because I found using an actual database entity framework created much more language/system/database dependency than I was comfortable with. Although the application was written to MySQL it should not be *too* hard to change it to use a different database.

C# Data entity object(s) for a given table or tables have to be re-created anytime a change is made to the table(s) involved. That is handled by the script SQLToEntity.pl found in the Tools folder of the IBM1410SMS project. This script writes the C# code for an entity into the Entities sub-folder.

The way I have used the application to date has been:

1. I used the information from the card location charts (most, but not all, of which are in Volume II, as referenced above) to build spreadsheets to use to pre-populate the card location information in the database.
2. I used the information from the SMS card volumes (there are two, one of which is referenced above) to analyze the circuits used for the cards that were identified during step 1.
3. Having completed the first two steps, I worked in parallel on the application code to edit information and to import information from the spreadsheets. This took place during the first few months of 2018.
4. Starting in May, 2018, I started processing the actual ALD logic diagrams and entering data into the database.
5. In March, 2020, I entered the Cable/Edge Connection Diagrams
6. In April and May 2020 I have been working on various Report classes for error / consistency checking – and they helped me find and fix quite a few inaccuracies in the captured information.
7. In June 2020 I worked on several items, including development of the ability to consolidate a group of related ALD signals (e.g., representing different bits) into a `std_logic_vector`.

Database Structure

The basic structure of the database, while relational, can be thought of as a few important hierarchies (naturally, this is nowhere near complete.) A standalone Python program, `databasecheck.py`, can be used to check certain things in the database.

- Machine (e.g., the 1411 CPU for the IBM 1410) [See the following page, too]
 - Feature
 - Frame
 - Machine Gate (The IBM 1410 really doesn't have these)
 - Panel
 - IBMSMSPackaging Class (not currently in database)
 - Card Slot
- Volume Set
 - Volume
 - Page
 - Card Location Page
 - ALD Diagram Page (a Diagram Page with Diagram Blocks)
 - Cable/Edge Connector Page
- Card Location Page
 - Card Location Block
 - Card Type
- SMS Card Type
 - Logic Family (e.g., SDRTL)
 - Card Gate
 - Logic Functions (IBM+, IBM-, "Standard")
 - Logic Voltage Levels
 - Gate Pin
- ALD Diagram Page
 - Diagram Block
 - Connection
 - DOT Function
 - Tie Down
 - ECO tags
 - `logicCheckRules` (There is no editor for this table. It is used in the `ReportConnectionErrors` class to decide if a given card gate's logic function (e.g. NAND, NOR, NOT, etc.) matches the Diagram Block in which it is used (symbol, polarity of output))
 - Edge Signals – which appears as inputs and outputs to and from the page in the connections table.

- busSignals – signals which are to be consolidated into a bus during HDL Group generation. (Currently there is no editor for this table)
- Cable/Edge Connector Page
 - Cable/Edge Connector Block
 - ECO tags
 - Cable Implied Destinations (There is no editor for this last one. It is a two text-column pages with cable source/destination rules of the form MMM,F,G,P,R,CC – Machine, Frame, Gate, Panel, Row and Column. The Row field of the source is typically “*”, meaning it matches any row)

ALD Diagram Location Designations

Unfortunately, IBM was not consistent in how they designated card locations in ALD diagrams. The IBM 1401 (SMS Module Type I), the 7090 (at first, called SMS Module Type II, then later called Sliding Gate Frames), the 1620 (for which I have not found a name) and the IBM 1410 (SMS Module IV aka) are all different.

The program uses the nomenclature for the IBM 1410 – SMS Module IV internally. This can lead to some confusion while using the application (especially between rows and columns). Below is a table of a recommended way to use the application, to help translate. The labels used for these in the application dialogs can be set in the machine table using the EditMachine dialog.

Originally I had set this up where Frame and Gate are separate, based on a limited understanding of the SMS Module II Sliding Gate system, but I misunderstood. So, in the application, essentially the Gate is always the same as the Frame.

Program Nomenclature	SMS Module IV / Rack and Panel Module (e.g. IBM 1410)	SMS Module I aka Swinging Gate Cube (e.g. IBM 1401)	SMS Module II / Sliding Gate Frame	IBM 1620
Machine	1411, 1414, 1415	1401	7100, 7606, 7607, 7151, ...	1620, 1621, ...
aldMachineType (database field in table machine)	Component. 11, 14, 15 (1411, 1414, 1415)	Frame: 01, 02, ...	Frame: 01, 02, ... (Note: May not correspond in order to Machine, above as I have listed them)	Machine Type: 01, 02, ...
Frame == MachineGate	Frame	Module	Module	Gate
Panel	Gate	Gate	Panel (Chassis)	Panel
Row (alphabetic)	Row	Column !!!	Row	Row
Column (2 digits)	Column	Row !!!	Column	Column

Things to watch out for:

- You will probably want to set things up so that the application MachineGates (which are below Frame in the application hierarchy) just match the Frame.
- **Note in particular the row/column reversal for the IBM 1401. This is because the application currently insists that column be numeric.**
- Internally, the application uses the machine in most places, except that it uses the “aldMachineType” column in the machine table for the first two characters of card location when drawing logic blocks or buttons that display a card location. (There may well be other uses for this field down the line, for example in decoding Edge Connections that appear at the bottom of ALD Diagram Pages)

Application Dialogs

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

This dialog is also where you set YOUR name to use for a given machine type for what the application internally calls Frames, Gates, Panels, Rows and Columns, and also the two character machine type prefix used in diagram boxes and buttons.

 Edit Machines

Machines

	Machine	Description	Type	Frame	Gate	Panel	Row	Column
▶	1401	IBM 1401 (Sample for showing how labels change)	01	Module	Module	Gate	Column	Row
	1411	IBM 1410 CPU	11	Frame	Gate	Panel	Row	Column
	1414	IBM 1414 I-O Synchronizer (General)	14	Frame	Gate	Panel	Row	Column
	1415	IBM 1410 Console	15	Frame	Gate	Panel	Row	Column
	SMS	SMS Card Machine to associate with SMS ECOs						
	TE99	Machine to use when testing code changes	99	Frame	Gate	Panel	Row	Column
•								

Apply

Cancel

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.

There is also a class, IBMSMSPackaging.cs which contains machine-dependent information on panels, which identifies:

- Panels which are "special" when it comes to cross-checking Sheet edge connectors (the lists at the bottom of diagram pages)
- Which panels are adjacent to other panels
- Which rows and columns are valid for a panel, and which of those rows are interconnect rows used to make connections to other panels that do not show up on cable/edge connector pages.

The frame, gate and panel 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

Dialog box titled "EditFramesForm". It contains a "Select machine:" dropdown menu with "1411" selected. Below the dropdown is a table with a "Name" column and four rows containing "B", "C", "D", and "F". A small asterisk is in the bottom-left corner of the table area. At the bottom are "Apply" and "Cancel" buttons.

Name
B
C
D
F

Dialog box titled "EditMachineGatesForm". It contains a "Select machine:" dropdown menu with "1411" selected. Below this is a "Then Select Frame:" dropdown menu with "B" selected. Below the dropdowns is a table with a "Name" column and four rows containing "B", "C", "D", and "F". A small asterisk is in the bottom-left corner of the table area. At the bottom are "Apply" and "Cancel" buttons.

Name
B
C
D
F

Features

An IBM Machine might have installed, whether at the factory or via an RPQ (Request for Price Quotation) a number of different features. For example, the IBM 1410 might or might not have the accelerator feature, or be configured with various amounts of core storage.

The Features dialog allows editing the features for a particular machine type. These are then used by card location chart pages to identify cases where a particular card location is present or used if a machine has a given feature, for use during troubleshooting or potentially while installing a particular feature.

This page is accessed via the menu entry Edit => Features and then selecting the desired machine type.

*Note: Currently the feature table for the IBM 1410 has some duplicate entries, of a sort, most of them starting with '\$' or 'S'. One set was the result of features on a feature page, where some features, e.g. \$40 used a '\$' instead of an 'S'. Other features were added automagically during import of card location chart entries, which use 'S'. These may eventually need to be reconciled. In the **meantime do NOT use the '\$' entries** (those that start with a dollar sign) when editing the card location chart or gates.*

Use THESE entries, that start with 'S':

Dialog box titled "EditFeaturesForm".

Select machine: 1411

Code	Feature
P1	THOSE ADDITIONS TO BASIC MACHINE REQUIRED FOR THE PAPER TAPE PUNCH ATTACHMENT TO CHANNEL
P2	THOSE ADDITIONS TO L2 REQUIRED FOR THE PAPER TAPE PUNCH ATTACHMENT TO CHANNEL TWO
Y1	THOSE ADDITIONS TO THE BASIC MACHINE REQUIRED FOR THE ATTACHMENT OF THE PRIORITY FEATURE T
S2	Added Automatically by Card Location Chart Import
S10	Added Automatically by Card Location Chart Import
NOTE	Added Automatically by Card Location Chart Import
F11	Added Automatically by Card Location Chart Import
S40	Added Automatically by Card Location Chart Import
S20	Added Automatically by Card Location Chart Import
S60	Added Automatically by Card Location Chart Import
S4	Added Automatically by Card Location Chart Import
S6	Added Automatically by Card Location Chart Import
AUTS	Automatic Start
O0	Portable CE Start/Stop Box (Not listed in feature list, but appears in ALD)
C0	Referred to in CH 2 Not Ready/Busy, page 13.66.05.1 Should maybe be L2?
*	

Buttons: Apply, Cancel

Do NOT use these entries, that begin with '\$'

