Angel Handler Application Model Development

All Rights Reserved - ADVANTEST CORPORATION

Goal: Customized the Appmodel for Angel handler, The Appmodel is developed for handtest automation, it lets Angel handler to communicate with thermostream to control testing temperature and handle with testing parts, make sure 93K system communicate with Angel handler properly and retrieve correct part ID and test temperature for temp cycling and profiling.

Benefit: Customer can VNC Angel handler and 93K system, remotely debug temperature testing without technician help.

All Rights Reserved - ADVANTEST CORPORATION

Install customized appmodel for angel handler

ADVANTEST

Customized Application model for angel handler will be supported from qappmodel version 4.6.

Run qapp_install in terminal, specify option –test angel.

term% *qapp_install –i testflow.ttf –test angel -rev 4.6*

Production test program will be generated: *testflow_handler_angel.tp.*

Applicat file also will be created: *qappmodel_handler_angel_4p6.app*

Soft link is set: *qappmodel_handler_angel_4p6*

testflow_handler_angel.tp

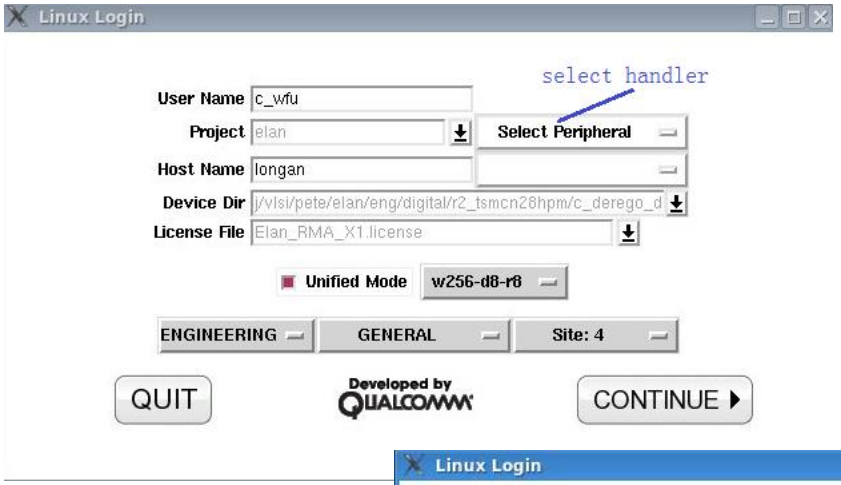
```
hp93000,testprog,0.1
Testflow: testflow.tff
Userproc:
Waferdescr:
Applicdata: qappmodel_handler_angel_4p6.app
Pdi_lib: /opt/hp93000/soc_common/Pdi_lib/exec_input_test
Hp_lib: /opt/hp93000/soc/PH_libs/GenericHandler
Dataform:
```

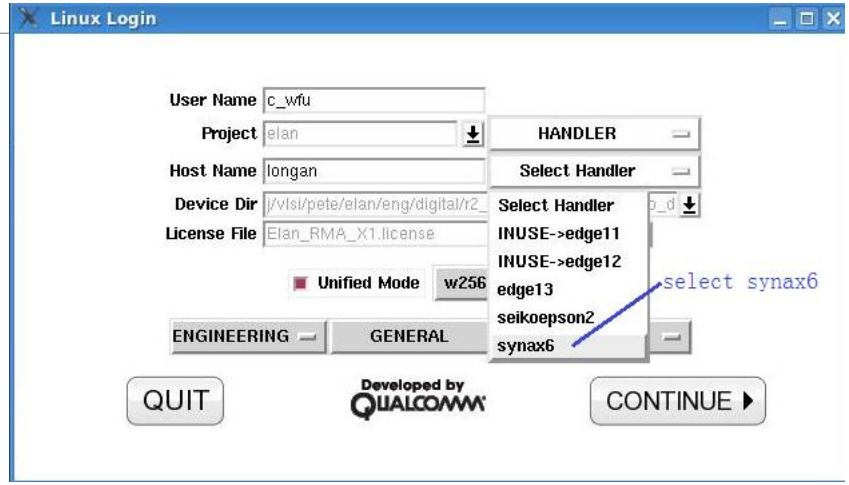
Application Model file is linking to the customized angel handler.

All Rights Reserved - ADVANTEST CORPORATION

Select handler driver for angel handler

sm to start up Smartest and select handler driver for angel handler.



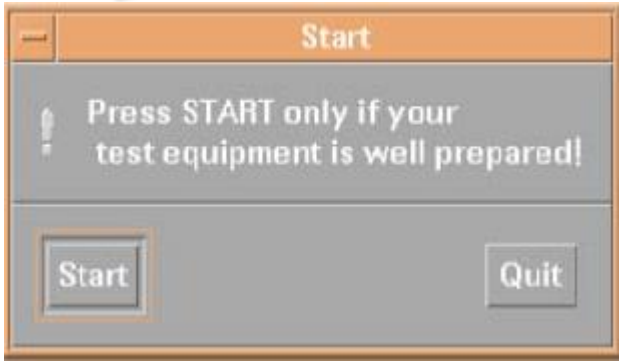


Use synax6 driver for angel handler.

