UNCLASSIFIED

NDIA Presentation

Proximity Fuze Branch

John E. Langan

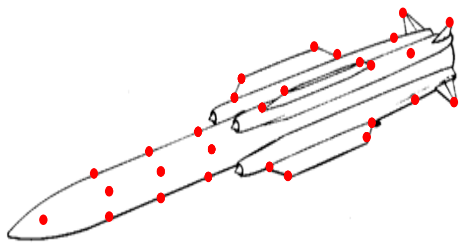
Code 478600D

China Lake, CA

john.langan@navy.mil

(760-939-3726)

Proximity Fuze Simulation with Embedded Tactical Software



UNCLASSIFIED

GenSim Fuze Simulation (1995 – present)

Proximity Fuze Branch

- GenSim runs missile endgame scenarios and outputs data in many formats. (Radar Proximity Fuze Simulation written in MS Visual C++). It is primarily written in "C".
- GenSim utilizes actual radar patterns / gains and implements Npoint target modeling and simulated radar clutter modeling.
- GenSim actually moves a missile reference and target reference along vectors toward Point-of-Closest-Approach (PCA) in its calculations. (This is called Time-Based processing).
- GenSim presently has about thirty target models and variations of target models. It has missile AAW targets, surface targets, and slow targets. It contains a low altitude clutter model.

Missile Proximity Fuzing

- Missile proximity fuzing is implemented in the last moments of missile flight as the missile and target converge to Point-Of-Closest-Approach (PCA). Proximity Fuzing is about detecting the target and timing the bursting of the warhead to optimize warhead fragment placement on the target.

- The design of missile proximity fuzes (Target Detecting Devices (TDDs)) requires analysis tools that simulate the fuzing system’s operation and measures of effectiveness for the TDD as well as the missile itself.

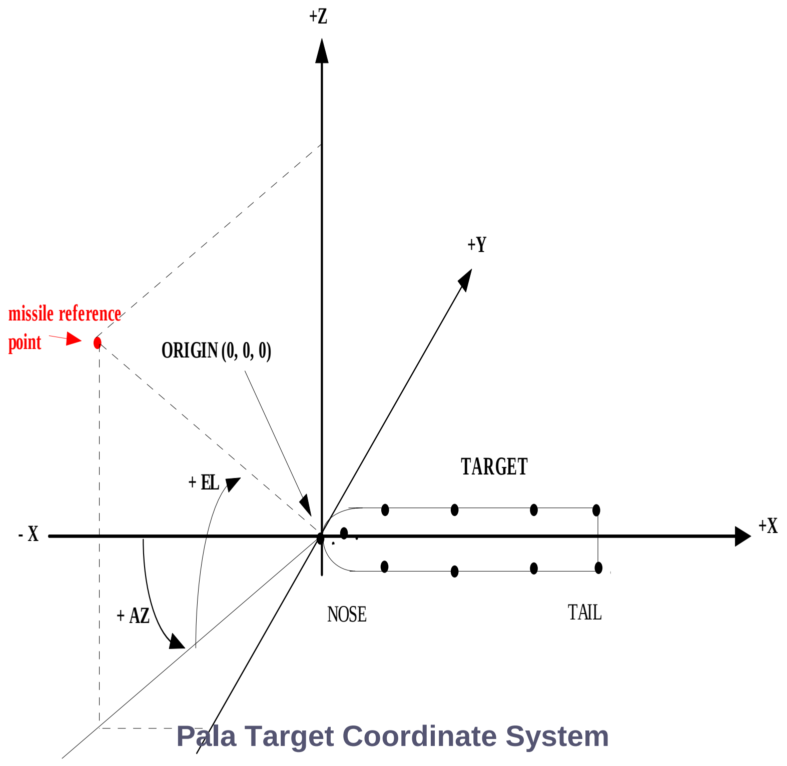
Legacy Fuze Simulations (1970 –1980's)

