Product Specification

AHA Galaxy

Simulation Tool Kit User's Guide

PSGalaxy_STIK-0100

A subsidiary of Comtech Telecommunications Corporation

2345 NE Hopkins Court ■ Pullman WA 99163 ■ tel: 509.334.1000 ■ fax: 509.334.9000 ■ www.aha.com ■

Table of Contents

Figures

1.0 INTRODUCTION

The Galaxy Simulation Tool Kit is a Turbo Product Code (TPC) simulator and Application Programmers Interface (API) that enables communication systems developers to integrate TPC coding/decoding technology with system level simulations. It allows system developers to evaluate the performance of TPC technology under various modulation formats and channel models prior to building a hardware based system.

Two API options are available, a C/C++ API and a Matlab API. The C/C++ code API includes a sample project (developed with MSVC++ v6.0) which gives samples of all API calls. The Matlab API also includes a sample script (developed with Matlab v5.0) which illustrates the API call. Included with both APIs is an electronic user guide.

This user guide is intended to assist designers in understanding how to operate the Galaxy Simulation Tool Kit. It is also a reference guide for understanding the various parameters that the user will specify prior to performing simulations of Turbo Product Codes (TPCs). Additional reference materials regarding TPCs are available from AHA in the form of Application notes and product specifications

Please note that the Galaxy Simulation Tool Kit is a licensed product subject to AHA's software license agreement. A copy of this agreement is included in the appendix of this document for your reference. Be sure to carefully review this agreement as you will be asked to accept the agreement when you run the software on a computer.

1.1 CONVENTIONS, NOTATIONS AND DEFINITIONS

- Code block - A data stream to be encoded or decoded is segmented into blocks for processing by the TPC core logic. Data in a code block is configured as a 2D, 3D or enhanced array.
- Axis iteration - Decoding one axis of an array (all x rows, all y columns, or all z columns).
- Full iteration - Decoding all axes of an array (all rows and columns).
- Soft value - Input to the decoder from either an Analog/Digital Converter(ADC) or digital demodulator. Soft decode inputs to the TPC decoder are used as confidence estimates of the binary value.
- Code rate - Ratio of the number of data bits to the number of data and ECC bits.
- Axis code rate - Ratio of the number of data bits to the number of total bits for a given axis.
- Data rate - The rate at which unencoded data is input to the device when encoding or output from the device when decoding.
- Channel Rate - The rate at which encoded data is output from the device when encoding or input to the device when decoding. Note that system channel rate may be different due to external synchronization marks or other overhead.
- (n1,k1)x(n2,k2) - A general representation of a 2D block code for use in the descriptions to follow in this specification. For example, in a (64,57)x(64,57) code; n1,n2=64 represents the length of the data + ECC bits, and k1,k2=57 represents the length of only the data bits. 3D codes are represented as (n1,k1)x(n2,k2)x(n3,k3)
- Vector - One row or column of data in a block.
- Latency - The time from the first bit of a block in to first bit of the same block out.
- Hex values are represented with a prefix of "0x", such as register "0x00". Binary values do not contain a prefix.

2.0 GALAXY SOFTWARE LICENSE

The Galaxy Simulation Tool Kit is a licensed product subject to AHA's software license agreement. The software is provided for evaluation purposes only. The software cannot be copied or shared to unauthorized users. By installing the software the user is obligated to abide by the terms of the software license. A complete copy of this agreement is included in appendix A of this document for your reference.

3.0 SOFTWARE INSTALLATION

Insert the Galaxy Simulation Toolkit CD and run the setup program. The program instructs the user to select a directory to place the Galaxy software and documentation. The following directories are created, and the files expanded and copied to the appropriate directory:

- AHA TPC Simulation
- Galaxy Toolkit
- Matlab Toolkit
- TPC Publications

3.1 SYSTEM REQUIREMENTS

The minimum recommended system for using the Galaxy simulation toolkit is a Pentium class PC running at least at 400 MHz with 128 MBytes of RAM. It will run on either Windows®95/98 or Windows NT. Approximately 10 MBytes of free disk space is required for installation. The C/C++ API requires Microsoft Visual C++ 5.0 or higher. The Matlab API requires Matlab version 5.3 or higher.

4.0 GALAXY CONFIGURATION

Prior to encoding or decoding using Galaxy, configuration parameters must be set. During encoding, the device appends Error Correction Code (ECC) bits to the current block and then outputs the encoded block. When decoding, Galaxy accepts soft decision values and stores the data internally. The block is then decoded iteratively by running it through the soft in/soft out (SISO) decoder. Galaxy iterates to the maximum programmed iteration limit set by the *numIter* configuration parameter.

4.1 CODES

Galaxy Turbo Product codes are applied along orthogonal axes as 2 dimensional (2D) product codes, 3 dimensional (3D) codes, and enhanced product codes. The codes applied along each axis do not have to be the same (e.g. the blocks do not have to be square). Table 1 lists the constituent code choices for the X, Y, and Z axes. Note, code shortening can be applied along any axis to achieve an exact block size as required by the application.

Table 1: Galaxy Constituent Codes

AXIS CODE	*AXIS CODE RATE*	*CODE TYPE*
(256,247)	.965	Extended Hamming
(128,120)	.938	Extended Hamming
(64,57)	.891	Extended Hamming
(32,26)	.823	Extended Hamming
(16,11)	.688	Extended Hamming
(8,4)	.500	Extended Hamming
(256,256)	.996	Parity
(128,127)	.992	Parity
(64,63)	.984	Parity
(32,31)	.969	Parity
(16,15)	.938	Parity
(8,7)	.875	Parity
(4,3)	.75	Parity

In addition to the 2D and 3D codes, Galaxy also supports enhanced Turbo Product Codes ("hyper" parity axis) which provide another dimension of error correction. Enhanced TPCs are parity codes applied along diagonals (the "hyper" axis) in the 2D or 3D blocks.

The choice of code rate depends on many things including desired overall code rate, data rate, gate count, latency requirements, etc. The overall code rate is the product of the axes code rates.

