1. Specman C interface for C API for MatLab dissection

1.1. C interface API ‘sn_matlab.c’

It contains the list of following functions to initialize and connect to MatLab API.
SN_TYPE(matlab_rc_t) sn_matlab_init(t__matlab_ifc_s ifc)
Description:
This function initialize the sn/matlab interface by following the list of step below.

1.1.1. Check if a connection has already been established.

The following code is from c interface API from Matlab software. For more detailed explanation on c interface API for MatLab software, consult Matlab documentation to know what engOpen and other API do. Simply, it tries to open link to Matlab engine. It this function cannot connect to Matlab, you will see message that says that “Can't start MATLAB engine.”
The cause of not being able to connect to the engine could be mismatch of gcc library or compilation error. Please, make sure that you installed decent Matlab software and first test their c api to open link to Matlab engine to test Matlab is installed properly. Matlab documentation tells you which gcc version has been used for its dynamic library. The gcc installation could be found at IUS installation directory which has serveral version and you will find it is very useful.

```
//Sample c funtion to open link to Matlab api.
if (!(ep = engOpen("\0"))) {
        fprintf(stderr, "\nCan't start MATLAB engine\n");
            return SN_ENUM(matlab_rc_t, ERROR);
}
```

1.1.2. If not try to establish one and save its engine pointer into mbsn->ep

```
if(mbsn->ep == 0) {
//get a pointer if we don't have one.
                printf("\n***** connecting to the matlab engine. please wait...*****\n");
                if (!(ep = engOpen("\0"))) {
                        return SN_ENUM(matlab_rc_t, ERROR);
                }
                printf("***** sn_matlab_init: ep=%x", ep);
                DBG_OUT(ifc->cdbg_flag, "sn_matlab_init():\n\t got a valid pointer." );
                //save the pointer to the matlab environment
                ifc->eng = ep;
                mbsn->ep = ep;
                //register sn_matlab_finish to be executed when we exit specman.
                if( atexit(sn_matlab_finish) != 0 ) {
                        return SN_ENUM(matlab_rc_t, ERROR);
                };
} else {
                //chk validity of mbsn->ep and assign it to ifc->ep
                ifc->eng = mbsn->ep;
                ep = mbsn->ep;
```

};

1.1.3. Specman C API ‘set_mx_ptr’

The following set_mx_ptr function sets address for mx_array that has been created by mxCreateNumericArray method in Matlab api. Notice that mx_ptr_upper keeps higher 32bit address and mx_ptr for lower address.

```
set_mx_ptr(ifc->mxc,(unsigned long) mbsn->mxcp);
void set_mx_ptr( s__main__matlab_mx_array_s * arr, unsigned long mxarray) {
  arr->mx_ptr = (unsigned long) mxarray;
  arr->mx_ptr_upper = mxarray>>32;
};
```

1.1.4. Specman C API ‘sn_matlab_exec’

This method evaluates MatLab command string and returns whether MatLab evaluates the string or not. 0 means ok and 1 means there is error at the MatLab command string.

```
SN_TYPE(matlab_rc_t) sn_matlab_exec(t__matlab_ifc_s ifc, SN_TYPE(string) line)
```

1.1.5. Specman C API ‘sn_matlab_get_var’

This method gets variable from matlab which has type for var of type sn_matlab_var_s.

```
SN_TYPE(matlab_rc_t) sn_matlab_get_var(t__matlab_ifc_s ifc,
SN_TYPE(sn_matlab_var_s) var)
```

1.1.6. Specman C API ‘sn_matlab_put_var’

The following method puts variable from has type for var of type sn_matlab_var_s. to matlab which has type for var of type sn_matlab_var_s.

```
SN_TYPE(matlab_rc_t) sn_matlab_put_var(t__matlab_ifc_s ifc,
SN_TYPE(sn_matlab_var_s) var)
```

1.1.7. Specman C API ‘sn_matlab_finish()’

sn_matlab_finish() will be called at exit. Therefore, it will be called when Specman exits.

2. Specman API ‘matlab.e’ and ‘sn_matlab.e’

These two e files has data types and api to call MatLab C API to load MatLab code and get value from MatLab workspace and put value into variable in MatLab workspace.