Proximity Fuze Branch

- Simple Geometric models: Event based, this means that the encounter did not actually move but detection was calculated geometrically.
- Slow: The computers these were run on were mainframes or mini-computers and took a long time to run encounter scenarios. Administration issues.
- Target and Clutter models were geometric models (or utilized tables of data, not actual sensor models)

Missile and Target Coordinates

GenSim fuze Detection is done in a Missile-Body Coordinate System. The fuze detection point is defined by a spherical coordinate R, α, ϕ

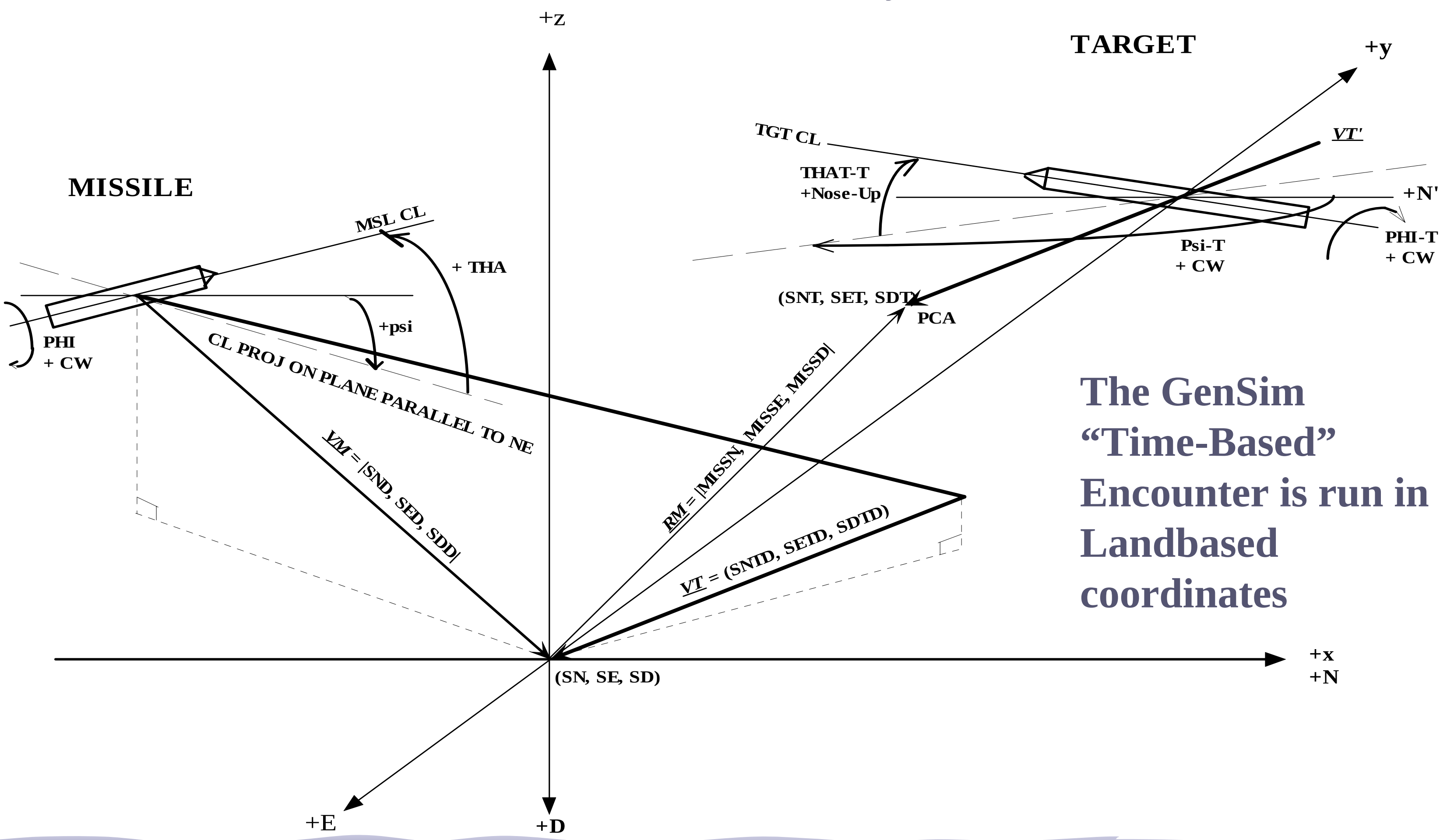


Pala Target Coordinate System

The Npoint Target is loaded into GenSim in Pala Coords.

Inertial Coordinates (Landbased)

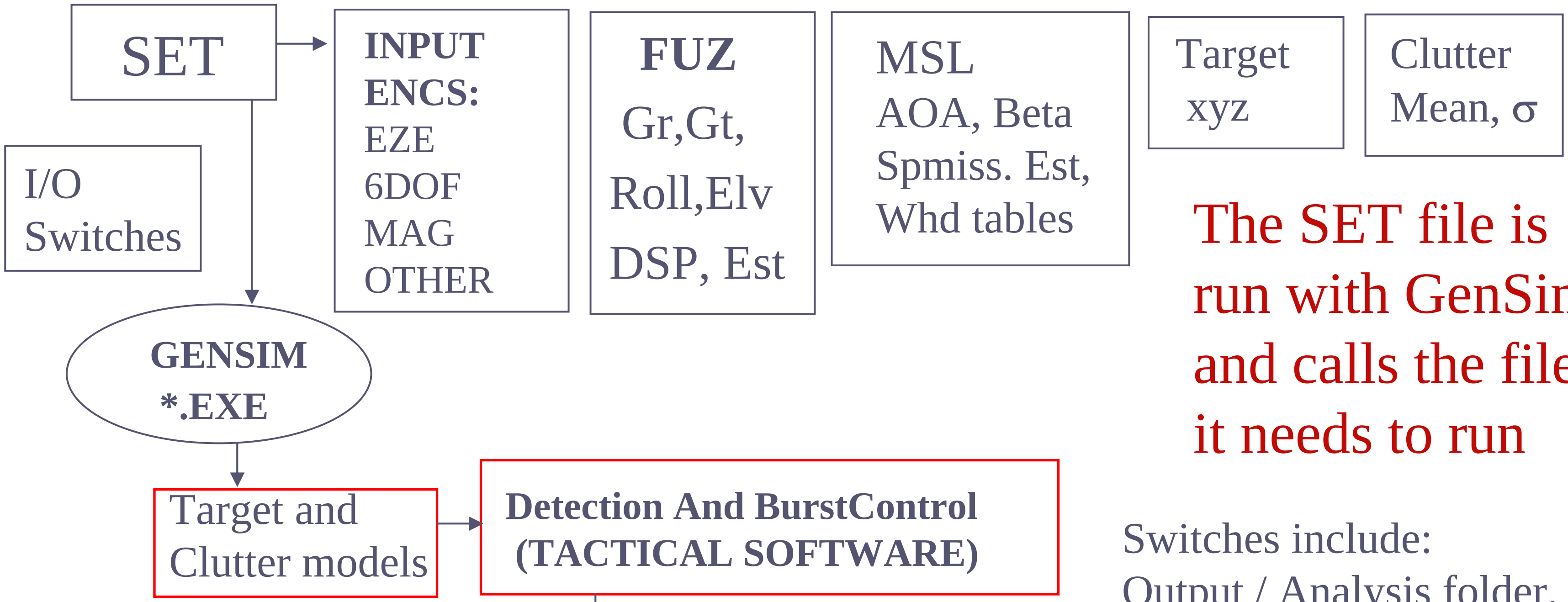
Proximity Fuze Branch



The GenSim "Time-Based" Encounter is run in Landbased coordinates

GenSim Input / Output Files

Proximity Fuze Branch


SET
INPUT ENCS: EZE 6DOF MAG OTHER
FUZ Gr,Gt, Roll,Elv DSP, Est
MSL AOA, Beta Spmiss. Est, Whd tables
Target xyz
Clutter Mean, σ
I/O Switches
GENSIM *.EXE
Target and Clutter models
Detection And BurstControl (TACTICAL SOFTWARE)