4.2 ITERATIONS

Decoding is done in an iterative fashion. Each full iteration begins by passing an x-row into the Soft Input Soft Output (SISO) decoder. The SISO output is multiplied by a programmable X feedback (*xfbk*) value. The completion of all x-rows constitutes one axis iteration.

Next, each y-axis column is passed into the SISO decoder. The SISO output is multiplied by the programmable Y feedback value. The completion of all y-columns constitutes one axis iteration.

If a 3D code is being decoded, each z-axis column is passed into the SISO decoder. The SISO output is multiplied by the programmable Z feedback (*zfbk*) value. The completion of all z-columns constitutes one axis iteration.

One full iteration is completed when one X and one Y axis iteration is complete for a 2D code; or one X, one Y, and one Z axis iteration is complete for a 3D code. The iterations continue until the iteration counter equals the number of iterations set in the *numIter* parameter.

4.3 FEEDBACK

The TPC algorithm uses feedback, or weighting, values for performance tuning. After each axis iteration, the output of the Soft In, Soft Output (SISO) Decoder is multiplied by the feedback constant for that axis. These values are then fed back into the SISO for future iterations.

The feedback multiplier values used for each code axis can vary from 1/32 to 31/32 depending on the number of iterations and system parameters (soft input bits, resolution). The feedback multipliers must be tuned to give optimum decoder performance in a given system.

The following describes the tuning process. The choice of feedback multiplier has no effect on throughput or latency. For 2D square codes (*Xcode* = *Ycode*), a typical feedback multiplier value for both axes at 3 or 4 iterations is 16/32. For 3D cubic codes (*Xcode* = *Ycode* = *Zcode*), a typical feedback multiplier value at 6 iterations is 12/32.

When using non-square or non-cubic codes, the following general rules should be applied. Parity codes should have their feedback multiplier values set higher than Hamming codes when mixed. For example, in a (32,26)x(32,26)x(4,3) code, the X and Y feedback (*xfbk* and *yfbk*) multipliers should be set to 16/32 while the Z feedback (*zfbk*) should be set to 18/32 or 20/32. When mixing Hamming codes with shorter Hamming codes, the feedback multiplier should be set slightly higher for the shorter code.

For example, in a (64,57)x(32,26) code, the X feedback (*xfbk*) multiplier could be set to 16/32, while the Y feedback (*yfbk*) multiplier could be set to 18/32.

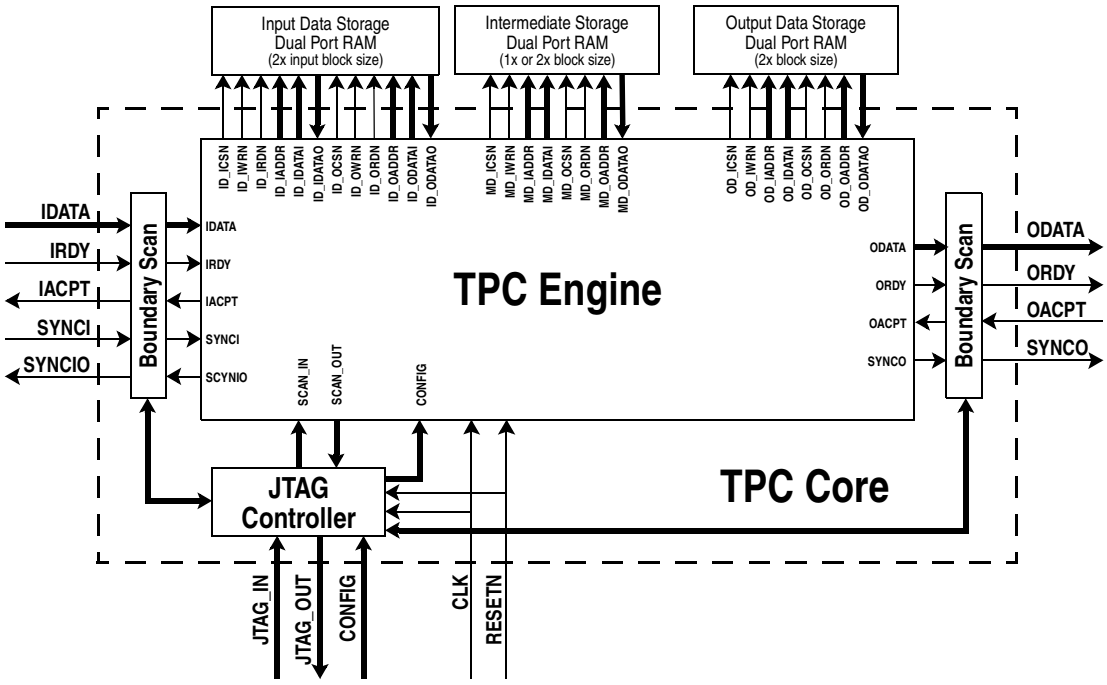
The feedback values must be tuned for the number of iterations allowed in a system. For less iterations than the above guidelines, the feedback values should be increased. For more iterations, the values should be decreased. For example, when using a (64,57)x(64,57) code with only 2 iterations, the feedback multiplier for both axes should be set to 20/32. Conversely, in a system that allows 12 or more iterations, the value for the feedback should be lowered to 16/32.

The feedback may also need to be tuned depending on the number of soft input bits (*quant_size*). This parameter will only affect the optimum feedback multiplier value slightly, meaning that it should be adjusted by only 2/32 or 4/32 to allow for these differences. Since systems vary widely, the system designer should experiment with various feedback multiplier values to obtain the best performance. Recommended starting values for feedback are listed in Table 2. The code combination shows 2D and 3D combinations of Hamming and parity codes.

Table 2: Suggested Feedback Starting Values

CODE COMBINATION	*X FEEDBACK (xbck)*	*Y FEEDBACK (ybck)*	*Z FEEDBACK (zbck)*
2D Hamming, Hamming	16/32	16/32	N/A
2D Hamming, Parity	20/32	20/32	N/A
2D Parity, Parity	30/32	30/32	N/A
3D Hamming, Hamming, Hamming	12/32	12/32	12/32
3D Hamming, Hamming, Parity	16/32	16/32	20/32
3D Parity, Parity, Parity	30/32	30/32	30/32

Figure 1: Galaxy Block Diagram



4.4 SOFT DECISION INPUTS

The inclusion of confidence information input to the decoder can significantly improve the performance of the decoder. The confidence information is in the form of soft decision bits from the demodulator. The more bits of soft information that are available, the more powerful the error correction.

