



Application Note #4424

Programming MATLAB[®] with Galil controllers

Due to the advanced mathematical and plotting capabilities, as well as GUI support, MATLAB can be an ideal host interface for Galil motion control applications. This application note will demonstrate the basic programming needed to communicate with Galil motion controllers using the GalilTools API, the ActiveX toolkit and the DMCWin32 API function calls. Examples are provided that will demonstrate how to create a simple m-file application, as well as a Windows based GUI front end. The examples are developed using MATLAB 7.1 (R14 service pack 3).

MATLAB is a registered trademark of The MathWorks, Inc. For more information, go to: <http://www.mathworks.com/products/matlab/>

1	USING THE GALILTOOLS API WITH MATLAB 7.0 R14.....	2
1.1	STEP BY STEP INSTRUCTIONS FOR MATLAB AND THE GALILTOOLS API.....	2
2	USING THE ACTIVEX TOOLS WITH MATLAB.....	3
2.1	CONNECTING TO THE CONTROLLER	3
2.2	PLOTTING DATA AND USER INPUT	6
2.3	USING THE ACTIVEX TOOLS WITHOUT A GUI.....	9
3	USING THE DMCWIN32 API WITH MATLAB	10
3.1	THE DMC32.DLL "HELLOGALIL" EXAMPLE.....	10
3.2	DMC32 DLL FUNCTION LIST	11

1 Using the GalilTools API with Matlab 7.0 R14

The GalilTools API uses Galil's latest driver to communicate from Matlab to a Galil controller and is recommended for new designs.

For more information on the GalilTools software, go here:

<http://www.galilmc.com/products/galiltools.php>

1.1 Step by Step Instructions for Matlab and the GalilTools API

(1) Install GalilTools version 1.3.0.0 or newer on Windows XP or newer. Lite is free of charge, the full version required a purchased password. Either is fine for this walkthrough.

<http://www.galilmc.com/support/software-downloads.php>

(2) Connect to the controller using GalilTools

<http://www.galilmc.com/support/manuals/galiltools/connections.html#available>

(3) Open MathWorks Matlab 7.0 R14, select File | New | M File.

(4) Enter the following code in the m file

```
%HelloGalil example shows basic communication set up using GalilTools API
g = actxserver('galil');%set the variable g to the GalilTools COM wrapper
response = g.libraryVersion;%Retrieve the GalilTools library versions
disp(response);%display GalilTools library version

g.address = ":%Open connections dialog box
response = g.command(strcat(char(18), char(22)));%Send ^R^V to query controller
model number
disp(strcat('Connected to: ', response));%print response

response = g.command('MG_BN');%Send MG_BN command to query controller for
serial number
disp(strcat('Serial Number: ', response));%print response

delete(g);%delete g object and close connection with controller
```

(5) Push F5 to compile and run the application. Choose a location and name for the m-file

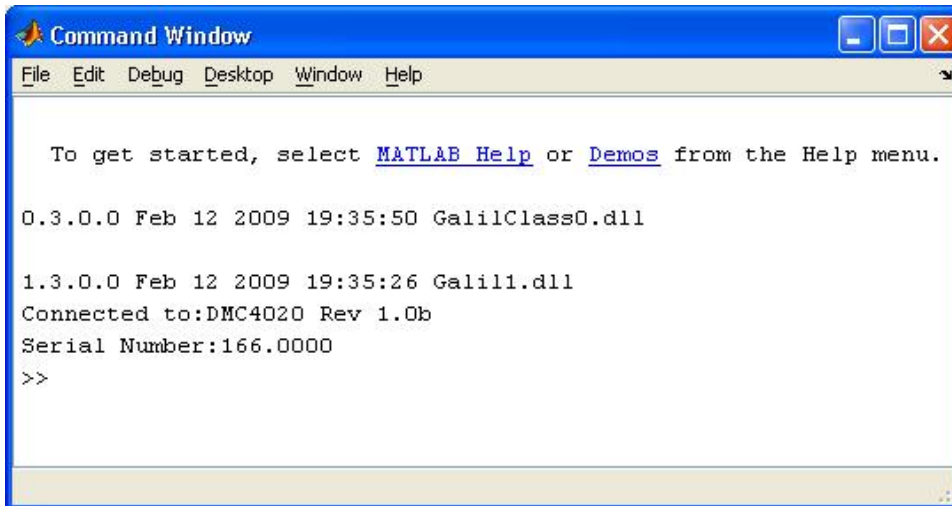
(6) The program will display the connections dialog, identical to GalilTools. Choose the same controller you connected to in step (2). The output should be similar to:

0.3.0.0 Feb 12 2009 19:35:50 GalilClass0.dll

1.3.0.0 Feb 12 2009 19:35:26 Galil1.dll

Connected to:DMC4020 Rev 1.0b

Serial Number:166.0000



The screenshot shows a 'Command Window' window with a blue title bar and standard Windows window controls. The menu bar includes 'File', 'Edit', 'Debug', 'Desktop', 'Window', and 'Help'. The main text area contains the following text:
To get started, select [MATLAB Help](#) or [Demos](#) from the Help menu.
0.3.0.0 Feb 12 2009 19:35:50 GalilClass0.dll
1.3.0.0 Feb 12 2009 19:35:26 Galil1.dll
Connected to:DMC4020 Rev 1.0b
Serial Number:166.0000
>>

Also See:

<http://www.galilmc.com/support/manuals/galiltools/library.html>

2 Using the ActiveX tools with MATLAB

The Galil ActiveX toolkit can be used to quickly develop a MATLAB GUI application since we have created ready-made control objects that are easily integrated into a GUI project. The ActiveX toolkit features various objects that simplify tasks such as communications, data display, and even creating a terminal interface. Since MATLAB 7 fully supports ActiveX control properties, methods, and events, the programming needed to take advantage of Galil features such as “Motion Complete”, and “Unsolicited message and Interrupt” servicing become simplified.

For more information on the ActiveX toolkit go to:

<http://www.galilmc.com/products/software/softwaretools.html#activetoolkit>

2.1 Connecting to the Controller

To begin, assuming the ActiveX toolkit is pre-installed on the PC, we will demonstrate a simple GUI application to connect to the controller, send a command, and then display the response. This example will demonstrate the use of the DMCSHELL.ocx object, which is the base ActiveX control needed to communicate with the controller and subsequently required for all other Galil ActiveX tools to operate.

First launch MATLAB 7.1 and then select **file/new/GUI** from the pull-down menu. Select a blank GUI and hit OK. In the GUI layout environment, select the ActiveX icon (large moving X) from the toolbar on the left and draw this anywhere on the form with the mouse. This will prompt a dialog box (Figure 1) where we will select the DMCSHELL control from the list and then hit create.