EditFeaturesForm

Select machine: 1411

Code	Feature
**	HARDWARE REQUIRED IN THE BASIC SYSTEM
\$10	THOSE ADDITIONS TO THE BASIC SYSTEM REQUIRED TO IMPLEMENT 10K STORAGE ONLY
\$2	THOSE ADDITIONS TO THE BASIC SYSTEM REQUIRED TO IMPLEMENT 20K STORAGE OR GREATER
\$20	THOSE ADDITIONS TO THE BASIC SYSTEM REQUIRED TO IMPLEMENT 20K STORAGE ONLY
\$4	THOSE ADDITIONS TO THE BASIC SYSTEM REQUIRED TO IMPLEMENT 40K STORAGE OR GREATER
\$40	THOSE ADDITIONS TO THE BASIC SYSTEM REQUIRED TO IMPLEMENT 40K STORAGE ONLY
\$6	THOSE ADDITIONS TO THE BASIC SYSTEM REQUIRED TO IMPLEMENT 60K STORAGE OR GREATER
\$60	THOSE ADDITIONS TO THE BASIC SYSTEM REQUIRED TO IMPLEMENT 60K STORAGE ONLY
\$8	THOSE ADDITIONS TO THE BASIC SYSTEM REQUIRED TO IMPLEMENT 80K STORAGE OR GREATER
\$80	THOSE ADDITIONS TO THE BASIC SYSTEM REQUIRED TO IMPLEMENT 80K STORAGE ONLY
Z0	THOSE ADDITIONS TO THE BASIC MACHINE REQUIRED TO IMPLEMENT COMPATIBILITY WITH THE 1401 MACH
Z1	THOSE ADDITIONS TO Z0 REQUIRED TO IMPLEMENT COMPATIBILITY WITH THE 1401 WITH 16K STORAGE.
L1	THOSE ADDITIONS REQUIRED TO PERFORM OVERLAP ON THE FIRST CHANNEL
L10	THOSE ADDITIONS REQUIRED TO PERFORM OVERLAP WITH 10K STORAGE ONLY
L2	THOSE ADDITIONS REQUIRED TO ADD A TAPE AND/OR FILE SYSTEM TO THE SECOND CHANNEL OF THE CPL
B1	THE BASIC I/O BUFFER, INCLUDING THOSE ADDITIONS TO THE BASIC MACHINE (. .) REQUIRED TO IMPLEMEN

Apply Cancel

ECOs

Many artifacts can reference Engineering Change Orders, or ECOs. These can either relate to a particular machine type (e.g. 1411, for the IBM 1410 CPU) or to the application-defined machine type “SMS”, used to capture ECO levels of SMS card drawing pages.

ECOs are accessed from the menu via Edit => ECOs, and then selecting the desired machine type or “SMS”.

When an ECO is entered while editing an Automated Logic Diagram (ALD) page, it is automatically entered into the ECO table. The description can be changed later, if desired.

Currently the application does not really use the ECO, though in theory one could have multiple ALD pages with the same page number, but different ECO levels if the application allowed for that. The database does make allowance for this eventual possibility by containing the ECO in the cardLocationPage and the diagramECOtag tables, but more work would be necessary, including identifying the “defining” (probably latest) ECO in the diagramPage table, which it does not currently contain. Then one could associate particular pages with particular ECOs into a given volume – and then use those throughout the application.

Alternatively, for now, one would need to “clone” the database and then alter it for a different machine. Definitely a shortcoming in the application.

Note: The database for machine 1411 currently contains an ECO with a blank ECO number that I should probably investigated (see below)

The screenshot shows a window titled "EditECOsForm" with a "Select machine:" dropdown menu set to "1411". Below the menu is a table with two columns: "ECO" and "description". The table contains 17 rows of data. The first row has a red exclamation mark icon in the "ECO" column. The "ECO" column contains values: 0250786, 0253453, 109306, 109333, 109353, 109370C, 109370D, 109370E, 109370F, 109370G, 109370H, 109370J, 109370L, 109372, and 109386. The "description" column contains the text "Added by ALD Diagram Page Form" for all rows except the first two. At the bottom of the window are "Apply" and "Cancel" buttons.

ECO	description
0250786	
0253453	
109306	Added by ALD Diagram Page Form
109333	Added by ALD Diagram Page Form
109353	Added by ALD Diagram Page Form
109370C	Added by ALD Diagram Page Form
109370D	Added by ALD Diagram Page Form
109370E	Added by ALD Diagram Page Form
109370F	Added by ALD Diagram Page Form
109370G	Added by ALD Diagram Page Form
109370H	Added by ALD Diagram Page Form
109370J	Added by ALD Diagram Page Form
109370L	Added by ALD Diagram Page Form
109372	Added by ALD Diagram Page Form
109386	Added by ALD Diagram Page Form

Volume Sets



Volume Sets are also at the top of the heap of the paper trail. A volume set is an associated set of volumes (which in turn contain 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

The screenshot shows a window titled "Edit Volumes" with a standard Windows interface (minimize, maximize, close buttons). At the top, there is a label "Select Volume Set:" followed by a dropdown menu displaying "Machine Type: 1411 S/N: Ohio - 2010 ID: 100006". Below this is a table with the following columns: "Vol. Name", "Order", "Machine S/N", "First Page", and "Last Page". The table contains 14 rows of data, including "SMS", "Unassigned Sheet", and volumes I through XI, followed by "Unassigned" and "TIEDOWN". At the bottom of the window are two buttons: "Apply" and "Cancel".

Vol. Name	Order	Machine S/N	First Page	Last Page
SMS	0			
Unassigned Sheet	0			
I	1	SN03950001	98.11.02.0	98.16.02.0
II	2	SN03950001	11.04.00.0	12.12.67.1
III	3	SN03950001	12.13.01.1	13.14.14.1
IV	4	SN03950001	13.42.10.1	14.13.05.1
V	5	SN03950001	14.14.01.1	14.42.05.1
VI	6	SN03950001	14.45.01.1	15.41.12.1
VII	7	SN03950001	15.49.01.1	15.70.33.0
VIII	8	SN03950001	16.11.01.1	17.12.05.1
IX	9	SN03950001	17.13.01.1	36.10.04.1
X	10	SN03950001	36.11.0.1	45.50.16.1
XI	11	SN03950001	25.04.03.0	25.97.01.0
Unassigned	12	SN03950001		
TIEDOWN	13	SN03950001		

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).

Volumes can also be identified for SMS card type entries, as these are derived from pages in their own separate IBM "blue binders".

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 one 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, possibly 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 or 1. 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’.

This table can also influence how DOT functions are generated. For example, SDTRL “DOT Functions” operate as “OR” gates, using the logic values of +S == 0V and –S == -12V, because SDTRL uses PNP transistors with a collector pull up to a negative voltage (-S), and when they are on, they pull the voltage to 0 (+S). So turning any one transistor on in the gates feeding the DOT function will pull the voltage to 0 – a logical OR. So by default the logic generation handles DOT functions as a logical OR. Other logic families’ DOT Functions are different at times – the IBM 1410 has some cases (logic level “B”) where +6 is logic one, and in such cases the DOT function acts as a logic AND. The default of “OR” can be overridden two ways. First of all, there is now a “dotFunctionLogic” column in the LogicLevels table (not this table) to specify the logic function to use. It is not required, but available so long as all of the input levels to and from other logic blocks and/or edge signals has a consistent logic level. Secondly, there is also a special field in the DOT function dialog for forcing the *electrical* logic function when necessary.

The next page shows is a screenshot of the Logic Levels dialog, accessed by Logic => Logic Voltage Levels in the menu.

 Edit Logic Levels

—

□

×

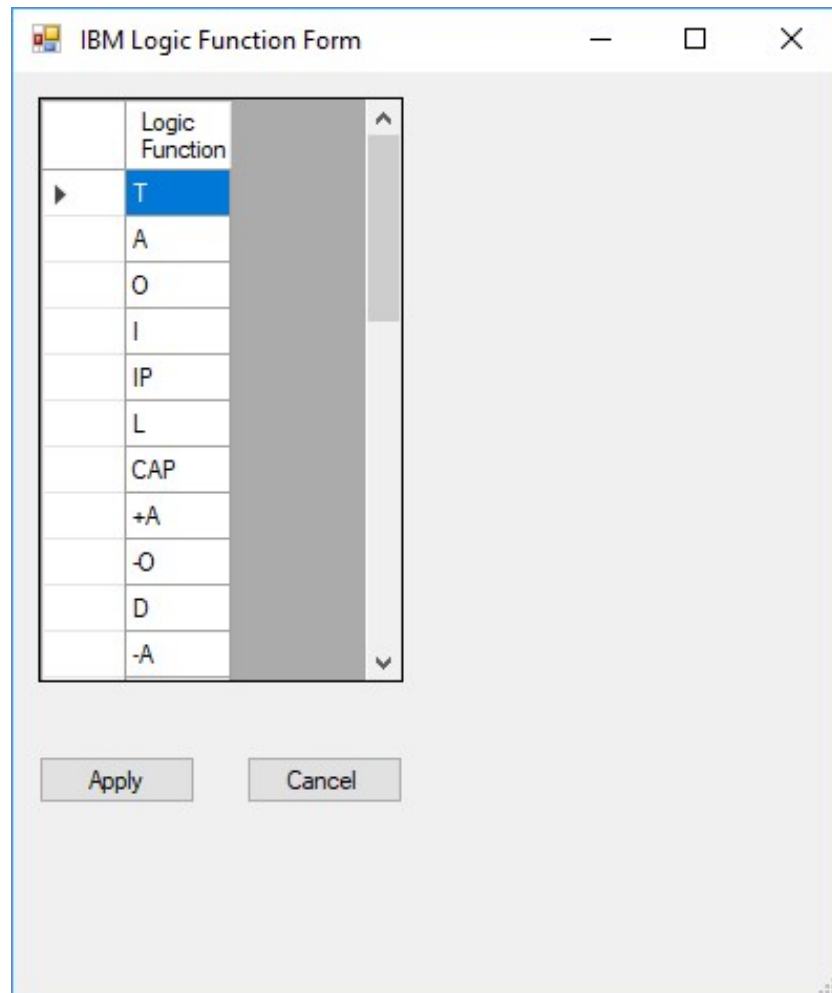
	Logic Level Name	Zero (Tenths/V)	One (Tenths/V)	Circuit Type	Logic for DOT Functions
▶	A	-120	50	STDTLx	
	B	-50	0	DDTL	AND
	S	-125	0	SDTRL	
	Y	-60	0	SDTDL	
	V	0	50	Testing?	
	C	135	0	Current in ma	
	PL	-360	0	CONS -36 ...	
	2	-360	0	CONS -36 ...	
	6	-360	0	-V 36 Volts	
	DK	-360	0	-V 36 Volts	
	1	-360	0	-V 36 Volts	
	3	-360	0	-V 36 Volts	
	4	-360	0	-V 36 Volts	