Two parameters that determine how Galaxy interprets soft decision input data are *quant_size* (Quantization Size) and *quant_mult* (Quantization Multiplier). *Quant_size* is the number of bits for each data value input to the Galaxy decoder. *Quant_mult* is a scalar to be multiplied times each input data value.

The multiplier improves the performance of the decoder when using smaller values for *quant_size* (1, 2, or 3 bits). For example, if a system is using 2 soft bits (values range from 0 to 3), and the internal decoder resolution is set to 5 bits (values range from 0 to 31), then a *quant_mult* of 9 could be used to increase the range of the input values (new range is from 0 to 27). The software will automatically add the correct constant to center the input values in the internal resolution range.

Quant_mult must be an odd integer. A general rule for setting the *quant_mult* is:

$$quant\_mult \leq \frac{2^{inres-1}}{2^{quant\_size-1}}$$

4.5 HELICAL INTERLEAVING

The Galaxy core can optionally scramble (helical interleave) when encoding and descramble when decoding. Scrambling data spreads bursts of noise across all axes of the block code for the best error correction performance in burst channel use. The scrambling is applied after encoding takes place. Descrambling takes place before the decoding operation.

Helical interleaving is applied along a diagonal path through the encoded block. Data is output along diagonal lines from the upper left to lower right corner (for a 2D code). The first diagonal output starts with the bit row 1, column 1 followed by the diagonal starting at row 1, column 2. For 3D codes, instead of reading diagonally through the 2D array, interleaving reads diagonally through a cube of data.

The example below shows how interleaving is applied for a 2D (64,57)x(64,57) code.

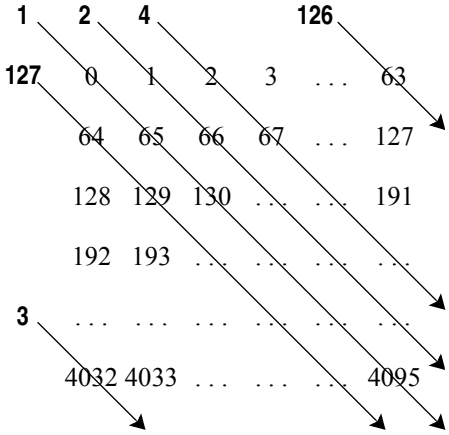
Figure 2: Input Block

0	1	2	3	. . .	63
64	65	66	67	. . .	127
128	129	130	. . .	. . .	191
192	193	. . .	. . .	. . .	. . .
. . .	. . .	. . .	. . .	. . .	. . .
4032	4033	. . .	. . .	. . .	4095

Note: The number reflects the bit order, including generated ECC bits.

The encoded, scrambled data output is taken along diagonal lines starting with bit 0 as shown below. The order of the interleaving is noted for each diagonal line.

Figure 3: 2D Helical Interleaving



For the (64,57)x(64,57) block, the data is: 0, 65, 130, ..., 4095, 1, 66, ..., 4031, 4032, 2, 67, ..., ..., 63, 64, ..., 4094 for a total of 4096 bits output. The decoder can automatically deinterleave the block to restore it to its original order.

Figure 4: Encoded/Interleaved Data Output

0	65	130	. . .	4030	4095
1	66	131	. . .	4031	4032
2	67	132	. . .	3968	4033
3	68	. . .	. . .	. . .	. . .
. . .	. . .	. . .	. . .	. . .	. . .
63	64	129	. . .	4029	4094

Data bits are output from the encoder in row order from left to right. 3D helical interleaving/ deinterleaving is done by reading/writing cells diagonally through the x, y, and z dimensions. Note

that the data rate drops when interleaving and/or deinterleaving.

4.6 SHORTENING

Shortening refers to removing a specific number of data symbols from an X, Y, or Z axis (*Xshort*, *Yshort*, and *Zshort* parameters), removing data symbols from the code block (*shortb*), or removing rows of data symbols from the code block (*shortr*). The purpose of applying shortening is to adjust a code size down to exactly match a particular application's required size and channel rate. The way shortening is implemented is that leading data symbols or rows of symbols are removed. The decoder knows the block size and shortening parameters and can insert zeroes into the code block in place of the shortened values for purposes of decoding. After the block is decoded these inserted zeroes are then removed before outputting the corrected data block. For more information on shortening TPCs, please refer to application note: ANTPC02.

5.0 AHA TPC SIMULATION SOFTWARE

The AHA TPC Simulation Software allows the user to model the performance of Turbo Product Codes (TPCs). The TPCs are fully configurable and support the Galaxy library of Turbo Product Codes. The software is designed to run under Windows® (95, 98, or NT4.0).

This software is the property of Comtech AHA Corporation (AHA) and is protected under the terms outlined in the software license agreement (see Help-About AHA TPC Simulation).

5.1 GRAPH/CONTROL WINDOW

In this window, the TPC Simulation software generates Eb/No vs. Bit Error Rate (BER) curves showing the performance of the code programmed in the Configuration/Currently Plotting window. All the settings (Codes, Iterations, Quantization, Feedback, etc.) for how the Galaxy core would perform are configured in the registers.

The simulation setup for Bit Error Rate (BER) curve generation is as follows:

1) Random data is input to a TPC encoder.

The binary output of the encoder is converted to channel symbols. For this simulation, a binary '1' is converted to a floating point 1.0 value, while a binary '0' is converted to a floating point -1.0 value.