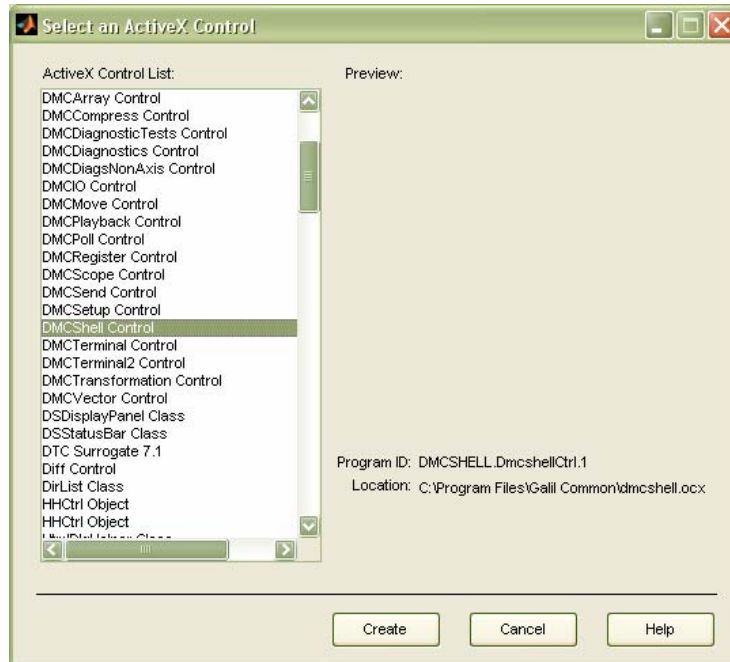


Figure 1 – ActiveX control selection Dialog

Note the **DMCSHELL** control is a hidden object and will not appear in the final execution of the form; hence, you can drag this to an inconspicuous place in the corner, for example. In the same manner, also select a “Push Button” and “Static Text” box to draw on the form. The application should look similar to figure 2 below.

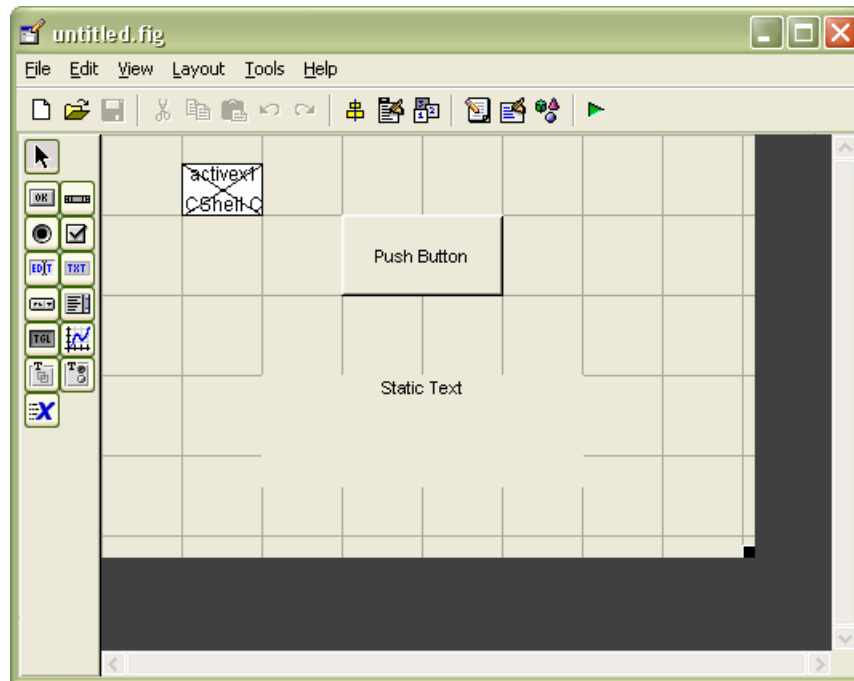


Figure 2 – GUI layout

At this point, we are ready to add the code needed to connect, send the command, and display the response. We will use the callback associated with the button to handle these tasks. To spawn the code editor and access the pushbutton callback, select **view/m-file editor** from the pull-down menu, this will prompt you to save the file before proceeding. Once in the code editor, scroll down to the pushbutton1_callback routine and enter the following:

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject      handle to pushbutton1 (see GCBO)
% eventdata    reserved - to be defined in a future version
% handles      structure with handles and user data (see GUIDATA)
handles.activex1.DMCCConnect=1;
handles.activex1.DMCCCommand='CW';
s=handles.activex1.DMCRresponse;
set(handles.text1,'String',s);
handles.activex1.DMCCConnect=0;
```

This simple application will basically connect to controller #1, send a “CW” command to display the Galil copyright information from the controller, and then disconnect. To run the application, hit the green start arrow on the GUI layout toolbar. Push the button and the following window (figure 3) should be displayed.

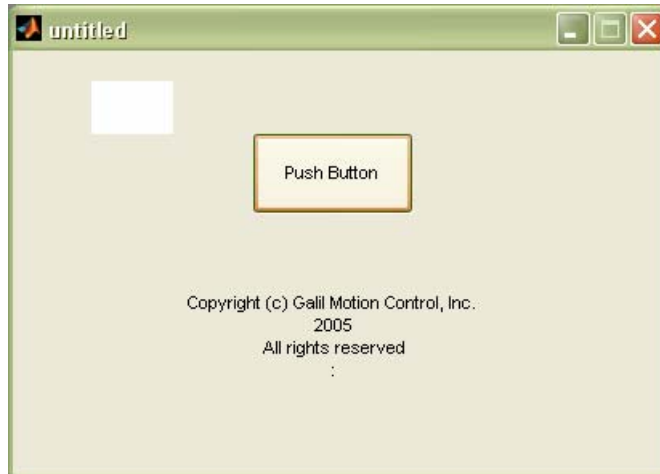


Figure 3 – GUI output

2.2 Plotting Data and User Input

At this point we are able to communicate with the controller and send and receive data. However since the controller responds to commands in ASCII strings, and most data used in MATLAB needs to be in some sort of numerical format, the following example will demonstrate how to convert the Galil responses into useful data in order to generate a plot. Similar to the Galil WSDK software, the following example will demonstrate how to retrieve data from a step response move and display the position data within a 2-D plot.

The key to converting Galil responses to numerical data is to use the `strtok()` and `str2Double()` function to first trim the ASCII response of the carriage return-linefeed-colon, then convert the string to a double precision numerical value.

```
data=str2double(strtok(handles.activex1.DMCResponse));
```

Using the technique learned in the previous example to connect to the controller, we will create a new application that includes a new button and an axis plot object. This application also demonstrates how to use edit text boxes to allow the user to input variable arguments to the DMC commands.

The following code demonstrates how to: dimension an array on the controller, download a simple user-defined program that moves the motor, collect position data into an array, upload the array data, and then display the data in a 2-D axis plot.

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
    % hObject    handle to pushbutton1 (see GCBO)
```

```

% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

%create carriage return and linefeed variable
CRLF=[char(13) char(10)];

%Create data array on controller
handles.activex1.DMCCCommand='DMTEST[300]';
%setup RC to record position
handles.activex1.DMCCCommand='RATEST[]';
handles.activex1.DMCCCommand='RD_TPX';

%create DMC program to download
move = get(handles.edit2,'String');
DMC = ['#A' CRLF];
DMC = [DMC 'DP0' CRLF];
DMC = [DMC 'SP100000' CRLF];
DMC = [DMC 'PR' move ';BGX' CRLF];
DMC = [DMC 'RC1' CRLF];
DMC = [DMC 'EN'];

handles.activex1.DMCFileBuffer = DMC;
handles.activex1.DMCFileOperation = 4; %download from buffer
handles.activex1.DMCCCommand='XQ';

%Wait for data recording to complete ...poll for _RC
s = 1;
while s > 0
    handles.activex1.DMCCCommand = 'MG_RC';
    s=str2double(strtok(handles.activex1.DMCResponse));
end

%Upload array data to MATLAB array
numpoints = 300;
for n = 1:numpoints;
    i = n-1;
    arrayval = strcat('TEST[', num2str(i), ']=?');
    handles.activex1.DMCCCommand = arrayval;
    data = str2double(strtok(handles.activex1.DMCResponse));
    mydata(n) = data;
end

plot(mydata)
xlabel('Time (samples)');
ylabel('Position (counts)');
grid on

```

Figure 4 below shows a typical plot display of position data from the example application.