All Rights Reserved - ADVANTEST CORPORATION

Application Model

```
START_DIALOG
```



" Press Start…" dialog

```
{ init:
    ACTIONS
        * APPMODEL_TYPE          = CONST_INPUT(bin);

     EXEC_INP_CALL(qappmodexec setFlags);
                 -- MOVE_EVENT_STREAM();     -- Clear Datalog Stream
```

Pre-action and setting system flag: CI_LOG_EVENTS_ENABLE

All Rights Reserved - ADVANTEST CORPORATION

Application Model

```
LOG_SPECIFICATION
    RESULT_LOG = ON;

{ lot:

    ACTIONS
        ----------------------------
        -- GET GENERAL INFORMATION --
        ----------------------------
        * PROGRAM       = GET_TEST_INFO (testprog);
        * PROGRAM_REV   = GET_TEST_INFO (test_rev);
        * PROGRAM_DESC  = GET_TEST_INFO (test_desc);
        * START_TIME    = GET_TEST_INFO (date_time);
```

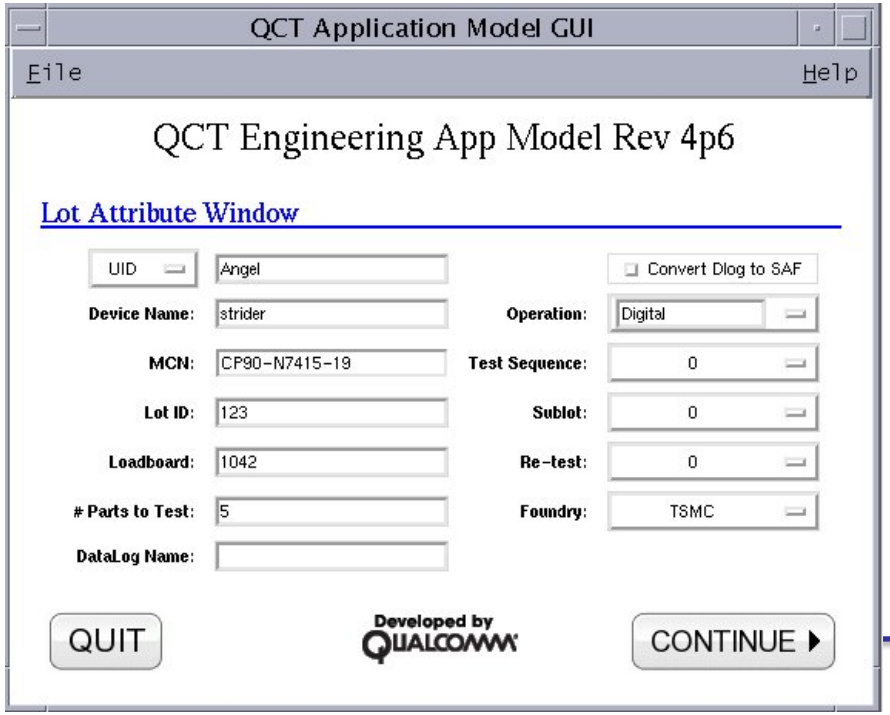
Specifies the results are logged, Determines whether the data is logged or not.

Get general information: testflow information, STDF keywords and thermal stream and loops variables.

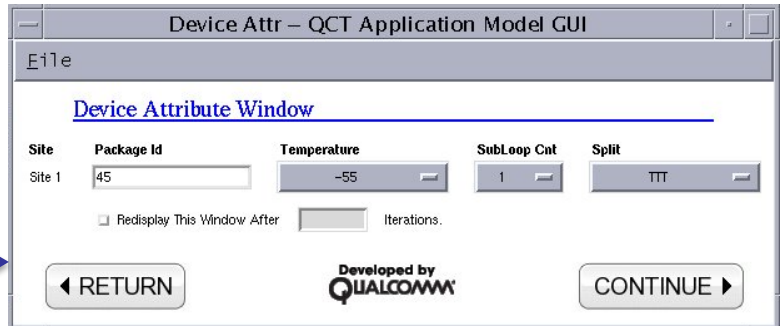
All Rights Reserved - ADVANTEST CORPORATION

Application Model

```
------------------------
-- Launch Perl/Tk GUI --
------------------------
EXEC_INP_CALL(qappmodexec getAttributes angel handler);
```



Invoke a function call that will launch Perl Device Window for user to input Lot attribute and device attribute (part ID, testing temperature, sub loop, split etc.) All these info will be saved in temp .xtra file, not log to DTR.