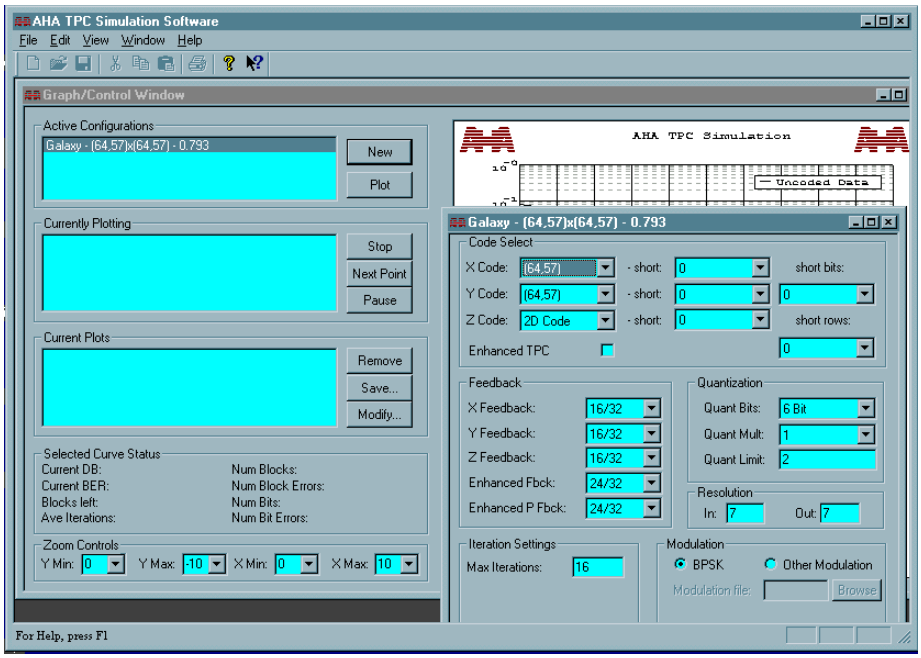
Additive White Gaussian Noise (AWGN) is added to the channel symbols, according to the Energy per Bit vs. Noise power (Eb/N0) setting chosen. This Eb/N0 value takes into account the code rate.

The received noisy channel data is quantized into soft decision values with the number of bits set in *quant_size* in the Register Configuration window.

These soft values are sent to the TPC decoder.

The output of the decoder is compared with the original data. All bit errors are counted. The BER is computed as bits in error out of decoder divided by total bits transmitted.

Figure 5: AHA TPC Simulation Software



5.2 ACTIVE CONFIGURATION GROUP

New

Available Configurations:

Galaxy Core Simulation
Run a new Turbo Product Code simulation using the Galaxy core library of codes. When selected, a Galaxy Configuration window pops up.

Load Data Point file
Load a previously generated data point file.

Plot

Curve Generation Settings Group:

Begin computing the Bit Error Rate statistics for the selected code and plot the results. The Plot Configuration Window pops up. The curve generation settings are programmed in the Plot configuration window.

Minimum Block Errors Per Point
Specify the minimum number of errors required for each plotted point in the Eb/No vs. BER graph. Typically, each point is based on approximately 50% more than this amount.

Eb/No Start
Specify a starting Eb/No value for the plot.

Eb/No Step
Specify the Eb/No step size in dB.

BER Limit to stop at
Specify how deep in Bit Error Rate the plot gets computed.

Graph Settings Group:

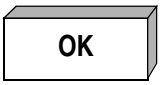
Key String
A comment field for the graph window.

Line Color
Allows the user to select the color for plotting.

Advanced Plot Controls

Advanced Plot Controls Window:

The Advanced Plot Controls are accessed by selecting the advanced button in the Plot Configuration Window allowing you to select Alternate Data Sources or define your own Noise Model. The Alternate Data Source can either be a binary data file that is continually cycled through (the file is rewound when the end is reached) or you can specify a DLL. The DLL feature allows you to generate your own data source to see how it performs with certain codes. The function in the DLL has a specific form and information on how to construct an appropriate DLL can be found with the Alternate Data Source sample code that was delivered with this release. The Alternate Noise Model can be specified as a DLL much like the Alternate Data Source. The function in the DLL has a specific form and information on how to construct an appropriate DLL can be found with the Alternate Noise Model sample code that was delivered with this release.



Once plot configurations are defined, begin plotting by pressing the "OK" button. This starts the plotting of a curve. A copy of the current configuration is created in a separate window so you can see the setting for the generated curve. Multiple curves can be generated at the same time by repeating this operation with the new configuration.

5.3 GALAXY CONFIGURATION WINDOW

This window opens when you select New → Galaxy Core Simulation → OK a new Turbo Product Code simulation . . . From the Graph/Control Window.

X code

Code for X axis of TPC array. Valid values are: extended hamming codes: (8,4), (16,11), (32,26), (64,57), (128, 120), (256, 247), simple parity codes: (4,3), (8,7), (16,15), (32,31), (64,63), (128,127), (256,255) and uncoded: (8,8), (16,16), (32,32), (64,64), (128, 128), (256, 256).

(X code) - short

Number of rows to shorten from the X axis. Shortening allows the user to exactly match any block size less than the maximum.

Y code

Code for Y axis of TPC array. Valid values are: extended hamming codes: (8,4), (16,11), (32,26), (64,57), (128, 120), (256, 247) and simple parity codes: (4,3), (8,7), (16,15), (32,31), (64,63), (128,127), (256,255).

(Y code) - short

Number of rows to shorten from the Y axis. Shortening allows the user to exactly match any block size less than the maximum.

short bits

Short bits Allows shortening of partial rows. Short bits refers to number of bits to be shortened.

Z code

Code for Z axis of TPC array. Valid values are: extended hamming codes: (8,4), (16,11), (32,26), (64,57), (128, 120), (256, 247), simple parity codes: (4,3), (8,7), (16,15), (32,31), (64,63), (128,127), (256,255) and uncoded: (8,8), (16,16), (32,32), (64,64), (128, 128), (256, 256).

(Z code) - short

Number of rows to shorten from the Z axis. Shortening allows the user to exactly match any block size less than the maximum. This option is only available for 3D codes.

short rows

Short rows allows shortening of partial planes by specifying the number of rows to be shortened.

Enhanced TPC

Enhanced TPC allows another dimension of coding in addition to X,Y,Z codes.

Feedback Group:

Feedback is used by Turbo Product Codes as a weighting function applied to the computed decode values during the iterative decoding.