Apply

Cancel

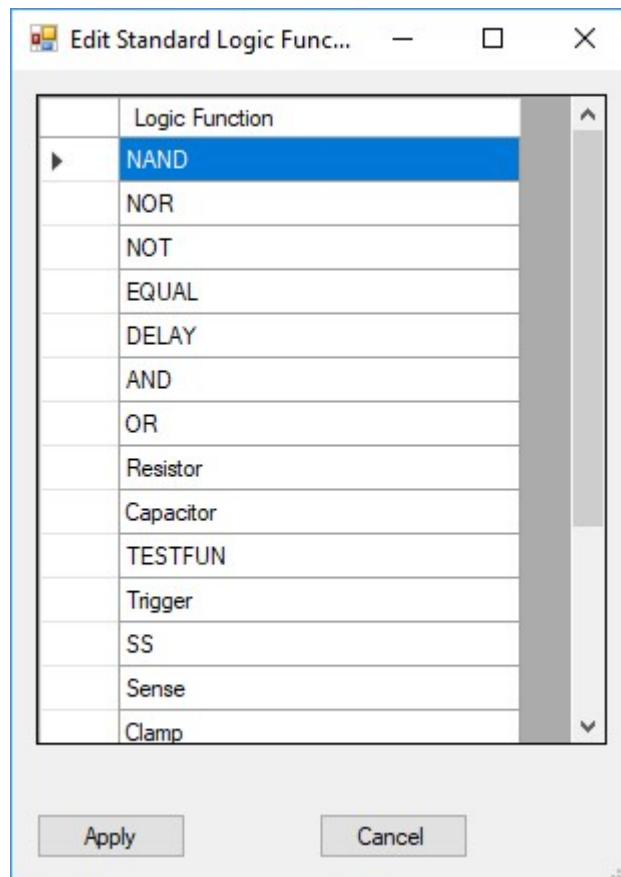
IBM Logic Functions

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 in the logic block symbol.** See the discussion in the logic Levels table, above. 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. Access this dialog via Logic => IBM Logic Functions.



“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 (a flip flop), Sense (amplifier), OSC (oscillator) SS (single shot), ONE (always logic ‘1’), ZERO (always logic ‘0’), Switch (which has special handling code in HDL generation), “special” (which would require specific code in the application to generate HDL) 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. This page is accessed via the menu Logic => Standard Logic Functions.



SMS Card Types

The information on SMS cards used by the IBM 1410 are handled specially in the application, rather than as pages, but are, in fact, derived from pages in the typical IBM blue volumes. There are three types of entities involving SMS cards: SMS Card Types, Therefore, an SMS card type is assigned a Volume First among these are the SMS card types, SMS card gates and card pins.

SMS card types are edited via the menu SMS Cards => Edit SMS Card Types, selecting the particular SMS volume containing the card type of interest and then selecting the card type itself.

This page also allows one to create a new SMS card type in a given volume by clicking on “New Card Type” or correcting/assigning a card type to a different volume using the “Changed Diagram Volume” pull down.

Each card type has a 3 or 4 character type (you can decide whether or not to include the trailing “-” in a card type, though I did not do so for the IBM 1410.) and an IBM part number.

The SMS Card Type entity identifies the volume in which the page describing the particular SMS card type appears, along with the logic family for the card, and various meta-data that appear on the SMS card type page. The Card Type itself is central to the application. The part number is used in pull-down lists and the like and the height is used during card slot error checking. The rest of the meta-data are not used by the application.

Edit SMS Card Type

Diagram Volume: Vol. Set 1410 SMS, Vol. II

Card (Existing): DMQ, Part No. 0372125 Part# 0372125 Type: DMQ New Card Type

Logic Family: DDTL Name: DDTL 1-6 WAY NL-1-2 WAY NL 2 LDS 390 OHM

NOTES

Note Name	Note
*	

CHANGED Diagram Volume: Vol. Set 1410 SMS, Vol. II

Card Height
☒ Single ☐ Double

APPROVAL: Date: 1/ 1/1960 Hole Pattern:

Type (Name): CARD ASM TSTR Dev. No.

Design: Date: 1/ 1/1960 Type: SMS Device:

Detail: Date: 1/ 1/1960 Scale: NONE

Check: Date: 1/ 1/1960 Draw: 1/ 1/1960

Appro: Date: 1/ 1/1960 Check: 1/ 1/1960

Apply Delete Cancel

ECOs

ECO	Date	Note Name	Approver
▶ 114262			
*			

SMS Card Gates

SMS card type gates (not to be confused with machine gates) is where “the rubber meets the road” in this application.

A given SMS card type may have one or more logic gates / flip flops / etc. These are determined by essentially reverse engineering the information on the SMS card page.

These are accessed in the application from the Menu via SMS Cards => Edit SMS Card Gates, selecting the appropriate SMS volume and then the desired Card Type.

The first column is an arbitrary gate number, usually assigned left to right, top to bottom.

The next column is “Pin”. In the application, each SMS card gate is said to have a “defining pin”. While this is somewhat arbitrary, it must be unique among the defining pins used in a given SMS card gate, and appears in the “Pin” column in the Edit Card Gates dialog. (Also, it turns out this is never actually USED in the current application, and the associated database column is not really correct. See issue #49)

The third column is “T#”, which stand for Transistor Number. It is not required by the application, but can be entered to avoid confusion in gate identification. Just pick one of the transistor numbers listed on the SMS card circuit diagram.

The next column identifies the “positive” logic function, as used on IBM ALD diagram pages. See “IBM Logic Functions” above. This is not actually used by the application, at least at present.

The next column is the “negative” logic function, as might appear on IBM ALD diagram pages. If the positive logic function is “+A”, then this will be “-O”, and if the positive logic function is “+O” then this will be “+A”. In other cases, like “I” (for inverter) or “DE” or “DLY” (for delay) it will be the same as the “positive” logic function. It is not actually used in the application.

The next column, “Logic Function: *is of critical importance to the application*”. This column is intended to be used during the process of HDL generation. It is the actual *physical* logic function performed by the gate, determined by reverse-engineering the SMS diagram page. See “Standard Logic Functions” above.

The Latch column is used to define cases where one gate is used in combination with another gate (via a cross connection) to form a latch. For gates that are not part of a Latch this will be a 0.


The next column defines whether or not the gate’s output(s) is open collector. Only at most ONE gate that is NOT open collector can participate in a “DOT function” on an ALD diagram page.

The next two columns define the gate’s *default* input and output logic levels (see Logic Voltage Levels, above). These can be overridden, however, on a given ALD logic diagram page.

The “No Out Exempt” column should be checked if this gate always / usually / often appears in an ALD with no output. It suppresses warnings of that fact when the ReportConnectionErrors report is run.

Next is an optional “HDL Name”. It is used to identify a particular block of code in the HDL generation process for gates that cannot be readily defined by just the “Logic Fun.” column. For example, on gate type AEK this column contains “SMS_AEK” for the HDL Name. (AEK is a power inverter, but with some special logic which is not readily defined as a standard logic function.)

Finally there is the Comp. (component) value column. Some “gates” are really just single electronic components like load resistors or capacitors. This column identifies the value of the component (ohms, microfarads, etc.).


Edit Card Gates

—

□

×

Volume:

Vol. Set 1410 SMS, Vol. I

Card Type:

AEK, Part No.0370215

Gates

	Gate #	Pin	T#	IBM+	IBM-	Logic Fun.	Latch	Open Coll.	In Lvl	Out Lvl	No Out Exempt	HDL Name	Comp. Value
	1	D	0	IP	IP	Special	0	<input type="checkbox"/>	S	S	<input type="checkbox"/>	SMS_AEK	
	2	F.	0	IP	IP	Special	0	<input type="checkbox"/>	S	S	<input type="checkbox"/>	SMS_AEK	
	3	P.	0	IP	IP	Special	0	<input type="checkbox"/>	S	S	<input type="checkbox"/>	SMS_AEK	
*								<input type="checkbox"/>			<input type="checkbox"/>		

Apply

Cancel

ach card gate connects to one or more pins. These pins are accessed from the menu via SMS Cards => Edit SMS Card Pins and then selecting the SMS Card Volume, Card Type and Gate (by number).

The first column is the Pin. This will usually be a letter, corresponding to an actual pinout on the card edge of the SMS card. However, there are also internal gate-to-gate connections that do not go to the card edge. The "Pin" column for these are generally left blank.

The next two columns identify gate to gate connections. If the "Pin" column is blank, then at least one of these should identify a gate from which the signal is derived (Input Gate) or to which the signal is sent (Output Gate). Note that it is also possible to have both a card edge pin identification and a gate-to-gate connection – but generally that should not be done, favoring instead using a common Pin in such cases.

The next two columns identify whether a given pin is used as an Input, an Output or both.

The next two columns identify whether a given pin is "DOT connected" (DOT-ed And/Or) with another gate. Dot Out indicates an output pin is connected to another gate's output on the same card. Dot In indicates that an given input pin is connected to multiple gates' outputs on the same card. The "Logic Function" field is read only – a reminder of what the "standard" logic function is set to for this gate.

The "Map To" column is special. For pins that are part of a gate with logic function "Special" (and perhaps others down the road), this specifies the corresponding HDL entity port/parameter name for this pin to be used when generating HDL.

The Voltage (Tenths) column indicates voltage sources. (0 does NOT mean ground. Probably something worth making more clear)

Edit Card Gate Pins

Volume: Vol. Set 1410 SMS, Vol. I

Card Type: DGR, Part No.0370258

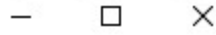
Gate: 1 Logic Function: NOT

Pins									
	Pin	Input Gate	Output Gate	In	Out	DOT In	DOT Out	Map To	Voltage (Tenths)
	A.	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		0
	F.	0	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		0
*				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		

Apply Cancel



Edit Card Gate Pins



Volume: Vol. Set 1410 SMS, Vol. I

Card Type: AEK, Part No.0370215

Gate: 1 Logic Function: Special

Pins

	Pin	Input Gate	Output Gate	In	Out	DOT In	DOT Out	Map To	Voltage (Tenths)
	C.	0	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	OUT1	0
	D	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	IN1	0
	E.	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	IN2	0
*				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		

Apply

Cancel

Pages

Volumes handled by this application are comprised of Tie Down Pages, Card Location Pages, Logic Diagram Pages and Cable/Edge Connection 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. A Cable/Edge connection page describes inter-panel, inter-gate and inter-frame connections.

Note: Pages from the volumes describing SMS card circuit diagrams are not currently handled as pages. Instead they are handled via the SMS Cards menu, as described earlier.

Card Location Pages

A Card Location page is a chart / matrix of which cards are located in which machine card slots, and often also identifies the ALD Diagram sheet(s) which use the gate(s) on the card.