All Rights Reserved - ADVANTEST CORPORATION

Application Model

```
--------------------------------
-- Read in values from a file --
--------------------------------
ATTR_FILE           = FILE_INPUT({DEVICE_DIR}/report/.temp_qappmodel/{USERNAME}.{HOSTNAME});
* TISIRT        = ELEMENT_OF_PARAM_LIST({ATTR_FILE}) / REGEXP (string);
* OPERATOR          = ELEMENT_OF_PARAM_LIST({ATTR_FILE}) / REGEXP (string);
* TESTER        = ELEMENT_OF_PARAM_LIST({ATTR_FILE}) / REGEXP (string);
* OPERATION         = ELEMENT_OF_PARAM_LIST({ATTR_FILE}) / REGEXP (string);
* SEQUENCE          = ELEMENT_OF_PARAM_LIST({ATTR_FILE}) / REGEXP (string);
* THERMAL       = ELEMENT_OF_PARAM_LIST({ATTR_FILE}) / REGEXP (string);
* POSTPROCESS       = ELEMENT_OF_PARAM_LIST({ATTR_FILE}) / REGEXP (string);
* DATABASE          = ELEMENT_OF_PARAM_LIST({ATTR_FILE}) / REGEXP (string);
* DEVICE_NAME       = ELEMENT_OF_PARAM_LIST({ATTR_FILE}) / REGEXP (string);
```

Read in values from created file.

```
----------------------------------------------------
-- Generate Directory Structure for saving datalog --
----------------------------------------------------
EXEC_INP_CALL(qappmodexec mkDirs {PATH}/report/templates/);
EXEC_INP_CALL(qappmodexec mkDirs {PATH}/report/ascii);
EXEC_INP_CALL(qappmodexec mkDirs {PATH}/report/ascii/{TISIRT});
EXEC_INP_CALL(qappmodexec mkDirs {PATH}/report/bin);
EXEC_INP_CALL(qappmodexec mkDirs {PATH}/report/bin/{TISIRT});
EXEC_INP_CALL(qappmodexec mkDirs {PATH}/report/stdf);
EXEC_INP_CALL(qappmodexec mkDirs {PATH}/report/stdf/{TISIRT});
EXEC_INP_CALL(qappmodexec cleanDirs eng stdf);
EXEC_INP_CALL(qappmodexec cleanDirs eng bin);
```

Generate directory structure for saving datalog.

All Rights Reserved - ADVANTEST CORPORATION

Application Model

```
-- A call to ph_driver_start is mandatory
PROB_HND_CALL (ph_driver_start);

-- Optionally the handler driver may be reconfigured
-- PROB_HND_CALL (ph_set_configuration debug_level: 1);
PROB_HND_CALL (ph_set_configuration trace_driver_calls: "no");
PROB_HND_CALL (ph_set_configuration driver_message_log: "/tmp/{LOT}.driver_messages" driver_error_log:   "/tmp/{LOT}.driver_errors");

-- Optional calls to retrieve information from handler driver
* PH_driver_id    = PROB_HND_CALL (ph_get_id driver);
* PH_plugin_id    = PROB_HND_CALL (ph_get_id plugin);
* PH_equipment_id = PROB_HND_CALL (ph_get_id equipment);
* PH_family       = PROB_HND_CALL (ph_get_configuration handler_family);
* PH_model        = PROB_HND_CALL (ph_get_configuration model);
* PH_plugin       = PROB_HND_CALL (ph_get_configuration driver_plugin);
* PH_config       = PROB_HND_CALL (ph_get_configuration configuration);
```

Starts and initializes handler driver according to the global configuration. Retrieve information from handler driver.

All Rights Reserved - ADVANTEST CORPORATION

Application Model

```
LOG_SPECIFICATION
    RESULT_LOG    = ON;
    -- ASCII Datalog Report File
    NEW_RESULT_FILE = ({PATH}/report/ascii/{TISIRT}/{TISIRT}_{MCN2}-{MCN3}_{LOT}_s{SUBLOT}r{RETEST}s{SEQUENCE}{DLOG_APPEND}dlog);
{ subloop:
    ACTIONS
        SUBLEVEL_COUNT  no_of_iterations = CONST_INPUT({DEVICE_LOOPCOUNT});
    { DEVICE device:
        ACTIONS
            --EXEC_INP_CALL(qappmodexec setPackageID);
            -- Get a device from the handler
            -- You may measure the handling performance by removing the
            -- comment signs of the ph_timer_start and ph_timer_stop calls
            -- and of the NEW_RESULT_FILE specification below.
            -- ph_timer_start and ph_timer_stop may be nested.
            -- PROB_HND_CALL (ph_timer_start);
            PROB_HND_CALL (ph_device_start);
            -- * PH_get_device_time = PROB_HND_CALL (ph_timer_stop);
             EXEC_INP_CALL(qappmodexec ReadParIDTemperatureWriteXtraFile);
            * SITE1_PART_ID  = CONST_INPUT({site1_part_id});
            * SITE1_PART_TXT = CONST_INPUT({site1_part_txt});
            * SITE2_PART_ID  = CONST_INPUT({site2_part_id});
            * SITE2_PART_TXT = CONST_INPUT({site2_part_txt});
            * SITE3_PART_ID  = CONST_INPUT({site3_part_id});
```