After each axis iteration, the output of the Soft In Soft Out (SISO) Decoder is multiplied by the feedback constant for that axis. These values are then fed back into the SISO for future iterations.

X Feedback

Feedback multiplier for the X-axis. Range: 1/32 to 31/32. Recommended starting value is 16/32.

Y Feedback

Feedback multiplier for the Y-axis. Range: 1/32 to 31/32. Recommended starting value is 16/32.

Z Feedback

Feedback multiplier for the Z-axis. Range: 1/32 to 31/32. Recommended starting value is 16/32.

Enhanced Feedback

Feedback multiplier for the Enhanced mode. Range: 1/32 to 31/32. Recommended starting value is 24/32.

Enhanced Parity Feedback

Feedback multiplier for the Enhanced mode. Range: 1/32 to 31/32. Recommended starting value is 24/32.

Quantization Group:

Quant Bits

Input quantization size. Specifies the number of bits for each symbol input to the TPC core. Range: 1 to 12 bits. Use 1 bit for hard decision input data. The default is 6 bits.

Quant Mult

The Quantization Multiplier is a shift applied to the input values to put them near the center of the quantization range. If you have 3 or more Quant Bits, leave Quant Mult set to the default value of 1.

Quant Limit

Quantization limit is a clipping factor applied in the demodulator model to scale the decode input data.

Resolution Group:

In

The number of soft decision bits for each decode input symbol. Set In = QuantBits + 1. For hard decision input data set In to 4. Defaults to 7.

Out

The number of soft decision bits for each decode output symbol. Set Out = In + 1. Defaults to 7.

Iteration Settings Group:

Max Iterations

Maximum Iterations is the number of iterations used by the decoder.
Range: 1 to 256.

Modulation Group:

The Galaxy Simulation software supports built in BPSK modulation as well as user defined modulation.

BPSK

Selecting BPSK enables BPSK modulation of the encoded data.

Other Modulation

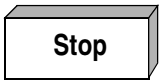
Allows user defined modulation. The modulation is contained in a file.

Modulation File

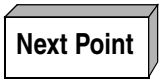
User provided modulation file. Example modulations for QAM16, QAM64, and 8 PSK are included with the evaluation software (see modulation files: qam16.mod, qam64.mod, and psk8.mod).

5.4 CURRENTLY PLOTTING GROUP

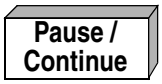
The code currently being calculated and plotted will be listed in the Currently Plotting message window. Whenever a new data-point is created the graph is updated. If you wish to stop the current plot then you can press the "Stop" button. If you are satisfied with the current point that is being generated you can press the Next Point button to move to the next Eb/No point. Every plot that is generated is added to the key in the upper right corner of the graph window and is designated by the code that it represents. More than one curve can be generated at a time by selecting an active configuration and hitting the plot button.



Stop calculating data points for the present curve.

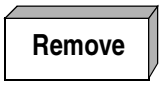


Next Point button allows the user to stop calculating errors at the present Eb/No value and move on to the next Eb/No value.

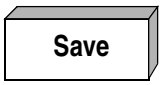


Pause/Continue button allows the user to stop the current plot and then resume.

5.5 CURRENT PLOTS GROUP



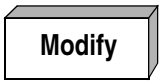
The Remove button allows the user to remove the highlighted plot and label from the plotted graph.



The Save button allows the user to save the highlighted plot data points to a user defined file. This function will save to a data file the data points from the graph window. The format for a data file is one x y value per line. For example,

0	0.127012
0.25	0.0907417
0.5	0.084484
0.75	0.0723518
1	0.0397889
1.25	0.0214485
1.5	0.0131982

Data files for comparison with other codes are available from Comtech AHA Corporation upon request.



You can modify the color of a plot by pressing the "Modify" button. This allows the user to modify the plot label and color.

5.6 SELECTED CURVE STATUS GROUP

The following curve status information is displayed for the currently plotting curve.

Current DB:

Current DB refers to the current Eb/No value that is being simulated.

Current BER:

Current BER refers to the approximate Bit Error Rate for point being calculated.

Blocks Left:

Blocks left is an estimate of the number of blocks left to complete the simulation for the current point.

Ave Iterations:

Average Iterations is the average number of decode iterations required per block for the current point.

Num Blocks:

Number of Blocks is the number of blocks that have been modeled for the current point.

Num Block Errors:

Number of Block Errors is the number of blocks that had decode errors for the current point.

Num Bits:

Number of Bits is the number of bits that have been decoded for the current point.

Num Bit Errors:

Number of Bit Errors is the number of bits that have been decoded incorrectly for the current point.

5.7 ZOOM CONTROLS GROUP

You can zoom in and out on the current graphs by selecting different X-axis and Y-axis values.

Y Min

Range: 0 to -30

Y Max

Range: 0 to -30

X Min

Range: -3 to 25

X Max

Range: -3 to 25

5.8 EXTERNAL DATA AND NOISE SOURCE DLL

The AHA TPC Simulation software has the ability to use functions found in external DLLs for generating noise and data. In the AHA TPC Simulation alternate noise and data samples directory you will find an example of each of these options.

There are two projects built with MSVC++ 6.0 in the included directory.

- datagenerator.dsw
- noisegenerator.dsw

Each of these projects builds a sample DLL from the source code located in the associated *.cpp file. Information about what parameters are passed to each of these functions can also be found in the *.cpp files.

The projects build the DLLs and place them in either the Debug or Release directories depending on which build method you chose. To use these DLLs in the AHA TPC Simulation Windows eval software, select the "Advanced" button on the "Plot Configuration" window. Please refer to the online help for more information.

6.0 AHA GALAXY API PROGRAMMING

Galaxy library functions are provided for purposes of evaluating the Turbo Product Codes supported by the Galaxy licensed cores by allowing the user to integrate the functions into either a Matlab system model or a "C" program system model.

6.1 PROGRAMMING PARAMETERS DEFINED

Parameters that are used in both APIs are defined here.