The top level dialog edits the page characteristics, and is accessed by Edit => Card Location Pages, and specifying the Machine, Volume Set, Volume and Page to be edited (or clicking “New Page” to create a new Card Location Page.

The screenshot shows a Windows-style dialog box titled "Edit Card Location Page Form". It contains several input fields and buttons for editing page characteristics. The fields are organized as follows:

- Select machine:** A dropdown menu with "1411" selected.
- Volume Set:** A dropdown menu with "1411" selected.
- Select Volume:** A dropdown menu with "II" selected.
- Select Page (existing):** A dropdown menu with "11.04.02.0" selected.
- OR:** A button labeled "New Page".
- Panel:** A dropdown menu with "Frame: C Gate: C Panel: 1" selected.
- Sheets:** A dropdown menu with "1" selected.
- Page Name:** A text field containing "11.04.02.0".
- Part No.:** A text field containing "5327094".
- Title:** A text field containing "CIRCUIT CARD LOCATION CHART: FRAME 11 GATE C PANEL".
- Stamp (opt):** An empty text field.
- ECO:** A dropdown menu with "253320" selected.
- Previous ECO (opt):** A dropdown menu with "252961" selected.
- Run (opt):** A text field containing "8849".

At the bottom of the dialog, there are four buttons: "Apply", "Remove", "Force Purge", and "Cancel".


Card Location Chart

The detail for card locations is accessed via Edit => Edit Card Location on the menu, and specifying the Machine, Page, Panel, Card Row and Card Column. Currently, to do this effectively requires that you find the card location chart of interest in the scanned diagrams or the spreadsheets derived from the card location charts, because of the correlation between Page and Panel. (For example, one would have to know that page 11.04.02.0 is the page containing the card location chart for IBM 1411 Frame C, (Gate C), Panel 1. (I have a TODO to allow one to select the page, and have it pre-populate the Panel, and vice versa.

The Card Type, Feature, Crossed out (with a great big "X" on the card location chart) and numbers that appear at the bottom of the block (Bottom Notes) are captured from card location chart.

In the data table, the Page column contains pages that have ALD Diagram logic blocks that refer to this card in this card slot. Sheet Column and Sheet Row identify the coordinates of the logic block that refers to this card. The ECO, if present, identifies the ECO level on the card location chart block. The On Sheet column means that during the spreadsheet capture, the card and card slot were verified to be consistent with the card location chart. Ignore means that this logic block was marked to be ignored for one reason or another. Missing means that while the card location chart indicates the given sheet location uses this card, it was found to be missing on the actual ALD diagram sheet.

Ignore is more complicated. This indicates that this entry should NOT be used when pre-populating information when capturing a logic block on an ALD diagram - for example, if it was found to not actually be present on the ALD diagram set. Ordinarily this is because a discrepancy between an ALD diagram page and the card location chart was noticed during data entry from the Card Location Chart.

 Edit Card Location

Drawing Page

Machine: 1411
Volume: Vol. Set 1411, Vol. II
Page: 11.04.02.0
Page ECO: 253320

Card Slot

Panel: Frame: C, Gate: C, Panel: 1
Card Row: E
Card Col: 8

Card Type: DGR
Feature: S10

Slot Crossed Out
☒

Bottom Note(s): 2, 3, 4, 6, 8 (1-2 chars. ea., Separate with Commas)

Sheet References

	Page	Sheet Col.	Sheet Row	ECO	On Sheet	Ignore	Missing	Note
	14.71.62.1	2	H		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
	14.71.62.1	4	E		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
	14.71.62.1	5	H		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
*					<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Apply

Delete

Cancel

Edit Card Location Chart Dialog

Tie Down Page

Tie Downs are identified on (one or more) pages. These are typically used for cases where a signal should be tied to ground or a specific logic level to provide signals where the card(s) for a particular feature are not present because the feature is not installed in the machine.

The “W/O” column identifies the feature where, if it is NOT present, requires the use of a particular tie down. The “With” column identifies the feature where, if it IS present, requires the use of a particular tie down. The “Check” column simply captured the check marks that were or were not present for a given entry (likely identifying the cases where a particular tie down was used on a particular machine.)

By default, a tie down is to ground, unless “Other Pin” is identified.

The actual page number of the tie down list was captured, but is not displayed and is not selectable as it should be. (And the pull down should be pre-populated with appropriate value(s) once a Volume Set is selected.)

Machine: 1411

Volume Set: Machine 1411, S/N Ohio - 2010

	Pin	Page	Card Type	Frame	Gate	Panel	Row	Col	W/O	With	Check	Other Pin	Note
▶	D	11.10.20.1	DEY	C	C	2	J	26	L1		<input type="checkbox"/>		
	D	11.10.25.1	DEY	C	C	2	K	14	L1		<input type="checkbox"/>		
	D	12.12.60.1	DAC	D	D	3	J	15	F3		<input checked="" type="checkbox"/>		
	K	12.12.67.1	DGR	D	D	4	C	11	F8		<input checked="" type="checkbox"/>		
	P	12.12.67.1	DGR	D	D	3	A	28	L2		<input type="checkbox"/>		
	H	12.12.67.1	DFS	D	D	3	H	28	L2		<input type="checkbox"/>		
	H	12.60.15.1	DFP	D	D	1	C	23	L2		<input type="checkbox"/>	F	CONNECT C23H TO C23F (NOT GROUND)
	K	12.62.02.1	TAU	D	D	3	F	5	M1		<input type="checkbox"/>		
	D	12.62.04.1	DAC	D	D	3	H	12	B1		<input type="checkbox"/>		
	E	12.62.04.1	DAW	D	D	3	G	26	F1		<input checked="" type="checkbox"/>		
	E	13.12.18.1	DFS	D	D	1	H	11	Y1		<input type="checkbox"/>		
	L	13.13.07.1	DFK	D	D	1	J	13	Y1		<input type="checkbox"/>		
	G	13.50.07.1	DFS	D	D	3	G	20	M1		<input type="checkbox"/>		
	D	13.60.03.1	DFS	D	D	3	G	18	B1		<input type="checkbox"/>		
	G	13.60.04.1	TAU	D	D	3	D	12	B1		<input type="checkbox"/>		

Apply Cancel

Edit Tie Downs Dialog

ALD Diagrams

The pages discussed heretofore are the support structure for the main event: ALD diagram pages. ALD diagram pages are the pages that are the “meat” of the application.

The top of the ALD diagram hierarchy is the ALD Diagram page, which is accessed via Edit => ALD Diagram Pages, and selecting the machine, volume set, volume and page. Selecting the machine, volume set and volume pre-populates the “Select Page” pull down. One can also add a new page to the volume using the “New Page” button.

This first screen contains meta-data for the page, such as its name (number), the IBM part number for the page itself and the title at the top of the page. The “No HDL Generation” checkbox tells the HDL generation routines to bypass this entire page during generation of HDL.

If the page has been stamped with a particular note (e.g. Field Use Only) that is also noted, along with any particular comments on the page (either generated on the ALD itself, or hand-written notes.).

If the page identifies one or more ECOs in its history, those are listed in the Diagram Page ECO’s table. The table can be edited.

Most sheets have **logical** connections from other sheets as inputs or to other sheets as outputs. The Sheet Edge Information table contains that information. The row is the **approximate** row on the ALD diagram sheet where the signal appears. They are NOT exact. Each signal name on a given sheet must be unique in the inputs and/or the outputs. The Count column is generated by this application, and identifies the number of other sheets related to a particular sheet edge signal. Inputs will ordinarily show a count of 1, of course. Except for the count column, this table can also be edited in place.

There are also two **special signal names**: “**LOGIC ONE**” and “**LOGIC ZERO**”. (Note the embedded space.) These are used (especially logic 0) when a connection to a gate is made to pin J (for logic one or +S) or M (for logic zero or -S). Most of the gates just ignore unconnected pins. A few that cannot (like triggers) will assume that an unconnected pin is logic 1, but if a logic zero is needed for a trigger input on an ALD, then the special “LOGIC ZERO” signal *must* be used. I entered them as normal input signals, originating from page “00.00.00.0” – but any other page you might like to use would be fine. These signals are also discarded when generating page group HDL.

When a **physical** connection is made from a card in one gate or panel to another card in a different gate and/or panel, that is identified at the very bottom of the ALD diagram as a pair of connections from a given machine/gate/panel/card slot/pin to another machine/gate/panel/card slot/pin. Those are edited by clicking on the “Edit Edge Connections” button. There is now a report that cross checks these against the other entries on this and other pages, along with the Cable/Edge Connection pages.

Unless the “Disable card slot checks” checkbox is checked, entries are validated against the panel information that is in class IBMSMSPackaging. The Pin part of the connection is always checked. There

is a report, the Edge Connection Check report for checking the consistency of these entries among all of the pages for a machine, and also against the connections identified on cable/edge connection pages.

Finally, and probably most importantly/interestingly are the logic block / interconnections shown in the main part of the ALD page, accessed by clicking on the “Edit ALD Blocks” button.

Edit ALD Diagram

Select machine: 1411 Volume Set: 1411

Select Volume: II

Select Page (existing): 11.10.17.1 OR: New Page

Page Name: 11.10.17.1 Part No.: 5327113 ☐ No HDL Generation

Title: LOGIC GATE RING H-ACC

Stamp (opt):

Edit Edge Connections **Edit ALD Blocks**

Comment:

Diagram Page ECO's

	Tag	E.C. No.	Date
	A	251773	06/13/62
*			

Sheet Edge Information

	Left (In)	Right (Out)	Row	Signal Name	Count
	<input checked="" type="checkbox"/>	<input type="checkbox"/>	B	+S LOGIC RING ON ADVANCE 1	1
	<input checked="" type="checkbox"/>	<input type="checkbox"/>	B	+S LOGIC GATE G	1
	<input checked="" type="checkbox"/>	<input type="checkbox"/>	C	+S LOGIC RING OFF ADVANCE 1	1
	<input checked="" type="checkbox"/>	<input type="checkbox"/>	C	-S LOGIC GATE G	1
	<input checked="" type="checkbox"/>	<input type="checkbox"/>	C	-S PROGRAM RESET 3	1
	<input checked="" type="checkbox"/>	<input type="checkbox"/>	E	+S LAST LOGIC GATE 1	1
	<input type="checkbox"/>	<input checked="" type="checkbox"/>	B	-S LOGIC GATE H	2
	<input type="checkbox"/>	<input checked="" type="checkbox"/>	C	+S LOGIC GATE H	10

Apply **Remove** **Cancel**

Top Level Edit ALD Diagram Page

Machine: 1411 Volume: 1411/Ohio - 2010 Volume: II Page: 11.10.10.1

☐ Disable card slot checks

	Ref.	Connection	Connection	Connection	Connection	Connection	Connection	Connection	Connection
▶	1	11C2H28E	11D1H01E	11D1B27D	11D2B02D				
	2	11C2J01E	11C1J28E	11C1C01K	11B2C28K				
	3	11C2J01E	11C1J28E						
	4	11C2C02L	11C1C27L	11C1C01N	11B2C28N				
	5	11C2C02L	11C1C27L						
	6	11C2G28A	11D1G01A	11D1Z04D	11D3Y04D				

Apply Cancel

Edge Connections Table

Editing the ALD Diagram

Clicking on the “Edit ALD Blocks” button brings up a chart which looks a bit like the ALD diagram page itself. Columns are numbered from 5 down to 1 (as they are in the ALD diagram itself) and the rows are similarly labelled A through I.