The SET file is run with GenSim and calls the files it needs to run

The output files include: Detect, No Detect, Guidance Reject, DR/XR, VM/X, Graph, Lethal Burst Interval, Banana, Mesa, Warhead Enc Format

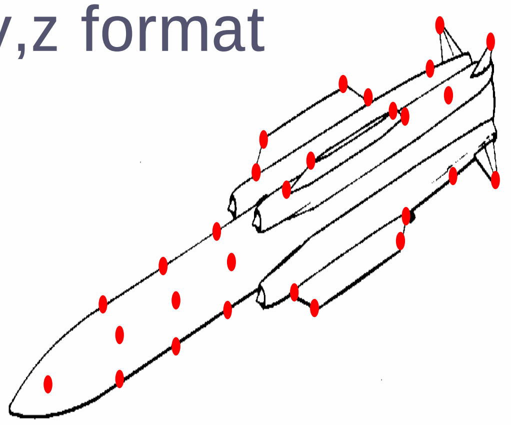
Switches include:
Output / Analysis folder.
Repeat Encounter, Clutter Rejection, Mirror,
Resp Est,
Detection Burst Cntl

Npoint Target Models

Proximity Fuze Branch

- An N-point model accounts for target RCS as well as radar characteristics, The "N" is the number of radar reflector points. N-point modeling is based on the theory that radar data tends to pool in specific areas on the target.
- Locations are specified relative to target nose (*.xyz file).
- The xyz file contains "N" points in x,y,z format
- The RCS specified in angle increments.
  - Az: -180 to +180.
  - El: -90 to +90.

At present there are over thirty N-point Targets developed for GenSim including Missiles, aircraft, slow targets and surface targets.



Clutter Modeling

- The GenSim Clutter Model loads an input file that contains the (Mean, σ) for Clutter Radar Cross Section (RCS) tabulated for various conditions and incidence angle of the beam.

- GenSim uses encounter geometry in looking up the (Mean, σ) value.

- The Threshold is calculated as:

$$RCS = Mean + K * \sigma + Offset:$$

where: Mean, σ are lookup table values.

Mean, σ, and Offset are in dBSm.

K and Offset are the clutter sigma multiplication factor.

Early GenSim / Fuze Algorithms

Proximity Fuze Branch

- GenSim was designed to do analysis “trade studies” for missile proximity fuze development.
- Early GenSim contained its own fuze detection and Burst Control. Burst Control contains time-delay algorithms.
- This simulation was used to make fuze design decisions in sensory development, signal processing complexity, and missile / fuze interface limits based on missile encounter conditions (to name a few).
- With the burst control software (guidance / fuzing / warhead combined effectiveness “PK” could be estimated).

Early GenSim / Common Header

- In early GenSim planning, it was intended that this simulation be implemented together with actual missile fuze tactical software to aid in improved tactical software development as well as provide “accurate” fuze effectiveness under varying missile endgame scenarios. GenSim would create the missile / target environment and would call the tactical software.

- To prepare for tactical software implementation a common header file “*.h” was created where both GenSim could place program definitions as well as the tactical software. This created a “common placeholder” where GenSim could pass and receive info from tactical S/W. Detection and BurstControl functions were defined with the prefix: “Common_” to prepare for Tactical implementation. The function“Common_DetTdLogic” contained GenSim detection and BurstControl algorithms.

GenSim, Early Tactical S/W

- For the first Tactical S/W interface, the “Common_DetTdLogic” function was replaced with a Tactical Interface (TI) function called “TI_TerminalExecutive( )”. This file contains (GenSim/ Tactical) interface (TI) files.