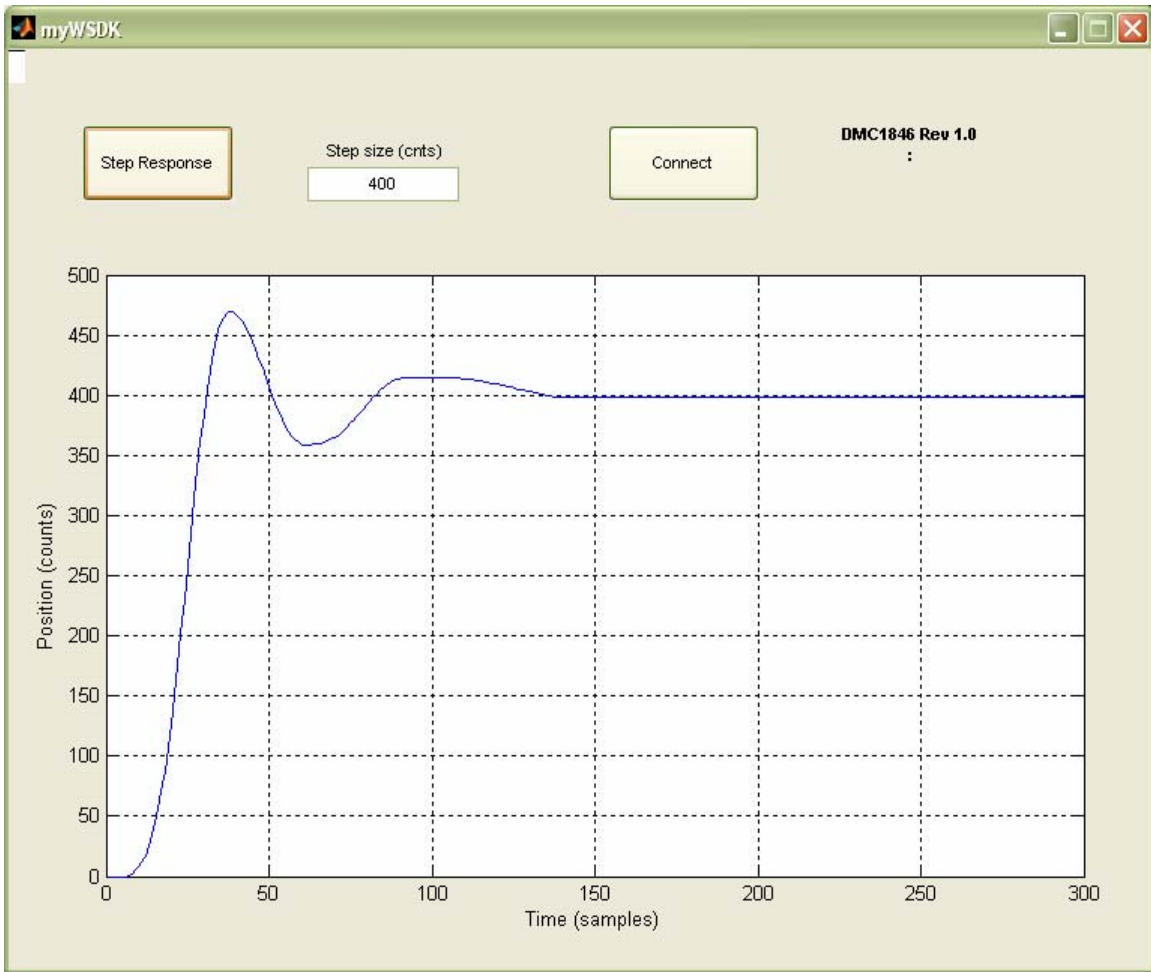


Figure 4 - myWSDK

2.3 Using the ActiveX Tools without a GUI

In some applications it may not be necessary or desirable to have a GUI interface. In this case, the Galil ActiveX controls can be accessed directly at the code level to provide communications with a Galil controller. The example below utilizes the `actxcontrol()` function call directly to establish the DMCSHELL object and make the connection to the controller. The `actxcontrol()` function requires the ActiveX control ID as the argument. A complete list of the ActiveX ID's are found in the Windows registry editor located at: (My Computer/HKEY_LOCAL_MACHINE/SOFTWARE/Classes)