In between each block are entries which indicate the existence of “DOT Functions”. A dash (“-”) is for no DOT function, an “A” for a **logical** “AND” and an “O” for a **logical** “OR”. Connections to DOT functions cannot be edited directly when a gate connects to or from a DOT function. In these cases, the connection is accessed via the logic blocks which provide inputs to them or to which they provide their outputs. Note that these are *logical* functions, and are *arbitrary* and often differ from what the electronics / generated HDL will do.

The logic blocks themselves may be empty (do not appear on the ALD diagram sheet), or may have a machine/frame/gate/ECO designation/panel/row/column/card type designation, just as they appear on the ALD diagram pages. At the bottom of the block is also additional information added by the application: the identified **card** gate and the number of identified inputs / outputs to/from this logic block.

The ECO designation is a letter, which corresponds to the table of Diagram Page ECOs, just like it does on the original IBM ALD diagram page.

Clicking on an individual block brings up the Edit Diagram Logic Block dialog for editing the information associated with a given logic block.

Edit Diagram Blocks

Machine:
Page:

Volume:

	5	4	3	2	1	
A						A
B						B
C		11CA 1G03 DFE G1:2/1		11C. 1G03 DFE G4:3/1		C
D		11CA 1E11 DFE G1:2/3				D
E		11CA 1H13 DAC G1:3/1		11CA 1G03 DFE G3:3/1		E
F		11CA 1F04 DAX G1:2/1	11CA 1B09 DAX G3:2/1			F
G		11CA 1C04 DFG G5:2/1		11CA 1F03 DAW G2:3/1		G
H		11CA 1F03 DAW G3:3/3		11CA 1F03 DAW G1:3/1		H
I				11CA 1F04 DAX G2:2/1		I
	5	4	3	2	1	

Edit Diagram Blocks Dialog

Editing a Diagram Logic Block or DOT Function

Accessed by clicking on an individual logic diagram block, the Edit Diagram Logic Block dialog contains all of the information associated with an ALD diagram logic block, except for its connections. This includes its logic function, in IBM terms (Block Symbol), its input and output modes / levels (see logic voltage levels, above), the location of the card (machine/frame/gate/eco tag/panel/card location) and the card type. On the right is a graphic representation that is intended to match up with the representation on the printed ALD diagram page itself, which makes cross-checking a little easier.

Some blocks have a title indicating the use of a particular logic block which appears above the printed box.

Some “gates” (typically things like load resistors, capacitors, etc.) have outputs on the left hand side, in which case that box is checked.

Some logic gates (typically flip flops) occupy two squares on an ALD diagram page. In that case, the Extended box is checked, and an indication is made whether this block is extended to the one above it, for the one below it (they come in pairs, of course.)

The “Gate” pull down identifies which of the logic gates on a particular SMS card correspond to this particular logic block. This is done by matching up the gate with (the most) corresponding pins to the pins identified on the original ALD diagram page. Sometimes (especially with flip flops) the pins are arbitrarily shared between the two extended logic blocks, but for simple gates, they should correspond exactly.

Connections to and from other logic blocks or to and from signals on the sheet edges are edited by clicking on the “(Apply and) Edit connections” button. The “Apply and” is to remind the user that clicking on this button first applies any changes that may have been made on this dialog before opening the connections dialog.

Note: If the logic block has any connections, you cannot change the card type via the card type pull down – that would cause connections to get lost in the Ether where only the Ether Bunny can track them down.

The “No HDL Gen” check box is used to inhibit generation of HDL from this logic block.

The “Verified / Exempt from Checks” can be used to suppress messages regarding this logic block from one or more error checking reports.

The Block Notes section is used for any notes the users might want, but it is also used during HDL Generation:

- For switches, if the notes contain “NOVECTOR” then this switch is *not* included in the generated switch vector in a page group (group generation). This is typically used for multi-deck rotary switches where the extra decks are *logically* just copies of the primary switch deck – and can be most efficiently handled by an HDL assignment to the secondary deck from the primary deck.
- Again, for switches, a note can contain “ACTIVE HIGH”. Most switches are active low in SMS systems (i.e., turning the switch on causes the voltage to go to the low logic level), but in some cases switches are active high, and this note allows those to be handled properly during HDL generation.

This form has some special keyboard shortcuts:

- ALT+E: Selects the ECO combo box
- ALT+F: Selects the Frame combo box (which may have a different name – see above)
- ALT+G: Selects the Gate combo box (which may have different name – see above)
- ALT+M: Selects the Machine combo box
- ALT+N: Selects the Notes text box
- ALT+T: Selects the card Type combo box

Unfortunately I was unable to find a way to make the little dashed box appear that makes it clear the control is selected – but it does appear once the user first presses tab on the form at any point.

Edit Diagram Logic Block

Diagram Machine: 1401 Volume: 1401/Demo dummy 1401 volume set Volume: II

Page: 1.1.2.1

Diagram Column: 5 Diagram Row: D Block Title:

Feature: ☐ No HDL Gen. Block Symbol: +A

Input Mode/Level: S Output Mode/Level: Y ☐ Output on LEFT

Card Location: Machine: 1401 Module: A

Module: A ECO Tag:

Gate: 1 Card Column: A Card Row: 03

Card Type: DFQ Block Configuration:

Block Notes:

☐ Extended ☐ Above ☐ Below Gate: (NONE) ☐ Verified / Exempt from Checks

(Apply and) Edit Connections

Logic Block

```

+-----+
| +A |
|-----|
| S  Y|
| 01A |
| 1A03|
| DFQ |
+-----+
      5D
  
```

Apply Delete Cancel

Edit Diagram Logic Block Dialog

Edit Connections Dialog

Finally, by clicking on the Edit Connections button, we reach the page where connections are displayed. At the top are read only fields identifying the ALD diagram page and logic block that these connections relate to.

Then there are two tables which list and allow editing of an individual connection, along with buttons to Add Input or Output connections.

In these tables, the Pin identifies the input or output pin on this gate.

There are three types of connections, identified by the Type column, and which are captured in different tables in the database (column references here start with the "Pin" column as column number 1.)

- P – gate connections from or to pins on other logic blocks on this same page, in which case the third column contains the coordinate of the connected logic block and the associated pin.
- E – edge connections from or to signals identified on the edges of this ALD diagram page, in which case the third column identifies the other sheet upon which this signal originates or to which this signal is sent, and the last column contains the row and signal name. (See below for a note on a couple of special signal names.)
- D – dot function connections from or to "DOT functions" (wired and/or connections) on this page, or, rarely, with gates on other pages, in which case the third column contains the coordinate of the logic block to the LEFT of the DOT function, the fourth column contains an "A" or "O" for the logical function of the DOT function. The next column contains a "+" or "-", indicating whether the DOT function is defined using negative logic or positive logic. These last two columns are guesses based on the logic function identified on the gates which connect to the DOT function, and are *not* used during HDL generation.

An individual connection may be edited by clicking the "Edit" button before the first actual data column. A new connection is added by clicking the Add Input button or the Add Output button.

Edit Connections

Diagram Machine: 1411

Volume: 1411/Ohio - 2010 Volume: II

Page: 12.12.44.1

Diagram Column: 2

Diagram Row: C

Card type: DFE

Inputs to this Logic Block / Gate

Add Input

	Edit	Pin	Type				*Ref -or- Row / Signal Name
▶	Edit	H	P	4C	C		
	Edit	K	P	4D	C		
	Edit	L	E	19.10.03.1			C / -S START INTERRUPT

Outputs from this Logic Block / Gate

Add Output

	Edit	Pin	Type				*Ref -or- Row / Signal Name
▶	Edit	G	D	2C	O	+	

Edit Connections Dialog

Editing a DOT Function

To edit a DOT function, click on the “A”, “O” or dash (to add a DOT function) that exists between the larger logic block entries on the diagram.

This dialog is a combination DOT function editing dialog (where all one can specify is AND or OR) and a list of connections to and from the DOT function. The rest of the information in the top of the dialog is derived from the parent dialogs and is read-only. The Verified / Exempt from Checks checkbox can be used to suppress errors or warnings regarding this DOT function in certain error reports.

Note that connections to the DOT function from a logic block on the same page or to a logic block on the same page cannot be edited here. Instead, such connections are edited from the logic block gate involved.

One can add/delete/edit connections to the DOT function from the edge of the sheet (originating from other sheets) or from the DOT function to the edge of the sheet (sending signals to other sheets) from the Edit DOT function dialog. The resulting dialog is described later.

The “Force Logic Function” if present, is used directly during HDL generation to force a particular logic function (the default being “OR”, at least for now, for the IBM 1410, or based on the dotFunctionLogic column of the LogicLevels table, as discussed earlier). The Verified /Exempt check box is there to exempt this DOT function from producing an error message during certain reporting functions.