- Other Tactical file definitions: “TI” was tactical interface,”LM” lightly modified tactical files, “SAL” were “Simulation Abstraction Layer” as opposed to the tactical “HAL” Hardware Abstraction Layer.

- The GenSim Side had to perform a lot of Tactical S/W initialization in the first implementation since the tactical software was not operating as it was designed. The tactical software was designed to run once. GenSim with embedded tactical must run multiple scenarios.

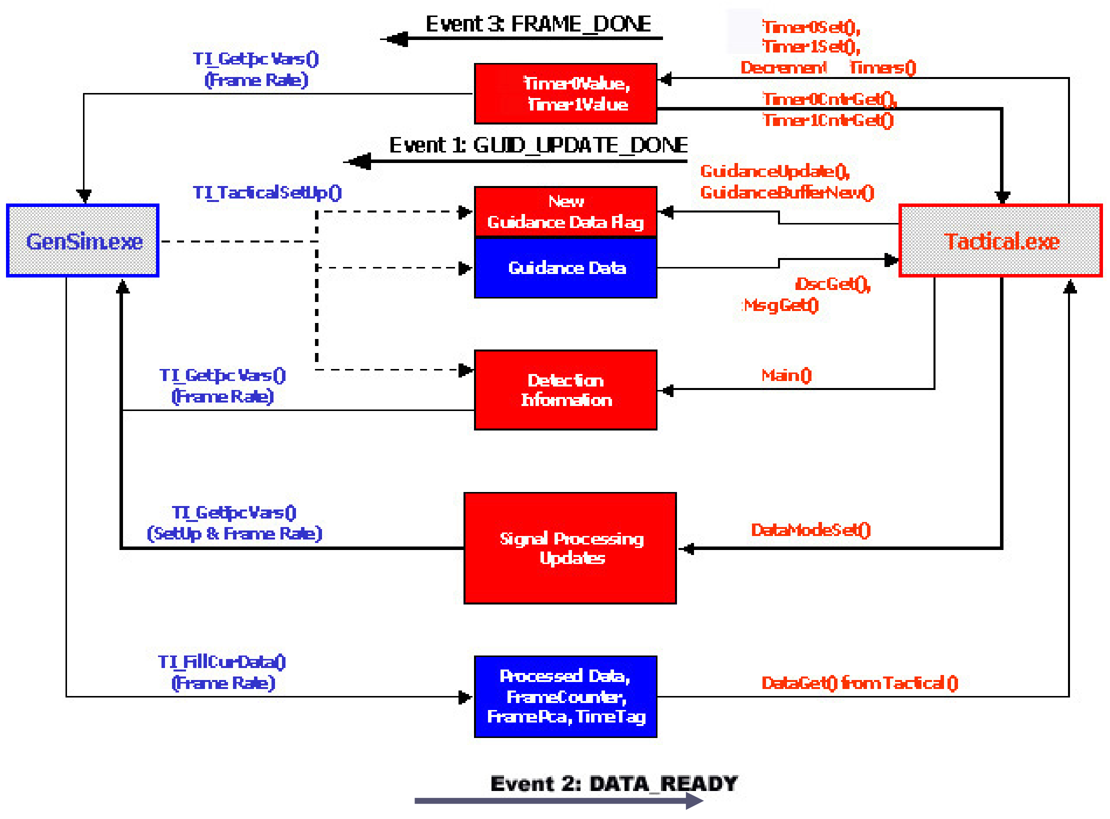
Early Tactical S/W (continued)

- GenSim is setup to run only the endgame portion of the encounter (last tenth of a second or so). The Tactical software is written to handle missile flight from intercept arm (last half second or so). GenSim has to properly handle the Tactical software for this part of the flight.
- GenSim would run the encounter and pass information to the Tactical S/W every frame while doing this pre-encounter initialization.
- In this early development our understanding of the Tactical S/W was poor and therefore our initialization methods were crude. The Tactical software was designed with microprocessors in mind and our embedded tactical simulation was not. The Dynamic-Linked-Library concept changed that.