2.1. Data type

2.1.1. Struct ‘sn_matlab_double_s’

This struct supports 64bit and double precision.

2.1.2. Struct ‘sn_matlab_complex_s’

This data type supports real value and imaginary value as well.

2.1.3. Struct ‘matlab_mx_arrray’

This struct contains name of array, and class(int32,int64 or double), number of rows and columns, and mx_ptr to point to an external mxArray MatLab struct.

2.1.4. Struct ‘sn_matlab_var_s’

This struct contains name, and its class, and real part data array, imaginary part data array, mx_array.

3. Usage example to integrate to verification environment

3.1. Making a Matlab file as a list of string

First of all, a user need to load Matlab file as a list of string. The following example shows how to load matlab file as a list of string and use matlab_ifc_s structure to load the string.

Matlab command as a list of string

```
var sl : list of string =
{
"mx_int_in = fix(100 * rand(20,90));";
"%create a random matrixwith entries in [1..100]";
"%20 rows on 90 colomns";
"mx_int = int32( mx_int_in );";
"%convert the matrix to an integer matrix.";
"mx_dbl_in = rand(20,90) ;";
"%round to 6 places";
"mx_dbl = fix(mx_dbl_in * 1000000) * 0.000001;"
};
```

3.2. Loading the command as string variable.

```
compute matlab.exec("s1");
```

3.3. Getting the variable value from MatLab

To get the variable value from the matlab code you have loaded, you need to have a variable using type sn_matlab_var_s and give constraint to the name field with the name you used in the loaded matlab code. Besides, use get_var() method in matlab_ifc_s. This

method has c_interface to interface MatLab api to look for the value with the variable name. Refer to the following example.

```
mx_int : sn_matlab_var_s; //retrieve and save the original matlab data

keep     mx_int.name == "mx_int"

compute matlab.get_var(mx_int);
```

3.4. Converting matlab_mx_array_s type to sn_matlab_var_s

Matlab_mx_array_s is used to point to the MatLab workspace by using mx_ptr which point to memory address of variable. To convert matlab_mx_array type to sn_matlab_var_s, you need to use mx_to_sn(mx: matlab_mx_array_s). This function has a c_interface called sn_matlab_mx_to_sn_var (t__matlab_ifc_s ifc, SN_TYPE(matlab_mx_array_s) mx, SN_TYPE(sn_matlab_var_s) snvar) and convert matlab data type to specman sn_matlab_var_s type.

3.5. Comparing data

3.5.1. Comparing two mx_arry type

To compare two matlab_mx_array type, use is_equal_mx(mx_int.mx_arr, mx_int_res.mx_arr) function of matlab_ifc_s. This function returns “TRUE” for match or “FALSE” for mismatch.

4. Running test

4.1. Setting up environment

In the specman_matlab_interface_example.tar, there is c shell script that will setup proper Specman and IUS tools. Currently, this shareware has been tested for 6.2 and 8.2 version.

4.2. Building

To build the shareware, type make which will do make using makefile. It will compile c Matlab api.

4.3. Running

To run the shareware, type make test which will run you through test instruction if you type ‘tick’ command at Specman prompt.

