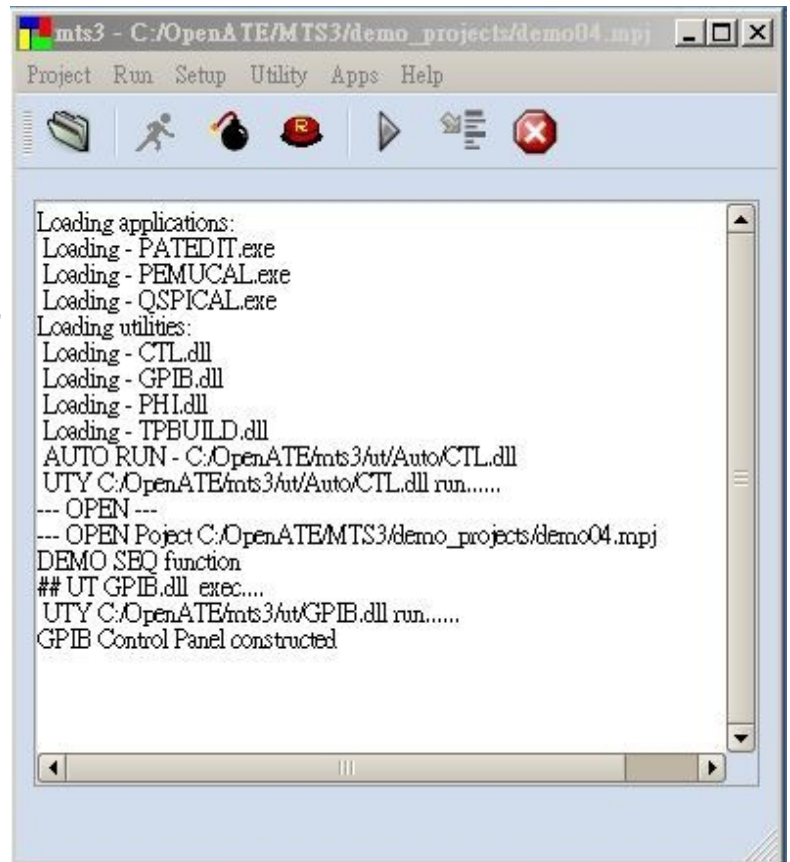


Key Features:

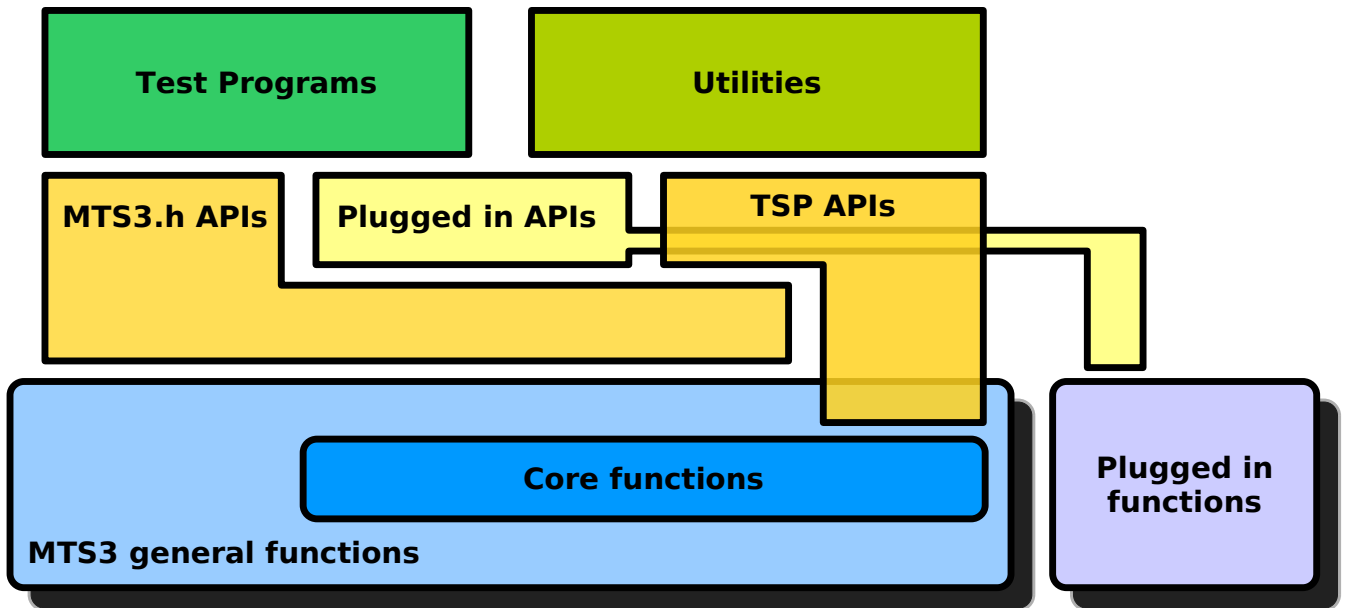
- ATE software platform which can run with any hardware on Windows.
- User reconfigurable for new instruments API and utilities with very low effort.
- User friendly test program development environments with C/C++.
- Integrated debugging capabilities: DATALOG , DEBUG, OVERRIDE, STOP-ON-FAIL.
- Loop test capability
- Summary report ready
- Datalog to file or STDF data format
- Multi-sites testing capability, no need to modify test program. One single test program for multi-sites testing.
- Reconfigurable prober / handler interface.
- Can connect and type of prober / handler from any peripheral supplier.



```
mts3 - C:/OpenATE/MTS3/demo_projects/demo04.mpj
Project Run Setup Utility Apps Help
Loading applications:
Loading - PATEDIT.exe
Loading - PEMUCAL.exe
Loading - QSPICAL.exe
Loading utilities:
Loading - CTL.dll
Loading - GPIB.dll
Loading - PHI.dll
Loading - TPBUILD.dll
AUTO RUN - C:/OpenATE/mts3/ut/Auto/CTL.dll
UTY C:/OpenATE/mts3/ut/Auto/CTL.dll run.....
--- OPEN ---
--- OPEN Project C:/OpenATE/MTS3/demo_projects/demo04.mpj
DEMO SEQ function
## UT GPIB.dll exec....
UTY C:/OpenATE/mts3/ut/GPIB.dll run.....
GPIB Control Panel constructed
```