GenSim Evolution (DLL’s)

- In the Dynamic-Linked-Library approach, the Tactical Software becomes an Executable (*.exe) called by the GenSim Executable program. The Simulation becomes multiple nested executable programs that pass information through a mailbox (DLL).
- GenSim initializations can be done on the GenSim side, Tactical initializations can be done on the Tactical side and pertinent information can be passed between the processes through the DLL. The DLL approach simplified Tactical “drop-in” to GenSim.
- Before the DLL was utilized, GenSim would have to be run once for each processor in the system. (working with each processor independently). With the DLL concept, each processor is now an executable called by the GenSim executable each frame.

GenSim Evolution (DLL's)

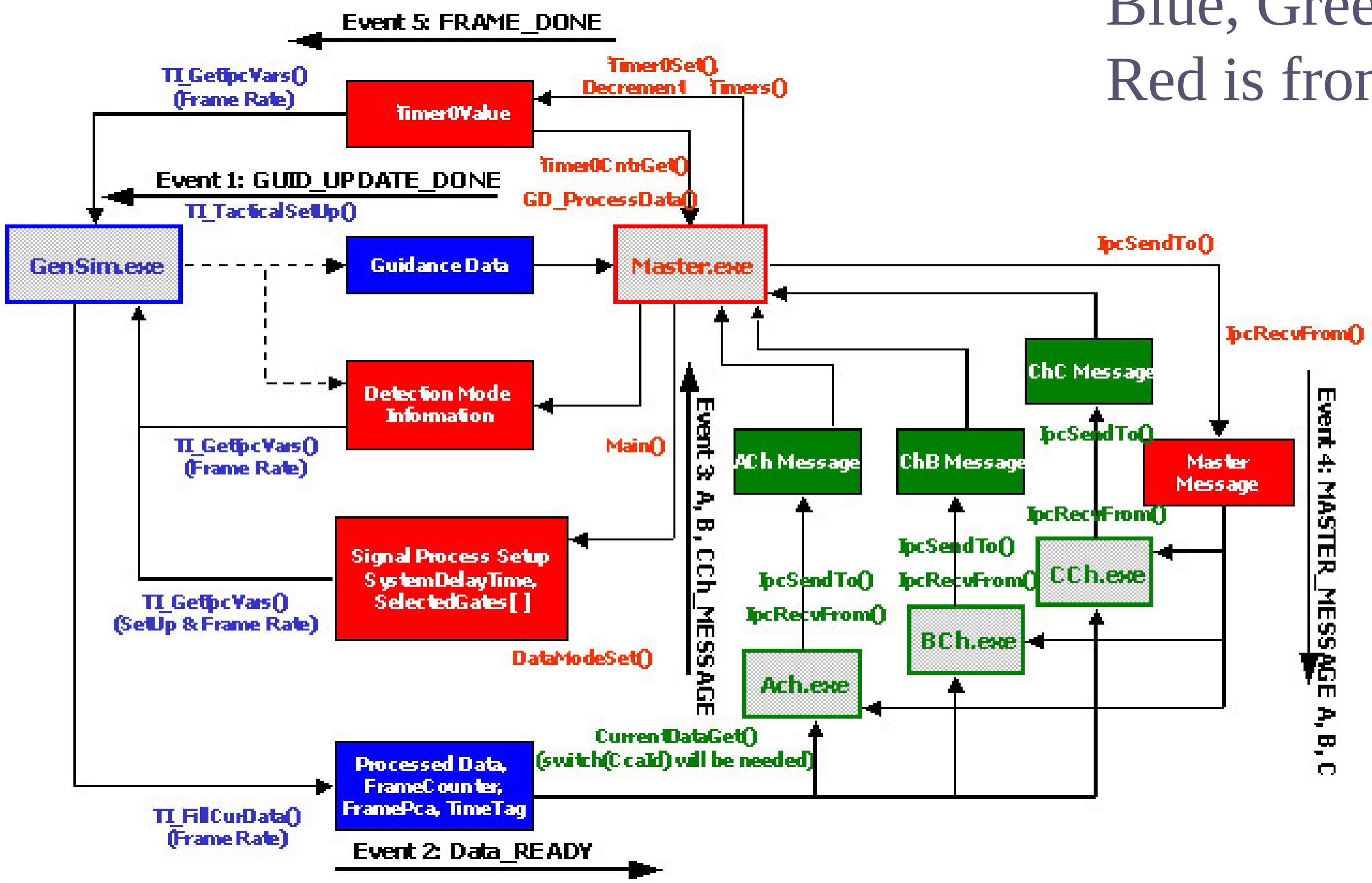

Event 3: FRAME_DONE
Timer0Set(), Timer1Set(), Decrement Timers()
TI_GetIpcVars() (Frame Rate)
Timer0Value, Timer1Value