4.4. Log file

The following log message is generated after running shareware. It tool is built properly, you will see MatLab log pop up and disappear. After connection with MatLab is properly setup, you can experience following log message.

```
Welcome to Specman Elite(64) sn_matlab (08.20.006-s)  -  Linked on Wed Sep  2
20:25:26 2009
```

```
@> load test_get_cmp_mdim.e

Loading test_get_cmp_mdim.e ...

@> test

Doing setup ...
Generating the test using seed 1...

Starting the test ...
Running the test ...
***** Matlab-Test: use the tick command to move through the stages (10 in total).
***** Matlab-Test: use:
        'MB deb'                to turn on the matlab interface debug messages.
        'MB cdeb'               to turn on the matlab C interface debug messages.
        'MB deb off'    to turn off the matlab interface debug messages.
        'MB cdeb off'   to turn off the matlab C interface debug messages.
        The default for debug messages is off
Use the "tick" command to advance the time
Specman test_get_cmp_mdim> display.change_window_width(".main0", 88)
Specman test_get_cmp_mdim> display.change_window_width(".main0", 90)
Specman test_get_cmp_mdim> display.change_window_width(".main0", 96)
Specman test_get_cmp_mdim> display.change_window_width(".main0", 106)
Specman test_get_cmp_mdim> display.change_window_width(".main0", 113)
Specman test_get_cmp_mdim> display.change_window_width(".main0", 117)
Specman test_get_cmp_mdim> display.change_window_width(".main0", 119)
Specman test_get_cmp_mdim> display.change_window_width(".main0", 120)
Specman test_get_cmp_mdim> display.change_window_width(".main0", 121)
Specman test_get_cmp_mdim> display.change_window_width(".main0", 121)
Specman test_get_cmp_mdim> MB deb
Specman test_get_cmp_mdim> tick
2. send commands..
matlab.exec(): execute cmd: mx_int_in = fix(100 * rand(20,90));
matlab.exec(): cmd output=
matlab.check_result(): last matlab command result = SUCCESS
matlab.exec(): execute cmd: %create a random matrixwith entries in [1..100]
matlab.exec(): cmd output=
matlab.check_result(): last matlab command result = SUCCESS
matlab.exec(): execute cmd: %20 rows on 90 colomns
matlab.exec(): cmd output=
matlab.check_result(): last matlab command result = SUCCESS
matlab.exec(): execute cmd: mx_int = int32( mx_int_in );
matlab.exec(): cmd output=
matlab.check_result(): last matlab command result = SUCCESS
```

matlab.exec(): execute cmd: %convert the matrix to an integer matrix.
matlab.exec(): cmd output=
matlab.check_result(): last matlab command result = SUCCESS
matlab.exec(): execute cmd: mx_dbl_in = rand(20,90) ;
matlab.exec(): cmd output=
matlab.check_result(): last matlab command result = SUCCESS
matlab.exec(): execute cmd: %round to 6 places
matlab.exec(): cmd output=
matlab.check_result(): last matlab command result = SUCCESS
matlab.exec(): execute cmd: mx_dbl = fix(mx_dbl_in * 1000000) * 0.000001;
matlab.exec(): cmd output=
matlab.check_result(): last matlab command result = SUCCESS
Specman test_get_cmp_mdim> tick
3. retrieve data..
matlab.get_var(v): getting variable = 'mx_int' from matlab workspace
matlab.check_result(): last matlab command result = SUCCESS
matlab.get_var(v): getting variable = 'mx_dbl' from matlab workspace
matlab.check_result(): last matlab command result = SUCCESS
Specman test_get_cmp_mdim> tick
4. do some calculations...
matlab.exec(): execute cmd: sin_res = sin(mx_dbl_in);
matlab.exec(): cmd output=
matlab.check_result(): last matlab command result = SUCCESS
matlab.exec(): execute cmd: asin_res = asin(sin_res);
matlab.exec(): cmd output=
matlab.check_result(): last matlab command result = SUCCESS
matlab.exec(): execute cmd: mx_dbl_res = fix(asin_res * 1000000) * 0.000001;
matlab.exec(): cmd output=
matlab.check_result(): last matlab command result = SUCCESS
matlab.exec(): execute cmd: r1 = fix(1000 * rand(20,90));
matlab.exec(): cmd output=
matlab.check_result(): last matlab command result = SUCCESS
matlab.exec(): execute cmd: t1 = mx_int_in + r1;
matlab.exec(): cmd output=
matlab.check_result(): last matlab command result = SUCCESS
matlab.exec(): execute cmd: mx_int_res = int32(t1 - r1);
matlab.exec(): cmd output=
matlab.check_result(): last matlab command result = SUCCESS
Specman test_get_cmp_mdim> tick
5. retrieve data..
matlab.get_var(v): getting variable = 'mx_int_res' from matlab workspace
matlab.check_result(): last matlab command result = SUCCESS
matlab.get_var(v): getting variable = 'mx_dbl_res' from matlab workspace
matlab.check_result(): last matlab command result = SUCCESS
Specman test_get_cmp_mdim> tick
6. comparing ...