6.1.1 XCODE, YCODE AND ZCODE PARAMETERS

Code[5:0] selects a code for each axis. This section applies to *Xcode*, *Ycode*, and *Zcode* parameters in both of the APIs.

bit5	*bit4*	*bit3*	*bit2*	*bit1*	*bit0*
UNCODED AXIS	PARITY	code[3]	code[2]	code[1]	code[0]

Table 3: Constituent Codes (different from AHA4501)

code[3:0]	*EXT HAMMING (PARITY=0)*	*PARITY ONLY (PARITY=1)*
0x02	na	4,3
0x03	8,4	8,7
0x04	16,11	16,15
0x05	32,26	32,31
0x06	64,57	64,63
0x07	128,120	128,127
0x08	256,247	256,255

PARITY - Set to 0 for extended Hamming and 1 for parity only.

UNCODED AXIS -Set to 1 for uncoded axis, 0 for coded axis.

6.1.2 GALAXY CONTROL PARAMETERS

xfbk feedback value for x axis, xfbk = feedback_value * 32

yfbk feedback value for y axix, yfbk = feedback_value * 32

zfbk feedback value for z axis, zfbk = feedback_value * 32

hfbk feedback value for hyper axis, xfbk = feedback_value * 32

hfpbk feedback value for hyper parity axis, xfbk = feedback_value * 32.

quant_size number of soft input bits for the decoder (1 to 6 bits)

quant_mult sets the shift value (must be odd).

inres number of input bits

outres number of output bits

*heli_interleave*1 = enable helical interleaving

numIter maximum number of iterations (1 to 256)

Xcode a 6-bit value to put in the register that will set the X Axis code. See Table 1, Galaxy Constituent Codes, (i.e. 16x11 == 4).

Ycode a 6-bit value to put in the register that will set the Y Axis code. See Table 1, Galaxy Constituent Codes, (i.e. 16x11 == 4).

Zcode a 6-bit value to put in the register that will set the Z Axis code. SeeTable 1, Galaxy Constituent Codes, (i.e. 16x11 == 4).

Xshort number of bits to shorten in x direction

Yshort number of bits to shorten in y direction

Zshort number of bits to shorten in z direction

shortb bit shortening for partial rows

shortr row shortening for partial plane

enhanced 1 = enable enhanced mode

*encode_flag*matlabgalaxyapi specific parameter, 1 = enable encoding, 0 = decoding.

inbits matlabgalaxyapi specific parameter, an array containing the input data bits to be either encoded or decoded.

evm galaxyapi.dll specific parameter, GalaxyStruct structure returned by GalaxyInit() function.

6.2 AHA GALAXY MATLAB SIMULATION API

Matlabgalaxyapi is an API containing one function, matlabgalaxyapi(). The function performs either encoding or decoding depending on whether the encode_flag is set to one or zero. A path to these files may be specified from the Matlab command line prompt or within a Matlab script file. If the path where the toolkit exists is:

c:\AHA\Galaxy Simulation Toolkit\Matlab Toolkit, then use the following command line script:

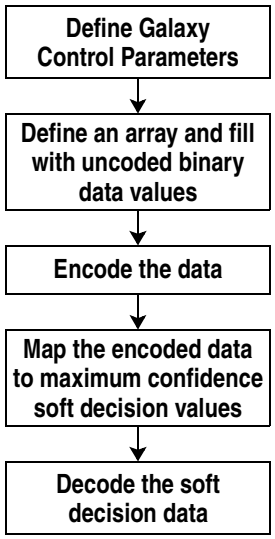
cd ([‘c:\AHA\Galaxy Simulation Toolkit\Matlab Toolkit ‘]).

To run the example script from the Matlab command line prompt type: galaxy_matlab_sim.

Galaxy_matlab_sim.m is an example Matlab script file that encodes and decodes a block of data. The Galaxy Windows evaluation software should be run first to determine the optimum settings for the Turbo Product Code parameters. Call the API function by defining Galaxy control parameters then make the following function call from a Matlab script file or Matlab command line prompt.

```
Array_bits = matlabgalaxyapi(encode_flag,... numIter,...
                                heli_interleave,...
                                xcode,...
                                ycode,...
                                zcode,...
                                xshort,...
                                yshort,...
                                zshort,...
                                shortb,...
                                shortr,...
                                enhanced,...
                                xfbk,...
                                yfbk,...
                                zfbk,...
                                hfbk,...
                                hpfbk,...
                                quant_size,...
                                quant_mult,...
                                inres,...
                                outres,...
                                inbits);
```

6.2.1 FLOW CHART, MATLAB EXAMPLE PROGRAM



6.3 AHA GALAXY C SIMULATION API

Galaxyapitest.c is an example source file. It is best to run the GALAXYAPITEST demo from a DOS window with output rerouted to a file by using the command line:

galaxyapitest > out.txt, and then display the output file.

6.3.1 GALAXY API FUNCTIONS

GalaxyInit();
/*
Initialize the evaluation software for TPC encoding or decoding operation and returns a pointer to a GalaxyStruct structure that is needed in all the other API calls.
*/

GalaxyGetLastError();
/*
Returns a char pointer to a string giving the last error produced.
*/

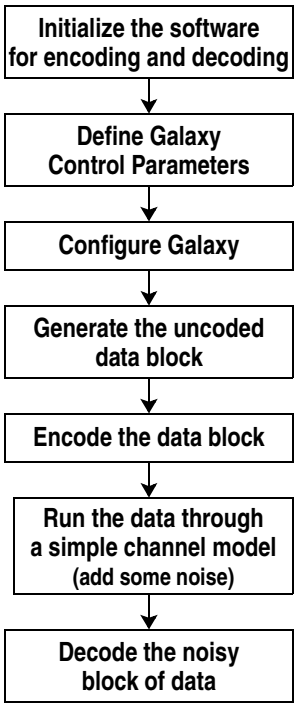
GalaxyConfigure(evm, xcode, ycode, zcode, enhanced, xshort, yshort, zshort, shortb, shortr, xfbk, yfbk, zfbk, hfbk, hpfbk, numIter, heli_interleave, inres, outres, quant_size, quant_mult);
/*
Configure Galaxy with the Galaxy control parameters.
parameters: see Section 6.1
*/

GalaxyDumpSettings(evm);
/*
Read current settings for the parameters set by GalaxyConfigure and send to stdout.
*/