Timer0CntrGet(), Timer1CntrGet()
Event 1: GUID_UPDATE_DONE
GuidanceUpdate(), GuidanceBufferNew()
TI_TacticalSetUp()
New Guidance Data Flag
GenSim.exe
Guidance Data
Tactical.exe
DscGet(), MsgGet()
TI_GetIpcVars() (Frame Rate)
Detection Information
Main()
TI_GetIpcVars() (SetUp & Frame Rate)
Signal Processing Updates
DataModeSet()
TI_FillCurData() (Frame Rate)
Processed Data, FrameCounter, FramePca, TimeTag
DataGet() from Tactical()
Event 2: DATA_READY

This figure shows a single processor, single executable DLL approach.

Red and Blue denote events where the DLL passes information between processors.

GenSim Evolution (DLL's)

Proximity Fuze Branch



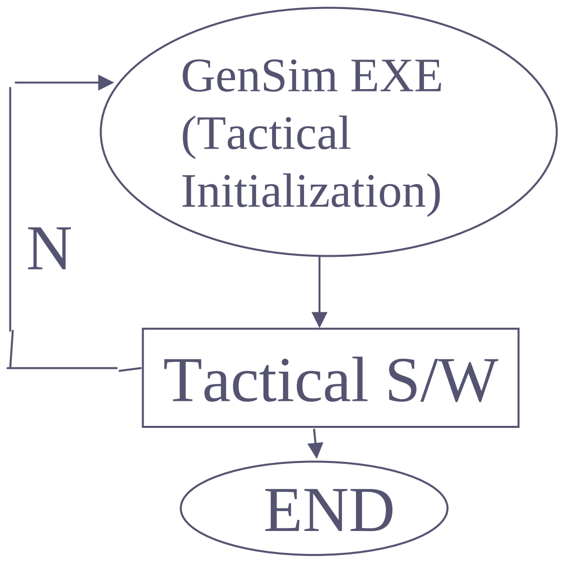
Blue, Green is To Master
Red is from Master

This figure shows a three processor, three executable DLL approach.

Red , Blue and Green denote events where the DLL passes information between processors.

GenSim Evolution (DLL's)