Tells handler to insert devices and waits for test start signal from handler. Call *EXEC_INP_CALL(qappmodexec ReadParIDTemperatureWriteXtraFile)* to retrieve Part ID and temperature from handler for dataloging purpose, these info will overwrite the info in .xtra file.

All Rights Reserved - ADVANTEST CORPORATION

Application Model

ADVANTEST

EXEC_INP_CALL(qappmodexec ReadParIDTemperatureWriteXtraFile)

```
else if(strcmp(argv[1],"ReadParIDTemperatureWriteXtraFile") == 0) {
  char tempfilename[200];
  char *filename;
  INST handler;
  char res[32];
  char act_tem[32]="";
  char act_part[32]="";
  int length;
  fprintf(stderr, "Retrieve the testing temperature from handler\n");
  handler = iopen("lan[192.168.0.101]:gpib0,6");
  itimeout(handler,2000);
  iprintf(handler,"TEMP?\n");
  iscanf(handler,"TEMP %s\r\n",&res);
  iclose(handler);
  length = strlen(res);
  strncpy(act_tem, res, length-2);
  fprintf(stderr, "Read back result for TEMP = %s\n", act_tem);

  fprintf(stderr, "Retrieve the testing device ID from handler\n");
  handler = iopen("lan[192.168.0.101]:gpib0,6");
  itimeout(handler,2000);
  iprintf(handler,"DVID?\r\n");
  iscanf(handler,"DVID %s\n",&res);
  iclose(handler);
  length = strlen(res);
  strncpy(act_part, res, length-2);
  fprintf(stderr, "Read back result for DVID = %s\n",act_part );

  makeXtraFile(tempfilename);
  filename = tempfilename;
  WriteXtraFile(filename,act_part,act_tem);
  readXtraFile(filename,1);
}
```

Send GPIB command to inquire temperature from handler and retrieve currently testing temperature.
Example:
Retrieve the testing temperature from handler
Read back result for TEMP = -40

Send GPIB command to inquire part ID from handler and retrieve currently testing part ID.
Example:
Retrieve the testing device ID from handler
Read back result for DVID = 004

Overwrite the temperature and part ID info in the .xtra file and log these info to DTR for datalog purpose.

All Rights Reserved - ADVANTEST CORPORATION

Application Model

```
                LOG_SPECIFICATION
                    RESULT_LOG = ON;
                    {
                        DEVICE_TEST
                    }
                    -- Reprobe may always be defined,
                    -- even if the used handler does not provide reprobing
                    REPROBE_ACTION = PROB_HND_CALL(ph_reprobe);
            ACTIONS
                -- Clearing package ID
                SET_PACKAGE_ID();
                -- Bin devices
                -- You may measure the handling performance by removing the
                -- comment signs of the ph_timer_start and ph_timer_stop calls.
                -- PROB_HND_CALL (ph_timer_start);
                -- PROB_HND_CALL (ph_device_done);
                -- * PH_bin_device_time = PROB_HND_CALL (ph_timer_stop);
                -- Calling only the Package and temp and split window
                -- EXEC_INP_CALL(qappmodexec setPackageAndTempAndSplit eng);
        } -- device
        ACTIONS
            PROB_HND_CALL (ph_device_done);
    } -- subloop
    ACTIONS
        -- close the handler driver
        PROB_HND_CALL (ph_driver_done);
        MOVE_EVENT_STREAM({PATH}/report/bin/{TISIRT}/{TISIRT}_{MCN2}-{MCN3}_{LOT}_s{SUBLOT}r{RETEST}s{SEQUENCE}{DLOG_APPEND}bin);
} -- lot
```

Send a reprobe command to handler, sends the binning data to the handler, terminates the driver.

All Rights Reserved - ADVANTEST CORPORATION

Customized Application Model Datalog

The customized APP is applied to Dino on Yuzu with SMT7.1.4, we have collected some datalog for several parts' temperature cycling with Site 3 enabled, part ID (001→002→003→004) and each part goes through temperature cycling(25 , -40, 110).

UIDAngel_ND629-90_1234_s0r0s0_172057.stdf

UIDAngel_ND629-90_1234_s0r0s0_172058.dlog

All Rights Reserved - ADVANTEST CORPORATION