GalaxyEncodeDataBits(evm,rawData,encodedBlock);
/*
Encode the raw data using software
parameters:
evm The GalaxyStruct structure returned by GalaxyInit()
rawData The Raw Data, an integer array of bits (1 bit per array location).

```
        encodedBlock        The Encoded Data, an integer array of bits (1 bit per array location).
    */

GalaxyDecodeEncodedBlock(evm,afterChannelBlock,decodedBlock);
    /*
    Decode the encoded block using the Galaxy software.

    parameters:
        evm                 The GalaxyStruct structure returned by GalaxyInit().
        afterChannelBlock   The Noisy Data, an integer array of soft values (1 value per array location).
        decodedBlock        The Decoded Data, an integer array of bits (1 bit per array location)
    */

GalaxyFreeResources(evm);
    /*
    Free memory resources.

    parameters:
        evm                 The GalaxyStruct structure returned by GalaxyInit()
    */

GalaxyGetDataSize(evm);
    /*
    Returns the number of data bits in the current configuration (shortening is taken into account)

    parameters:
        evm                 The GalaxyStruct structure returned by GalaxyInit()
    */

GalaxyGetBlockSize(evm);
    /*
    Returns the number of data plus ECC bits in the current configuration (shortening is taken into account).

    parameters:
        evm                 The GalaxyStruct structure returned by GalaxyInit()
    */

GalaxyCopyDataBits(evm,eccBlock,dataBlock);
    /*
    Strip data bits from the eccBlock and place in the dataBlock.

    parameters:
        evm                 The GalaxyStruct structure returned by GalaxyInit().
        eccBlock            The block with ECC bits included.
        dataBlock           The block with OUT ECC bits.
    */
```

6.3.2 FLOW CHART, GALAXY API EXAMPLE PROGRAM



6.4 INSTALLATION TIPS AND POSSIBLE PROBLEMS

1. DLL file not in current directory or Windows\System directory.
Program exits gracefully with a Windows Error message.

7.0 RELATED PUBLICATIONS

DOCUMENT #	*DESCRIPTION*
PB4501	AHA Product Brief – AHA4501 Astro 36 Mbits/Sec Turbo Product Code Encoder/Decoder
PB4501EVM	AHA Product Brief – AHA4501 TPC EVM ISA Evaluation Module
PB4501EVSW	AHA Product Brief – AHA4501 TPC Windows Evaluation Software
PBGALAXY	AHA Product Brief – Galaxy Core Generator Turbo Product Code Decoder Cores
PBGALAXY_EVSW	AHA Product Brief – AHA Galaxy TPC Windows Evaluation Software
PBGALAXY_STK	AHA Product Brief – AHA Galaxy Simulation Tool Kit
PSGALAXY_IG	AHA Product Specification – AHA Galaxy Turbo Product Code Cores Integrator's Guide
PS4501	AHA Product Specification – AHA4501 Astro 36 Mbits/Sec Turbo Product Code Encoder/Decoder
ANTPC01	AHA Application Note – Primer: Turbo Product Codes
ANTPC02	AHA Application Note – Use and Performance of Shortened Codes with the AHA4501 TPC Encoder/Decoder)
ANTPC03	AHA Application Note – Turbo Product Code Encoder/Decoder with Quadrature Amplitude Modulation (QAM)
ANTPC04	AHA Application Note – Use and Performance of the AHA4501 TPC Encoder/Decoder with Differential Phase Shift Keying (DPSK)

DOCUMENT #	DESCRIPTION
PB4501	AHA Product Brief – AHA4501 Astro 36 Mbits/Sec Turbo Product Code Encoder/Decoder
PB4501EVM	AHA Product Brief – AHA4501 TPC EVM ISA Evaluation Module
PB4501EVSW	AHA Product Brief – AHA4501 TPC Windows Evaluation Software
ANTPC05	AHA Application Note – AHA4501 Turbo Product Code Encoder/ Decoder Designers Guide
ANTPC06	AHA Application Note – AHA4501 Turbo Product Code Encoder/ Decoder Frequently Asked Questions (FAQ)
ANTPC07	AHA Application Note – Turbo Product Codes for LMDS
TPCEVAL	AHA Evaluation Software – Turbo Product Codes - Windows Evaluation Software

APPENDIX A: AHA SOFTWARE LICENSE

COMTECH AHA CORPORATION LICENSE AGREEMENT

IMPORTANT READ CAREFULLY: This Comtech AHA Corporation License Agreement (hereinafter "AGREEMENT") is legally binding agreement between YOU (either as an individual or an entity) and Comtech AHA Corporation., its parents, successors, subsidiaries, suppliers and/or licensors (collectively referred to hereinafter as "AHA").

THIS LICENSE applies to a copy of the AHA TPC Simulation Software (hereinafter "SOFTWARE"). By the process of installing, copying or otherwise using this SOFTWARE, YOU hereby agree to be bound by the terms and conditions of this Agreement. If YOU do not agree to all of the terms and conditions of this Agreement, return this software to AHA with written notice stating that you do not accept THIS LICENSE.

YOU ARE ADVISED THAT ANY THIRD PARTY SOFTWARE PROVIDED HEREWITH, INCLUDING, BUT NOT LIMITED TO ANY GNU OR non-AHA ROUTINE, ALGORITHM, CODE, OBJECT, DATA FILE OR OTHERWISE WHICH MAY BE PROVIDED HEREWITH IS INCLUDED SOLELY FOR USE AT YOUR OPTION FOR PURPOSES OF DEMONSTRATION AND EVALUATION OF THIS SOFTWARE. AHA MAKES NO REPRESENTATIONS AS TO THE INTEGRITY, ACCURACY, VALIDITY OR RELIABILITY OF ANY SUCH GNU OR non-AHA ROUTINE, ALGORITHM, OBJECT, CODE, DATA FILE OR OTHERWISE AND SHALL NOT BE RESPONSIBLE FOR ANY LOSSES OR DAMAGES WHICH MAY BE SUSTAINED OR WHICH MAY OCCUR AS A RESULT OF THE INSTALLATION, COPYING OR USE OF THIS SOFTWARE AND/OR ANY THIRD PARTY GNU OR non-AHA ROUTINE, ALGORITHM, OBJECT, CODE, OR DATA FILE.