Overview:

- The OpenATE MTS3 test system is designed specifically of the evaluation of mixed-signal devices in both engineering environment; including a set of rich debugging tools and utilities. Also the MTS3 test system supports multi-site testing capabilities; programming/debugging activities are very easy under such multi-site environment. Also the linking between the MTS3 test system and probe/handler is easy to make.
- The MTS3 test system is built on Windows operating system, which operated on a PC (Personal Computer) hardware platform. The MTS3 test system provides a visual IDE (Integrated Development Environment) for creating test program, executing instructions in test program and test data management during testing.
- With graphical user interface applies PC power to aid program development, the less experienced test engineer can accelerate the learning curve and allow fast test program development.
- Adding different instrument card into the system is very easy, and MTS3 test system provides different utilities to help user to master different instrument cards.

Software Architecture:**Test program structure:**

- The MTS3 test system defines a default program structure to organize the whole test program. This default program structure divides the test program into four modules: INIT, TEST, EOT, POWERDOWN. Each module has their own specific function and will be invoked according to their function during one test sequence. User must follow the module specific function to arrange the test program code.

INIT -- The INIT module is responsible for the initialization of the test program. User should place all the initial routines or start-up procedures in here.

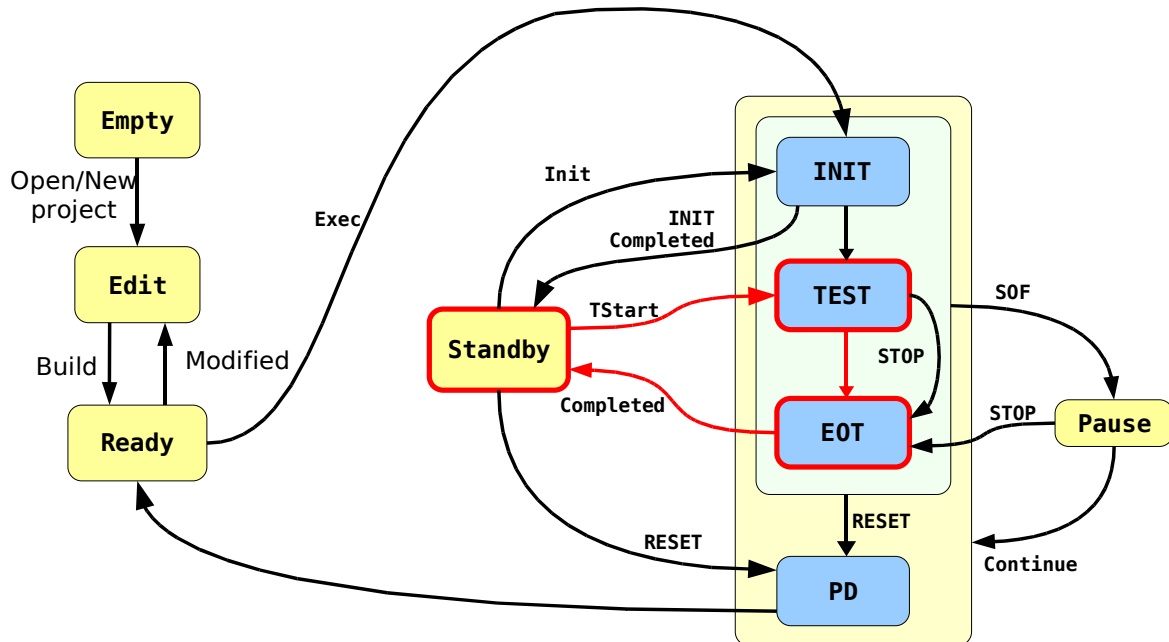
TEST -- The TEST module is the main test sequence. User should put all the test statements that applied to DUT in this module. Device pass/fail decision and binning strategy should be placed in this module.