matlab.is_equal_mx(): comparing data of:
	mx1.name= 'mx_int' and
	mx2.name = 'mx_int_res' with 20 rows and 90 columns.
matlab.is_equal_mx(): comparing data of:
	mx1.name= 'mx_dbl' and
	mx2.name = 'mx_dbl_res' with 20 rows and 90 columns.
Specman test_get_cmp_mdim> tick
7. convert to sn_var...
matlab.mx_to_sn(): converting mx_arr.name = 'mx_int' with
	20 rows and 90 columns to a sn_var.
matlab.mx_to_sn(): converting mx_arr.name = 'mx_int_res' with
	20 rows and 90 columns to a sn_var.
matlab.mx_to_sn(): converting mx_arr.name = 'mx_dbl' with
	20 rows and 90 columns to a sn_var.
matlab.mx_to_sn(): converting mx_arr.name = 'mx_dbl_res' with
	20 rows and 90 columns to a sn_var.
Specman test_get_cmp_mdim> tick
8. do element wise comparison of sn_var and a mx_arr...
sys.compare_sn_var_to_mx_arr_by_item(sn_var, mx_arr):
	comparing sn_var with mx_arr.name = 'mx_int_res' element wise.
	mx_arr (rows,cols)=(20,90).
	sn_var number of uint elements = 1800
sys.compare_sn_var_to_mx_arr_by_item(sn_var, mx_arr):
	comparing sn_var with mx_arr.name = 'mx_dbl_res' element wise.
	mx_arr (rows,cols)=(20,90).
	sn_var number of uint elements = 3600
Specman test_get_cmp_mdim> tick
9. put the sn_var's back into matlab...
matlab.put_var(v): putting variable = 'sn_var_int' to matlab workspace
matlab.check_result(): last matlab command result = SUCCESS
matlab.put_var(v): putting variable = 'sn_var_int_res' to matlab workspace
matlab.check_result(): last matlab command result = SUCCESS
matlab.put_var(v): putting variable = 'sn_var_dbl' to matlab workspace
matlab.check_result(): last matlab command result = SUCCESS
matlab.put_var(v): putting variable = 'sn_var_dbl_res' to matlab workspace
matlab.check_result(): last matlab command result = SUCCESS
Specman test_get_cmp_mdim> tick
10. read back the sn_var_dbl_res into mx_dbl_res_pb and compare with mx_dbl...
matlab.get_var(v): getting variable = 'sn_var_dbl_res' from matlab workspace
matlab.check_result(): last matlab command result = SUCCESS
matlab.mx_to_sn(): converting mx_arr.name = 'sn_var_dbl_res' with
	1800 rows and 1 columns to a sn_var.
sys.compare_sn_var_to_mx_arr_by_item(sn_var, mx_arr):
	comparing sn_var with mx_arr.name = 'sn_var_dbl_res' element wise.
	mx_arr (rows,cols)=(1800,1).
	sn_var number of uint elements = 3600

```
matlab.is_equal(): comparing data of:
        sn_var with 1800 elements and
        mx_arr.name = 'sn_var_dbl_res' with 1800 rows and 1 colomons.
***** finish checking interface *****
  session.num_of_dut_errors = 0
  session.num_of_dut_warnings = 0
Specman test_get_cmp_mdim> display.change_window_width(".main0", 121)
Specman test_get_cmp_mdim> display.change_window_width(".main0", 121)
Specman test_get_cmp_mdim> display.change_window_width(".main0", 121)
Specman test_get_cmp_mdim> display.change_window_width(".main0", 121)
Specman test_get_cmp_mdim> display.change_window_width(".main0", 121)
Specman test_get_cmp_mdim> display.change_window_width(".main0", 122)
Specman test_get_cmp_mdim> display.change_window_width(".main0", 125)
Specman test_get_cmp_mdim> display.change_window_width(".main0", 129)
Specman test_get_cmp_mdim> display.change_window_width(".main0", 132)
Specman test_get_cmp_mdim> display.change_window_width(".main0", 133)
Specman test_get_cmp_mdim> display.change_window_width(".main0", 134)
Specman test_get_cmp_mdim> display.change_window_width(".main0", 134)
Specman test_get_cmp_mdim> display.change_window_width(".main0", 134)
Specman test_get_cmp_mdim> display.change_window_width(".main0", 134)
Specman test_get_cmp_mdim> display.change_window_width(".main0", 134)
Specman test_get_cmp_mdim> display.change_window_width(".main0", 134)
Specman test_get_cmp_mdim> quit
```