YOU ARE FURTHER ADVISED THAT THE SOFTWARE YOU ARE INSTALLING, COPYING OR USING MAY CONTAIN LESS FEATURES, FORMS OR FUNCTIONS THAN ANY COMMERCIALLY AVAILABLE VERSION OF THIS SOFTWARE OR COMMERCIALLY RELEASED PRODUCT INCORPORATING THIS SOFTWARE AND AHA MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, AS TO THE ACCURACY, VALIDITY, OR RELIABILITY OF THIS SOFTWARE.

This AGREEMENT is a license agreement and is not intended to operate as an agreement for sale or transfer of any rights in the SOFTWARE, except as expressly provided herein. AHA continues to own the SOFTWARE, including any copies thereof, and solely grants to YOU a non-exclusive right to use this SOFTWARE for demonstration and evaluation purposes only. AHA EXPRESSLY RETAINS ANY AND ALL RIGHTS NOT EXPRESSLY GRANTED HEREUNDER THIS AGREEMENT AND NOTHING IN THIS AGREEMENT SHALL CONSTITUTE A WAIVER OF AHA'S RIGHTS UNDER U.S. OR INTERNATIONAL PATENT, TRADEMARK, OR COPYRIGHT LAW OR ANY OTHER FEDERAL OR STATE LAW.

Grant of License:

Subject to the provisions contained in this AGREEMENT, AHA hereby grants YOU a non-exclusive and non-transferable license to install and use this SOFTWARE for demonstration and evaluation purposes only. YOU MAY NOT COPY THIS SOFTWARE IN ANY MANNER OR FORM, EXCEPT FOR A SINGLE COPY FOR ARCHIVAL PURPOSES PROVIDED SUCH COPY CONTAINS ALL OF THE SOFTWARE'S PROPRIETARY NOTICES, INCLUDING A COPY OF THIS AGREEMENT.

Restrictions:

YOU may not modify, translate, change, decompile, reverse engineer, disassemble, create derivative works or otherwise alter this SOFTWARE (except to the extent expressly authorized by law) without the express written consent of AHA. ANY AND ALL SUCH MODIFICATIONS, CHANGES, ALTERATIONS OR DERIVATIVE WORKS SHALL REMAIN THE SOLE AND EXCLUSIVE PROPERTY OF AHA TO THE MAXIMUM EXTENT UNLESS OTHERWISE PROHIBITED BY LAW.

YOU may not copy this SOFTWARE, or any supporting documentation accompanying this SOFTWARE (except for archival purposes as previously set forth herein) without the express written consent of AHA.

YOU may not rent, lease, assign or otherwise transfer any rights to this SOFTWARE granted hereunder this AGREEMENT without the express written consent of AHA.

YOU may not alter, remove, or destroy any proprietary mark, trademark, patent or copyright markings, notices or labels placed upon or contained within the SOFTWARE or any supporting documentation which may accompany the SOFTWARE.

Support:

YOU shall not be entitled under the terms of this AGREEMENT to receive any support services of any kind whatsoever.

Disclaimer of Warranty:

THIS SOFTWARE AND ANY SUPPORTING DOCUMENTATION PROVIDED HEREWITH ARE PROVIDED FOR DEMONSTRATION AND EVALUATION PURPOSES ON AN "AS-IS" BASIS, WITH NO WARRANTIES OF ANY KIND, EXPRESS, IMPLIED OR OTHERWISE INCLUDING, BUT NOT LIMITED TO, ANY WARRANTY OF MERCHANTABILITY, IMPLIED WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE, OR ANY WARRANTY OF NONINFRINGEMENT, WHETHER ALLEGED TO ARISE BY OPERATION OF LAW, BY REASON OF CUSTOM OR USAGE OF TRADE OR BY COURSE OF DEALING.

Limitation of Liability:

BY ACCEPTING THIS AGREEMENT, AND THE TERMS HEREUNDER, YOU SPECIFICALLY ACKNOWLEDGE THAT THE ENTIRE RISK ARISING OUT OF ANY INSTALLATION, COPYING OR USE OF THIS SOFTWARE IS SPECIFICALLY AND SOLELY ALLOCATED TO AND REMAINS WITH YOU. TO THE MAXIMUM EXTENT PERMITTED UNDER THE LAW, AHA SHALL NOT BE LIABLE FOR ANY CONSEQUENTIAL, INCIDENTAL, DIRECT, INDIRECT, SPECIAL, PUNITIVE OR OTHER DAMAGES WHATSOEVER, INCLUDING WITHOUT LIMITATION, ANY DAMAGES FOR LOST BUSINESS OR PROFITS OR OTHER PECUNIARY LOSS ARISING OUT OF THIS AGREEMENT AND/OR THE INSTALLATION, COPYING OR USE OF THE SOFTWARE.

Termination:

This AGREEMENT shall automatically terminate in the event YOU breach any of the terms or conditions of this AGREEMENT. Upon the expiration or termination of this AGREEMENT for any reason, all licenses granted hereunder shall terminate and YOU shall destroy all copies of the SOFTWARE.

General Provisions:

In the event that any provision of this AGREEMENT shall be held to be illegal, unenforceable or in conflict with any law of federal, state or local government having jurisdiction over this AGREEMENT, the validity and enforceability of all other remaining provisions shall not be affected thereby.

This AGREEMENT constitutes the full and complete understanding and obligations between YOU and AHA and is complete in and of itself. This AGREEMENT fully supersedes any and all prior understandings or agreements pertaining hereto and is not subject to any other terms or conditions not clearly set forth herein. This AGREEMENT may not be modified or altered, except by a written instrument signed by YOU and an authorized representative of AHA.

This AGREEMENT shall be governed by and determined in accordance with the laws of the State of Washington, excluding its conflicts of laws provisions and excluding the 1980 United Nations Convention on Contracts for the International Sale of Goods. Both parties agree that any dispute pursuant to this AGREEMENT shall be submitted to binding arbitration under the rules of the American Arbitration Association.