EOT -- The EOT module will be called after TEST module is executed completed; so the EOT module also applied to each device under test.

POWERDOWN -- The POWERDOWN module (PD) is invoked when user want to leave the MTS3 system and hoping to turn off all instrument powers insure safety of the test system.

- The following is the state diagram for MTS3:
 - When user opens or creates a project, Exec it, the INIT module will be invoked and then the system stays at Standby state and waiting for TStart from PHI.

- When MTS3 receives TStart, the TEST module will be invoked, then followed by EOT module, and then after TEST & EOT have completed execution, the system goes back to Standby again and waits for next TStart.
- When RESET button is pressed by user during above operations, the PD will be invoked, and then the system goes to Ready state.



Creating & editing test program:

- At MTS3's main form, select Project → open to open an existing project.
- To create new project, select Utility->TPBUILD
Select File->New to create new project.
- In TPBUILD, Push "Compile" to build working project.
- Once build has completed successfully, select Run → Exec to start the test program.

General functions:

```
void mts3_msg( const char *sFormat, ... );
```

- Print message.
- Usage: arguments are same as c standard function "printf".

```
int mts3_getState();
```

- Get current sequencer state.
- Return:
 - 1 = STATE_EDIT
 - 2 = STATE_READY
 - 3 = STATE_INIT
 - 4 = STATE_TEST
 - 5 = STATE_EOT
 - 6 = STATE_PD
 - 7 = STATE_STANDBY
 - 8 = STATE_PAUSE

```
int mts3_getSite();
```

- Return the site# of current site in multi-site test application.

```
bool mts3_get_SOF();
```

- Return button 'SOF' state of main dialog.

```
bool mts3_get_debug();
```

- Return button 'Debug' state of main dialog.

```
bool mts3_get_dataLog();
```

- Return button 'DataLog' state of main dialog.

```
bool mts3_get_override();
```

- Return button 'Override' state of main dialog.

```
void mts3_start_timer();
```

```
double mts3_get_timer();
```

- These two functions are used to measure the time elapsed from one point to another in the test program.

- `mts3_start_time()` : Mark current time.
- `mts3_get_timer()` : Return the time elapsed (in ms) since last call to `mts3_start_time()`.
- If datalog is on, the time will also be printed in the datalog pane.

`void mts3_pause(char *msg);`

- Pause execution of test program until user hits continue button.