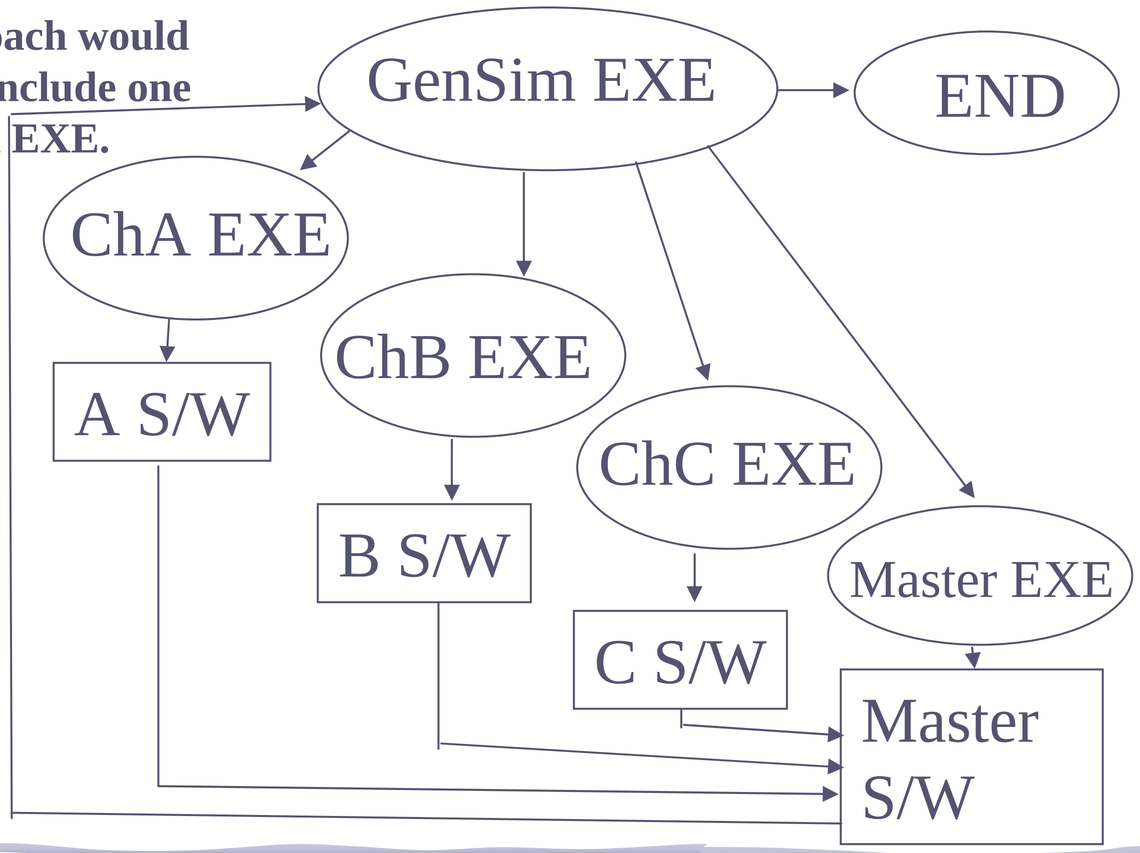
- Before DLL:



Run "N" times
"N" being number
of processors

Note the single processor approach would only include one called EXE.

DLL Version:



GenSim Used for Flight Testing

- GenSim can interface with missile six-degree-of-freedom (6DOF) files (two different file formats).
- 6DOF's can be run to simulate flight test conditions and the files run with GenSim to see how the fuze tactical software will respond to the flight test encounters.
- With this approach, we have been able to diagnose errors in the tactical software that have been fed back to the contractor for fixes. The new tactical software can then be put in the simulation and the process repeated. Flight Test TM data can be compared to simulation output data for post-flight analysis.
- This approach led to the improved DLL tactical software interface.

Notes and Comments

- By modeling Sensor TXT/RCV and Target reflectivity in the GenSim simulation we have a much improved simulation for doing missile fuze design verification and validation.
- Having the ability to “drop-in” a version of proximity fuze tactical software and run numerous tactical missile scenarios gives us an ability to find defects in the tactical software as well as predict tactical operation before any actual flight tests are performed. Post-Analysis with flight test TM can be compared.
- The Dynamic-Linked-Library (DLL) approach to interfacing the tactical software to the GenSim simulation simplified the interface, and improved the information handshake between GenSim and Tactical.