Machine: 1411 Page: 12.12.44.1

Volume: 1411/Ohio - 2010 Volume: II

Diagram Column: 2 Diagram Row: C Force Logic Function:

Logic Function
☐ AND ☒ OR ☐ Verified / Exempt from Checks

Connections (Gates are Read only)

	Edit	In/Out	Type	Coord.	Pin/*Ref	Polarity	Row / Signal Name
▶	(N/A)	I	P	2C	G	+	
	Edit	I	E	16.42.01.1			C / +S SET B CYCLE CTRL "ARITH"
	Edit	I	E	12.60.04.1			C / +S SET B CYCLE CTRL "BR OPS"
	(N/A)	I	P	2E	G	+	
	(N/A)	I	P	2G	E	+	
	(N/A)	I	P	2H	D	+	
	(N/A)	I	P	2I	C	+	
	Edit	O	E	12.12.21.1			C / +S SET B CYCLE CTRL

Edit DOT Function Dialog

Logic Block (Gate) Connections

The three types of connections to and from logic blocks are edited using the same dialog – the Edit Logic Block (Gate) Connection dialog. There are two major flavors of this dialog, depending upon whether one edits the connection from the input side or the output side. The machine, volume, page, diagram column/row and card type fields are read-only, filled in from the information from the parent Edit Diagram Logic Block and Edit Connections dialogs. To move a connection to a different gate, deleted it from one gate and add it back in the correct location.

For an input connection from another logic block on the same page, specify the input pin, select “Gate” from the Type radio button list, and specify the logic block coordinate and pin of the source of the connection. Some logic block connections (most often connections that are outputs to the sheet edge on the right hand side), there is a number reference that refers to the connection list near the bottom of the page. Those are specified using the *REF pull down list.

For an output connection to a gate, the dialog is similar, but specifies the gate which receives the signal as an Input. One can edit a gate-to-gate connection from either side of the connection. Also, for output signals, one specifies whether the connection is positive logic or negative logic using the Polarity radio buttons. Generally, connections that appear on top half of the output of a logic block in the original IBM ALD diagram are positive logic, and negative ones come out from the bottom half of the logic block.

Diagram Machine: 1411 Volume: 1411/Ohio - 2010 Volume: II

Page: 12.12.44.1 Diagram Column: 2 Diagram Row: C

Card type: DFE

Edit Input to Pin: H

Polarity: ☒ + ☐ -

Type

☒ Gate Coordinate: 4C Pin: C *REF

☐ Dot Function Coordinate:

☐ Edge Connector Row/Signal:

Source Sheet: *REF *REF

Apply Populate Delete Cancel

Input connection from another logic block on the same page

Inputs that come from the (left hand side) sheet edge are specified by selecting the input pin, as before, but then selecting the “Edge Connector” radio button. This enabled one to select the edge signal from the “Row/Signal” pull down, and then specify the sheet that is the source of the signal, along with up to two references, as described above.

For an output connection to an Edge, the dialog is similar, except specifying an output signal on the right hand side of the original IBM ALD diagram page.

There are two **special signal names**: “**LOGIC ONE**” and “**LOGIC ZERO**”. (Note the embedded space.) These are used (especially logic 0) when a connection to a gate is made to pin J (for logic one or +S) or M (for logic zero or -S). Most of the gates just ignore unconnected pins. A few that cannot (like triggers) will assume that an unconnected pin is logic 1, but if a logic zero is needed for a trigger input on an ALD, then the special “LOGIC ZERO” signal *must* be used. I entered them as normal input signals, originating from page “00.00.00.0” – but any other page you might like to use would be fine.

Diagram Machine: 1411 Volume: 1411/Ohio - 2010 Volume: II

Page: 12.12.44.1 Diagram Column: 2 Diagram Row: C

Card type: DFE

Edit Input to Pin: L

Polarity: ☐ + ☐ -

VV From the Output Below VV

Type: ☐ Gate ☐ Dot Function ☒ Edge Connector

Coordinate: Pin: *REF:

Coordinate:

Row/Signal: C / -S START INTERRUPT

Source Sheet: 19.10.03.1 *REF: *REF:

Apply Populate Delete Cancel

Input connection from a sheet edge

Inputs from DOT functions and outputs to DOT functions are specified by selecting the “Dot Function” radio button, and specifying the logic block coordinate which is to the immediate LEFT of the DOT function letter on the overall diagram.

Again, connections to and from DOT functions on the same page as the gates are edited like this, from the gates involved, rather than from the Edit DOT Function page.

Note that the *logical* (not electrical) type (And/Or) of the DOT function is not specified in the connection, but rather specified by editing the DOT function entry itself, as described earlier.

Edit Logic Block (Gate) Connection

Diagram Machine: 1411 Volume: 1411/Ohio - 2010 Volume: II

Page: 12.12.44.1 Diagram Column: 2 Diagram Row: E

Card type: DFE

Edit Input to Pin: A

Polarity: ☐ + ☐ -

WW From the Output Below WW

Type:

- ☐ Gate
- ☒ Dot Function
- ☐ Edge Connector

Coordinate: Pin: *REF:

Coordinate: 3E

Row/Signal:

Source Sheet: *REF: *REF:

Apply Populate Delete Cancel

Input connection from a DOT function

Editing DOT Function Edge Connections

As mentioned earlier, DOT functions can, sometimes, receive connections from a sheet edge or provide outputs to a sheet edge. These are accessed from the Edit DOT Function dialog page by clicking on the “Edit” button for the connection. A new edge connection for a DOT function is created by clicking the Add Input/Output button. (Again, connections to or from DOT functions from or to logic blocks (gates) on the same page are edited from the logic block dialog.)

For a new connection, one specifies the In or Out radio button. A DOT function sheet edge connection cannot be changed between input and output once created. Instead, delete the incorrect one, and add the correct connection.

As with logic block gate connections, the information at the top of the dialog is read-only, inherited from the parent dialogs. Once one specifies In or Out (when adding), one then selects a Row/Signal from the drop down list, specifies a source or destination sheet and optionally an edge connection reference from the drop down list.

For convenience, selecting the Row/Signal will pre-populate the Source/Destination Sheet.

The screenshot shows the 'Edit DOT Function Edge Connection' dialog box. It has a title bar with a standard Windows icon and window controls. The main area contains several input fields and controls:

- Machine:** A text box containing '1411'.
- Page:** A text box containing '12.12.44.1'.
- Volume:** A text box containing '1411/Ohio - 2010 Volume: II'.
- Diagram Column:** A text box containing '2'.
- Diagram Row:** A text box containing 'C'.
- Radio Buttons:** Two radio buttons labeled 'In' and 'Out'. The 'In' button is selected.
- Row/Signal:** A dropdown menu showing 'C / +S SET B CYCLE CTRL *ARITH*'. A small downward arrow is visible on the right.
- Destination Sheet:** A text box containing '16.42.01.1'.
- *REF:** A dropdown menu with a small downward arrow.
- Buttons:** Three buttons at the bottom: 'Apply', 'Delete', and 'Cancel'.

DOT function Edge Connection dialog for an input to the DOT function

Editing Cable/Edge Connections

The Cable/Edge Connection dialogs are similar in hierarchy to the ALD Diagram pages, but a bit simpler.

The top of the Cable/Edge connection diagram hierarchy is the Cable/Edge Connection page, which is accessed via Edit => Cable/Edge Connection Pages, and selecting the machine, volume set, volume and page. Selecting the machine, volume set and volume pre-populates the “Select Page” pull down. One can also add a new page to the volume using the “New Page” button.

For a new page, fill in the Page Name (e.g. 11.04.02.3) which appears at the top of the page on the right, the Part Number, which appears at the top of the page on the left, and the title, which appears at the top of the page in the middle, and click “ADD”, and then select the page you just added to proceed (as of this writing, it is not automatically selected for you).

Once a page is selected, the Cable/Edge Connection Page ECO’s table is accessible. This is just like the one for the ALD diagrams. At the bottom of each Cable/Edge connection page in the drawing set is a list of one or more ECO’s with an accompanying ECO identification label (starting with “A”), and a date, which is entered into the Cable/Edge Connection Page ECO’s table.

Then click “Edit Cable/Edge Connection” button to edit the individual cable/edge connections. That produces a matrix of blocks similar to that of an ALD Diagram, but instead of containing logic blocks, this contains representations of cable/edge connection blocks for the following “card types”

- “CONN” – On an IBM 1410, these are flat ribbon cables that go between panels of a given frame/gate or between frames.
- “CABL” – These are cables other than the flat ribbon cables
- “RPQ” – These are card slot cable connections reserved for “request for price quotation” enhancements to a machine.
- “SAVE” – These serve a similar function to “RPQ” slots
- “STRL” – These are connection slots reserved for use with the Pound Sterling RPQ

As with an ALD diagram, clicking an individual block brings up a dialog to edit the characteristics of that block. The sheets that document cable/edge connection blocks are labeled using the same sheet coordinate system (rows A-I, top to bottom and columns 1-5, RIGHT TO LEFT) as the ALD diagram sheets.

The Connector Check report can be used to check cable/edge connections identified on these pages for consistency.

Machine:
Page:

Volume:

	5	4	3	2	1	
A	11CA 1A01 CONN	11CA 1A22 CABL	11CA 1A26 CABL	11CA 1A27 CONN	11CA 1A28 CONN	A
B	11CA 1B01 CONN		11CA 1B28 CONN	11CA 1C01 CONN	11CA 1C27 CONN	B
C	11CA 1C28 CONN	11CA 1D01 CONN	11CA 1D27 CONN	11CA 1D28 CONN	11CA 1E01 CONN	C
D	11CA 1E27 CABL	11CA 1E28 CONN	11CA 1F01 SAVE	11CA 1F26 RPO	11CA 1F27 CONN	D
E	11CA 1F28 CONN	11CA 1G01 CONN	11CA 1G27 CABL	11CA 1G28 CONN	11CA 1H01 CABL	E
F	11CA 1H27 CONN	11CA 1H28 CABL	11CA 1J01 CONN	11CA 1J27 CONN	11CA 1J28 CONN	F
G	11CA 1K01 CONN	11CA 1K02 CABL	11CA 1K03 CONN	11CA 1K04 CONN	11CA 1K05 STRL	G
H	11CA 1K26 CONN	11CA 1K27 CABL	11CA 1K28 CONN			H
I						I
	5	4	3	2	1	

The Edit Cable/Edge Connection Blocks “matrix” Dialog

Editing a Cable/Edge Connection Block

The page for editing an individual cable/edge connection block, which corresponds to the connection block on the page in the original documentation, is accessed by clicking on that block. If the only information for a cable/edge connection block comes from the card location pages (i.e., has never been edited and saved), the characters in the block will be grey. Once edited and saved, the characters will appear black. (In the example below, all of the blocks had been edited). The Machine, Volume, Page, Diagram Column and Diagram Row are read-only as with the ALD diagrams.