`int mts3_getMaxSite();`

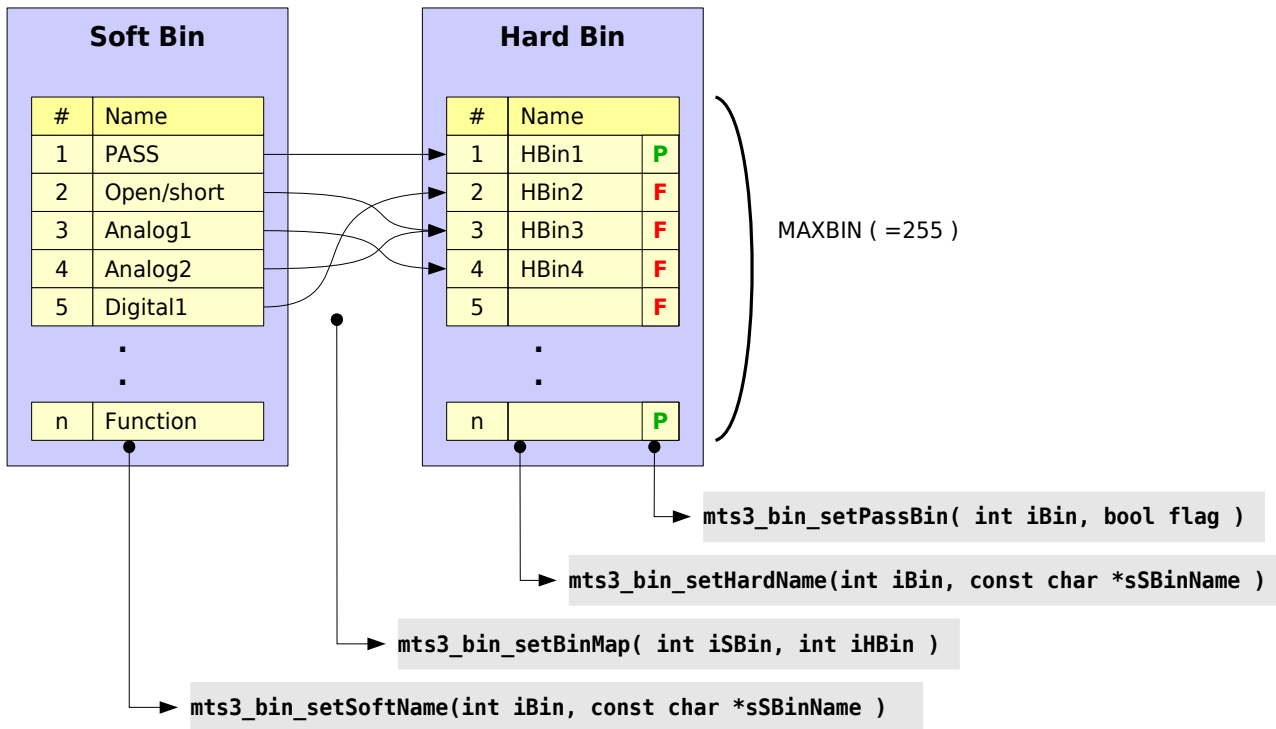
- Return maximum sites available by system. For now, it should be 64.

`int mts3_getNOS();`

- Return number of sites for current test configuration.

Bin related functions:

- The following diagram illustrated the binning database. In a test program, calling **mts3_bin(int iBin)** will set the bin of current test.



```
void mts3_bin( int iBin );
```

- Issue bin for current test.

```
void mts3_bin_setSoftName( int iBin, char *s );
```

- Set name of specified software bin.

```
void mts3_bin_setHardName( int iBin, char *s );
```

- Set name of specified hardware bin.

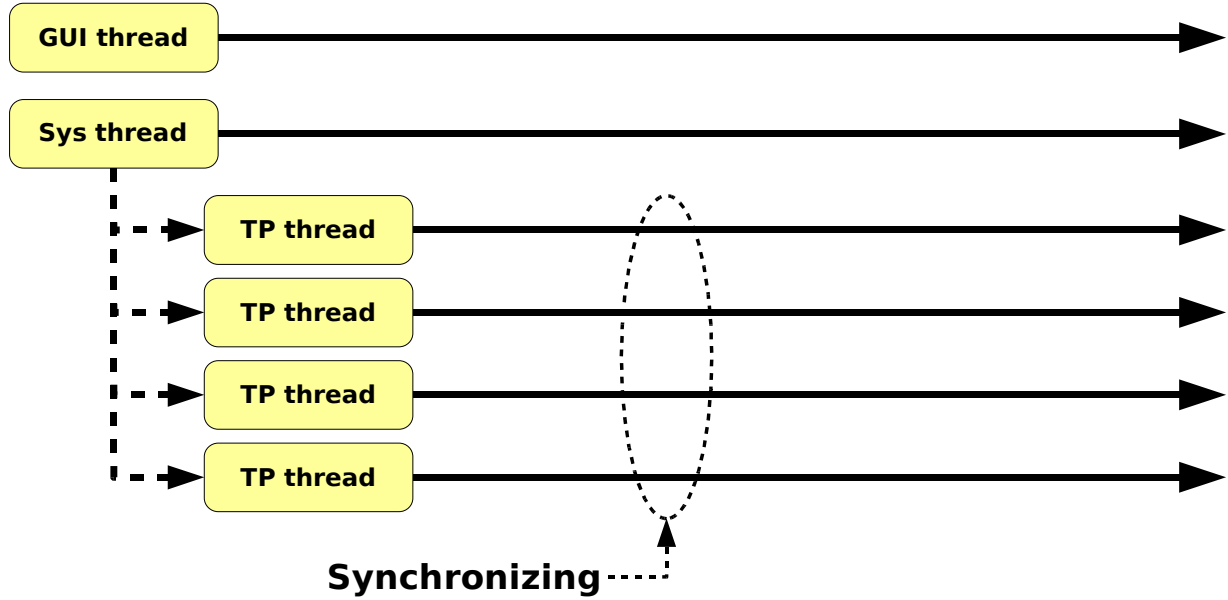
```
void mts3_bin_setMapping( int iSBin, int iHBin );
```

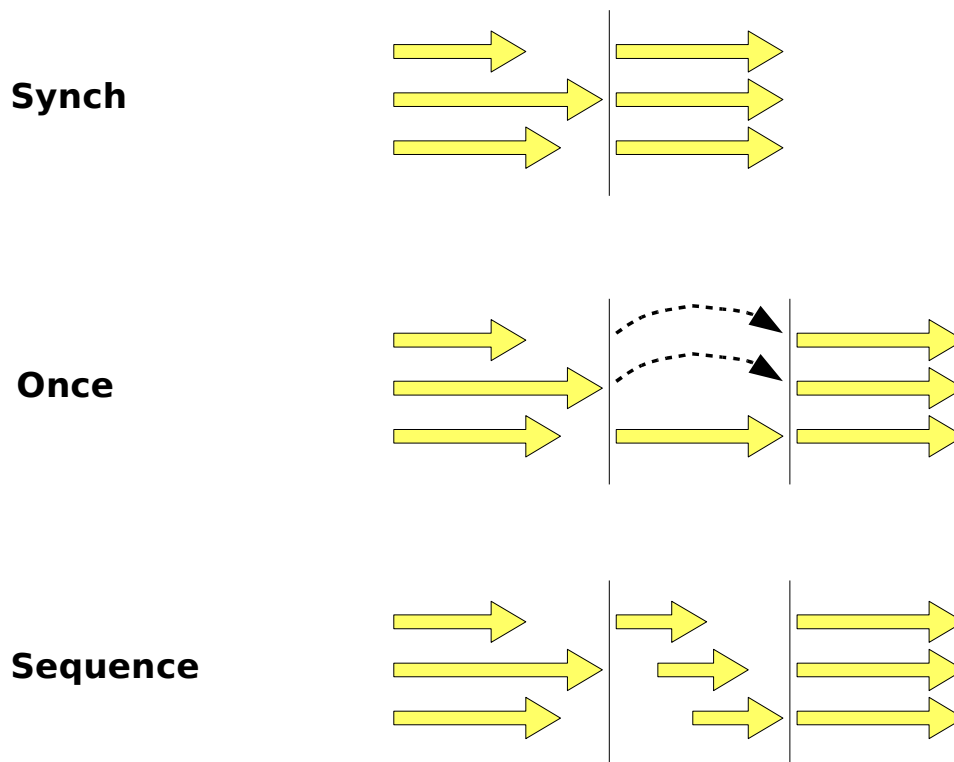
- Map specified software-bin to hardware-bin.

```
void mts3_bin_setPassBin( int iBin, bool flag );
```

- Defines pass or fail for specified hardware-bin.

Threading:



Synchronizing:**Synchronization related functions:**

```
bool mts3_once();
```

- Sometimes for a multi-site application, a piece of code may need to be executed only once by one thread. For example, to initialize a shared resource. `mts3_once()` is used for such situation.

- Usage:

```
...  
if ( mts3_once() ) {  
    // statements in this block will be executed only once by one thread.  
}  
...
```

- `mts3_once()` is also used to sync the execution among test threads.

```
void mts3_seq_begin() / void mts3_seq_end();
```

- Sometimes for a multi-site application, a piece of code may not be able to be executed concurrently by more than one thread at the same time.
- Usage: Suppose we have a four-site application. In the following examples, first site that arrives at point A will pause, waiting for other threads to arrive, and so does the second and the third thread, until the last thread arrives at point A. Then, one thread will continue to execute, when it reach point B, it will pause again, then the second thread will continue to execute, then the third and the fourth thread, until all thread arrive at point B, then all thread continue execute together.

```
...
mts3_seq_begin(); // ---- Point A
// statements in this block will be executed by one thread at a time.

mts3_seq_end(); // ---- Point B
...
```

```
void mts3_check_stop();
```

- In user's test program, add this function call in a infinite loop, so that system can interrupt the loop when user click 'stop'.
- Ex: In the following code, if hardware failed to response, the program may stuck in infinite loop.

```
while (1) {
    if ( hardware_ready() ) exit;
}
```

- Here we put a call to mts3_check_stop() in the loop, so that user can hit stop button to stop the infinite loop:

```
while (1) {
    if ( hardware_ready() ) exit;
    mts3_check_stop();
}
```