For a cable/edge connection block there are two general areas. The top part relates to where the cable comes from, or its source. The bottom portion, below the horizontal bar in the dialog relates to where the cable goes to, or its destination. To the right is a display in a format similar to the one that appears in the original documentation.

Both the source and destination have a:

- Machine
- Frame and Gate (for the IBM 1410 they are the same)
- Panel
- Card Row
- Card Column

All of these are pull-downs except for the Card Column, which is a number corresponding to column in the card panel in the machine (not to a diagram column).

On the IBM 1410 and likely other machines, lots of connections follow a pattern. For example, a connector that is in column 01 of a panel would typically go to 28 of the panel next to it. 02 to 27 and 03 to 26 are the same – so the cables do not cross each other. In the parlance of this application, those are called “implied destination” because they don’t actually appear in the diagram. Indeed in the IBM installation manual for the IBM 1410 there is language which reads, for 1411D(D)1K01 “In this case socket K28 is assumed since it is the corresponding socket in 1411C2 and there is no note to indicate otherwise”

When first editing a block that is new but that appears in a card location diagram, if that block has a corresponding entry in the Cableimplieddestinations table, then the Destination will show that implied destination and the “Explicit Destination” checkbox near the bottom will be checked, rendering the destination fields read-only.

The destination can be changed between an implied destination and an explicit one by checking the Explicit Destination checkbox. If the Checkbox is cleared, but there is not matching rule for the source, a warning message appears, and the checkbox is left as checked.

The card types other than “CONN” and “CABL” cause special things to happen in the drawing on the right, clear the explicit destination check box (making the destination read only, because if the entry is not “CONN” or “CABL” there is no known destination).

Because the original pages I worked from did not show the card types other than “CONN” and “CABL” in a consistent way, the drawing on the right may not exactly match the original. In particular, many of the originals show the source Machine (2 digits), Frame and Panel for these card types, but did not always do so, so that top line is blank in the application.

At the very top of the drawing, above the rectangle is what I called a “Top Note”, and it can be entered to match the original drawing. If that is left blank by the user, and then an explicit destination is identified, then when the entry is saved and then opened again, that Top Note will be automatically filled in. Save it again, and it gets saved in the database. (Having the application fill that in during the first editing of the destination proved tricky enough that I didn’t bother doing that.)

There are two checkboxes on the form that relate to reporting, as well:

- The “No Conn. Check” checkbox is used to tell the reporting subsystem that it should skip checks involving this box relating to consistency with a partner check box being consistent with this one. This would typically be used where the other end of the connection is something like a CE panel or (on the IBM 1410) panel 7 which is used to connect to cables to external units.
- The “No Edge Check” checkbox is used to tell the reporting subsystem that it should skip checks involving this box relating to the Edge Connector Location Lists that appear on the bottom of ALD Diagram sheets.

As with the other dialogs, the Apply button applies updates, the Delete button deletes the entry (but leaves any underlying card location data from the card location chart entries alone), and Cancel cancels any changes.

Diagram Machine:

1401

Volume:

1401/Demo dummy 1401 volume set Volume: II

Page:

1.1.1.10

Diagram Column:

5

Diagram Row:

A

Top Note:

TO 01A1A01

Conn. Location: Machine:

1401

Module:

A

Module:

A

ECO Tag:

Gate:

1

Column:

A

Row:

02

Card Type:

CONN

☐ No Conn. Check

☐ No Edge Check

Destination

Machine:

1401

Module:

A

Module:

A

Gate:

1

Column:

A

Row:

01

☒ Explicit Destination

Apply

Delete

Cancel

Cable/Edge Connection Block

TO 01A1A01

|01A1|

|CONN|

|01A1|

|01A |

|1A02|

|----|

5A

The Edit Cable / Edge Connection Block Dialog (Above)

cableSource	cableImpliedDestination
1411,B,B,2,*,28	1411,C,C,1,*,01
1411,C,C,1,*,01	1411,B,B,2,*,28
1411,C,C,1,*,27	1411,C,C,2,*,02
1411,C,C,1,*,28	1411,C,C,2,*,01
1411,C,C,2,*,01	1411,C,C,1,*,28
1411,C,C,2,*,02	1411,C,C,1,*,27
1411,C,C,2,*,26	1411,C,C,2,*,26
1411,C,C,2,*,27	1411,D,D,1,*,02
1411,C,C,2,*,28	1411,D,D,1,*,01
1411,C,C,3,*,01	1411,B,B,4,*,28
1411,C,C,3,*,27	1411,C,C,4,*,02
1411,C,C,3,*,28	1411,C,C,4,*,01
1411,C,C,4,*,01	1411,C,C,3,*,28
1411,C,C,4,*,02	1411,C,C,3,*,27
1411,C,C,4,*,27	1411,D,D,3,*,02
1411,C,C,4,*,28	1411,D,D,3,*,01
1411,D,D,1,*,01	1411,C,C,2,*,28
1411,D,D,1,*,02	1411,C,C,2,*,27
1411,D,D,1,*,03	1411,C,C,2,*,26
1411,D,D,1,*,27	1411,D,D,2,*,02
1411,D,D,1,*,28	1411,D,D,2,*,01
1411,D,D,2,*,01	1411,D,D,1,*,28
1411,D,D,2,*,02	1411,D,D,1,*,27
1411,D,D,3,*,01	1411,C,C,4,*,28

Some Cable/Edge Connection Implied Destination Rules

Importing Data

As described earlier, development started by creating some spreadsheets with data, as doing so was bound to be much more efficient than using the application to enter some things (like the card location charts.) Also, this allows the application to pre-populate quite a bit of information, which helps speed things along.

The spreadsheets used originally to import data (and, in some cases, kept up to date as errors were found while editing pages later) can be found in folder

IBM1410-ImportedSpreadSheets

The main spreadsheet is IBM1410-Drawing-Data.xlsx.

Individual .csv files were then created by exporting information from this spreadsheet to use by the import classes in the application.

Notice the use of past tense – these were one-time imports. Importing them again would be disastrous.

Data which can be imported includes:

- (Machine) Features
- SMS Card information (type, pins, internal connections, etc.)
- (Card) Location Pages to pre-create the pages containing card location charts
- Card Location Charts
- Tie Downs
- Bussing of groups of related signals representing bits of a given signal register or bus (std_logic_vector)

Importing of cable interconnection pages has not yet been completed – was done manually.

Logic Synthesis

As stated earlier, the end-goal of this project is to be able to recreate the logic for an IBM SMS machine (and the IBM 1410 in particular) from the information gathered in the database.

The beginnings of this process can be found in the “Generate HDL” portion of the menu.

The Generate HDL entry in the menu allows generation of HDL from individual pages. It also, optionally, automatically generates a test bench file for that page – or if such a test bench already exists, updates it, bringing user test bench code in the designated area forward to the updated test bench. It will accept “%” SQL wildcards in order to generate HDL for multiple individual ALD pages.

Currently code only exists for VHDL generation however the class framework of the application is designed to support generation of more than one kind of HDL.

There are some special cases that Generate HDL recognizes in logic diagrams:

- It recognizes logic blocks with extensions, and merges them.
- It skips blocks marked for no HDL generation in the database
- It recognizes combinatorial “loops” – latches, and follows each gate in the loop with a “D” flip flop. (It might be enough to break the loop with a “D” flip flop in just one place, however.)
- It checks for cases where a DOT function has exactly two inputs, one of which is from a Trigger, with exactly one output. This indicates a special case where in SMS logic, the non-trigger gate DOT-ed with a trigger was used to force the trigger into a given state – essentially acting as another DC Set or DC Reset input – which is how the HDL generator handles it, by creating a “faux” input. Currently, pins are only 1 character long. Fortunately, Triggers are always single height cards (at least on the IBM 1410), so the faux input pin is named via original pin name – ‘A’ + ‘S’ (S is the first pin on the second have of a double height card.) This could be generalized by allowing two character pin names, should this not work on, for example, a 7094. This allowed leaving the ALD as it was originally, but generating correctly functioning logic by changing the connections during HDL generation.
- It checks to see if the DOT Function has a forcedLogicFunction column specified, and if so, this overrides the normal “OR” logic function (and also overrides any special cases listed below.)
- It checks to see if a DOT function has the same logic levels on *all* of its inputs and outputs, and if so, and if that logic level has a dotFunctionLogic column value, that will override the default of “OR”.
- It checks for the special case where a DOT function in the ALD is fed ONLY from one or more rotary switches and they are all active low (the default unless the notes field for the switch contains the string “ACTIVE HIGH”). In such a case, the DOT function logic function is set to AND.

- It checks for the special case where a DOT function in the ALD is fed ONLY from signals whose names begin “-C”. I am not sure exactly why, but electrically speaking, these function as AND gates (logically as OR gates, though – but using negative voltage as logical one) – both inputs must be at a positive voltage to generate a positive voltage result.
- Lamps are generated as output signals of the form LAMP_mmfrcc (e.g., LAMP_15A1A06) on individual pages, and are also recognized as outputs when generating group pages. This actually requires spinning thru the individual pages’ diagram blocks during generation, but I felt this was preferable to introducing lamps as an output signal which did not appear on the original ALD pages. (They are then also good candidates for setting up as HDL busses – see below)
- Oscillators are set up with gates that are configured with a Logic Function of Special and an HDLName of Oscillator and generate a generic with a parameter FREQUENCY with an integer value in KHz, a parameter CLOCKPERIOD measured in ns and include the FPGA_CLK. This frequency is derived from the TITLE of the logic block that instantiates the SMS oscillator “gate” (e.g. 1.5 MC is 1500 KHz). It does not use the component value in the gate because that is fixed for that gate. There is a parameter “fpgaclockperiod” in table parameters that is used in this calculation, to set the CLOCKPERIOD paramter. If it is not present, a 100Mhz clock (10ns period) is assumed.
- Delay lines are set up with gates that are configured with a Logic Function of Special and an HDLName of ShiftRegister or InvShiftRegister (the latter being those with inverted outputs). The time can be specified in “ns” or “us” in the logic block title. If the HDL generation cannot figure out the time, it issues a message and sets the delay to 0. [On the IBM 1410 I think that these only appear in two places: in the clock generation circuit on the first page, where no delay is needed, and in the memory circuits which probably won’t be used. However, I also added one on page 12.65.01.1, where originally a capacitor was used to delay a signal during the power on/computer reset sequencing.]
- Single shots are set up with gates that are configured with Logic Function “SS” and an HDL Name of “SingelShot” and generate a generic with a parameter of PULSETIME with an integer value in ns and a parameter CLOCKPERIOD measured in ns, as with the oscillator. As with the oscillator, the pulse time is derived from the TITLE of the logic block that instantiates the SMS one shot gate – a number followed by US, USEC, MS or MSEC. The single shot uses the same parameter “fpgaclockperiod” as do oscillators, which is used to set the value of the generic CLOCKPERIOD parameter.
- Switches come in different types. They are recognized by the Logic Block SYMBOL. All are assumed to be coming in from the outside as active high, even if the original switch was not.
NOTE: If a TOG, MOM or REL switch connects to more than two outputs, there will be a warning message to double check the switch outputs, just in case. Also, for TOG, MOM and REL switches, the code takes pains to not output the logic assignment to a given pin more than once. This could also be done for ROT switches, but currently this is not done.
 - “TOG” and “ALT” switches are toggle switches. They can have two outputs. If there is an active high output, it should either have a mapPin of OUTON or a pin N. If there is an active low output, it should either have a mapPin of OUTOFF or be pin T.

- “MOM” are momentary switches, and are handled the same as toggle switches. Note that any timing that might result from an attached RC network in the ALD is expected to be handled by whatever is feeding the switch signal.
- “REL” are momentary relay switches. These are like MOM switches, except that they default to active low (i.e., OUTON (typically pin N) is negative going on activation).
- “ROT” are rotary switches. Rotary switches are handled as a VHDL bit vector. Each pin must have a mapPin of the form PIN## where ## is a two digit number. This identifies the bit in the bit field. Ordinarily bit 0 is not used and is wasted (it corresponds to the input pin, if any). In the bit vector a “1” bit is used to indicate the position of the switch. However, on real machines these are usually active low and are sometimes connected together in a kind of DOT function, and the following gates are typically set up to expect an active LOW signal. So, to make that work right, these get a NOT prefix in the generated HDL UNLESS the notes field for the logic block contains the string “ACTIVE HIGH”. Also, see above for a special DOT function where two terminals of a rotary switch are joined together in what is entered on the ALD in the application as a DOT function. In these cases, the logic function for the DOT function is set to AND.
- A special sub-case of rotary (ROT) switches: If they are active high, then instead of reading out as just a switch position, if they have a single input, then that signal is passed through the switch. (Currently it does NOT verify that the input signal is not simply a -V (logic 0) signal, but it may have to in the future).
- There are two **special signal names**: “**LOGIC ONE**” and “**LOGIC ZERO**”. (Note the embedded space.) These are used (especially logic 0) when a connection to a gate is made to pin J (for logic one or +S) or M (for logic zero or -S), as described above. These are translated into the appropriate literals (for a given HDL) and removed so they don’t appear in the generated HDL entity input list. These signals are also ignore during group generation.

There are four template files that logic generation uses so that the user can customize things like VHDL library names, test bench termination and template test bench code:

- HDLTemplate.ext (e.g. HDLTemplate.vhdl) This file is placed near the start of every HDL file generated for both the page HDL and its test bench.
- TestBenchDeclares.ext This template file is inserted after the signal declarations and before the “begin” of the architecture the first time a given test bench is generated. It can be used to declare signals and processes to be included in test benches. If the test bench file already exists, the code is instead copied from the existing test bench.
- TestBenchTemplate.ext This file is inserted near the end of the generated test bench the first time a given test bench is generated. If the test bench file already exists, the code is instead copied from the existing test bench. In this file the tag <FPGA CLOCK> denotes where the FPGA clock template code will be inserted (see next file)

- **TestBenchFPGAClock.ext** This file simulates the FPGA clock using a simple process. It is inserted into the test bench when it is first generated at the location of the <FPGA CLOCK> tag in the test bench template file.

The Generate Group entry is designed to support generation of HDL from a group of pages, so that their edge signals can be combined. As with individual pages, there is also the option to generate an associated test bench. The Page(s) to generate can use “%” style SQL syntax to specify the group of pages to use as input to the process. The test bench generation process uses the exact same template files identified above (i.e. the template files are used for both individual page test benches and for page group test benches). Group generation also allows the user to specify that vectors for lamps and switches be generated in the test. This is typically used when generating at the topmost level (“Page(s) to generate set to “%)” and consolidates all of the switches into a single vector, and all of the lamps into another single vector. Finally, Group generation also provides generation of the sizes and indexes for the aforementioned vectors in C# as *const int*, for use by a host support program so that if those vectors change, only the indexes need to change.

NOTE: There is a new database column which allows assigning diagram pages to a “subsystem” but the HDL group generation has not yet implemented that.

In addition, the VHDL Generate Group entry supports creation of busses. If a signal appears in the BusSignals table in the database, it will be recognized as a bussed signal in the HDL. Bussed signals are set up such that the lowest bit defined in the BusSignals table 0 is the low order bit – in VHDL they are declared as STD_LOGIC_VECTOR (*top bit DownTo lowest bit.*).

For *inputs* the group will have the entire bus in its input VHDL ports. The bus is then “ripped” down to individual bits to pass to any given ALD diagram, which typically only actually uses one or two bits. In the associated test bench, these are initialized to a string of 1’s for active low signals, or -0’s for active high signals for test purposes, much as individual signals are initialized in the generated test bench. One exception: If the signal is defined as a bus, but it is internal to the pages in the group (is not an input from or output to a page outside of the group), then it is not “ripped” in this fashion.

For *outputs* things are more complicated. If all of the bits of the bus defined in the database are not present as outputs among the grouped sheets, then the bus is not used or generated, and generation for the group proceeds as if the signal were not bussed. If and ONLY if ALL of the bits defined in the database are present as outputs in the grouped sheets, then several things happen:

- In the Port declaration for the group, the bus name is used as an output.
- Because HDL does not always allow using something declared as an output internally, a set of “buffer” signals are generated, one for each signal that participates in the bus.
- An aggregate assignment is made aggregating the individual signals to the bus name.
- Therefore, unlike input bus signals, the bus name is NOT “ripped” to feed to individual sheets in the group, whether as input or output. Instead, the buffer signal mentioned above is used.

- If the bus definition in the database “skips” bits (e.g., the IBM 1410 Operation Register does not contain a Word Mark bit – bit 6), then the aggregate assignment will include a ‘0’ for the “missing” bits.

Reports and Queries

The Reports menu is for generating various types of reports that may turn out to be useful in order to locate and correct errors in the (manually) captured data.

- Card Type Usage Report, which creates a report of every ALD page on which a particular card type appears (or “ALL” for any card type), its coordinates, its levels (In and Out) and its pins, as an aid to locating errors. It can now be optionally filtered by ALD Logic Block symbol, Particular gate on the card and the ALD logic block output polarity, as an aid to tracking down errors identified in the Connection Errors Report. Output is to a Data Grid View, which can be easily copied.
- Connector Check Report, which checks the connectors and cables on cable/edge connector sheets against each other. Output is to a dialog – the text can be easily copied.
- Edge Connection Check report, which checks sheet edge connections appearing at the bottom of ALD diagrams for inconsistencies among the connections which appear all of the sheets, and for inconsistencies or omissions with respect to the cable/edge connector sheets. The check can be run either including the pin (useful for signal-level checks) and without the pin (useful for checks against the cable/edge connectors.) It is recommended to first run the check with the pin, resolve those errors, and then run the check without the pins and resolve those errors, and then repeating the two step process until you are satisfied with the results. Perfection is not likely unless you have a completely consistent print set for a given actual serial number. Output is to a file.
- Signal Query: This is a signal cross-reference query. Enter a signal name pattern in SQL “LIKE” syntax (% wildcard). It lists each signal that fits the pattern, and the pages it is referenced from. Page(s) which output the signal are asterisked, and listed first. Output is to the dialog as text (which can be copied)
- Signal Report: This checks signals for consistency – things like ensuring every signal has exactly one source.
- Connection Checks – the Connection Errors Report. This report makes numerous checks on DOT Functions, ALD Logic Blocks, consistency of ALD logic blocks with the standard logic function for the card gate used, looks for cases where gates connected to a DOT function have more than one member which is not open collector, and so on. This report uses a special data base table, logicCheckRules (which does NOT have an editor – use your database tools to modify it) which identifies the rules to follow for a given machine. The rules are ordered by the identified priority, and checked sequentially for a match. This report currently takes a few minutes for the IBM 1410, so blinks “Working” to let you know it is busy. Output is to a file.
- databasecheck.py is a standalone Python program for doing some consistency checks of the database. Mostly these are related to referential integrity at a logical level.

	idLogicCheckRule	machineName	priority	logicFunction	diagramBlockSymbol	logicBlockType	char1	char2	lastChar	outputPolarity	comment
▶	1	1411	1	NAND	*	and	+	*	*	+	NULL
	2	1411	1	NAND	*	or	-	*	*	+	NULL
	3	1411	1	NOT	*	inv	I	*	*	+	NULL
	4	1411	1	NAND	*	inv	*	*	*	+	NAND gate as inverter
	5	1411	2	NOT	*	drv	D	*	*	+	NULL
	6	1411	2	NOT	*	and	+	*	*	+	Inverter with +AA symbol
	7	1411	3	Trigger	T	*	T	*	*	*	NULL
	8	1411	4	Resistor	R	*	R	*	*	*	NULL
	9	1411	4	Resistor	L	*	L	*	*	+	NULL
	10	1411	4	Capacitor	CAP	*	C	*	*	*	NULL
	11	1411	5	EQUAL	*	drv	D	*	*	-	NULL
	12	1411	5	EQUAL	DL	drv	D	*	*	+	NULL
	13	1411	6	EQUAL	DE	drv	D	*	*	+	NULL
	14	1411	8	AND	*	and	+	*	*	-	NULL
	15	1411	8	AND	*	or	-	*	*	-	NULL
	16	1411	10	NOR	*	and	-	*	*	+	NULL
	17	1411	10	NOR	*	or	+	*	*	+	NULL
	18	1411	10	NOR	*	inv	I	*	*	+	NOR gate as inverter
	19	1411	12	OR	*	or	+	*	*	-	NULL
	20	1411	12	OR	*	and	-	*	*	-	NULL
	21	1411	14	Special	*	*	*	*	*	*	NULL

The Logic Check Rules table