In the simple example below, a connection to controller #1 is made and “^R^V” and “MG_BN” commands are sent to the controller to retrieve the controller model information and serial number and display the responses to the command window.

```
%Create Controller shell object
Controller = actxcontrol('DMCSHELL.DmcsShellCtrl.1');

%Connect to controller (1=connect; 0=disconnect)
Controller.DMCCConnect = 1;

%Send "^R^V" to query controller model
Controller.DMCCCommand = strcat(char(18), char(22));
%print response
disp(strcat('Connected to:', ' ', Controller.DMCResponse));

%Send "MG_BN" command to query controller serial number
Controller.DMCCCommand = 'MG_BN';
%print response
disp(strcat('Serial Number:', Controller.DMCResponse));

%Close connection to controller
Controller.DMCCConnect=0;
%delete all resources for the Galil DMCSHELL object
delete(Controller);
```

The output of this example (shown below) is simply text displayed to the Command Window; however, since the ActiveX tools are objects, a figure window will spawn upon creation of the DMCSHELL object. This figure could be used if the data output is to be plotted, or just simply ignored.

```
Connected to:DMC1846 Rev1.0
:
Serial Number: 45.0000
:
>>
```

3 Using the DMCWin32 API with MATLAB

In MATLAB 7.1, direct access to Windows Dynamic Link Libraries (DLL's) is now supported. As an alternative to using the Galil ActiveX toolkit, the Galil DLL libraries can be used to make a connection to the controller and perform task such as sending commands, downloading programs, and accessing data record information. To access the Galil libraries, download the DMCWin32 API (application program interface) from the Galil website at: <http://www.galilmc.com/support/download.html>

For complete information on the DMCWin32 API, download the user manual at: <http://www.galilmc.com/support/manuals/dmcwin.pdf>

3.1 The DMC32.dll "HelloGalil" Example

The DMC32.dll file is the fundamental library needed to interface with Galil controllers. The installation of DMCWin32 includes the necessary header file needed to define the function calls that are present in the DMC32.dll library, as well as the drivers and DLL files themselves. The DMCCOM.h header file that defines the API interface is included in the Galil directory as noted in the example below.

```
%Hello_Galil
%Sample code for communicating with a Galil controller using the
%DMCWin32 api under Matlab 7.1

% Loading the Galil DMC32.dll library into Matlab:
hfile = ['C:\Program Files\Galil\DMCWIN\INCLUDE\dmccom.h'];
dllfile = ['C:\WINDOWS\system32\DMC32.dll'];
loadlibrary (dllfile, hfile);

%To view all DMC32 api functions in dialog box (uncomment line below)
%libfunctionsview DMC32

hDmc = 0; % create handle variable with dump value
p_hDmc = libpointer('int32Ptr', hDmc); % creating a pointer

Controller=1; %Select controller #1 in Galil registry

%Call DMCOpen to connect to controller
rc = calllib('DMC32', 'DMCOpen', Controller, 0, p_hDmc);
hDmc=get(p_hDmc, 'Value'); % get the real handle to the controller

%report if failed connection
if rc ~= 0
    status = rc;
    data=0;
    disp(['Could not connect to controller. RC=', num2str(rc)]);
    return
end
```

```

szBuffer = num2str(zeros(1,256)); %create response variable of fixed
length 256
% Now you can send commands to the controller using the function call
DMCCCommand():

%send "^R^V" to get controller model info
sCommand=strcat(char(18),char(22));
[rc,sCommand,sResponse] = calllib('DMC32', 'DMCCCommand', hDmc,
sCommand, szBuffer, length(szBuffer));
Controller = sResponse

%send "MG_BN" to get controller serial number
sCommand='MG_BN';
[rc,sCommand,sResponse] = calllib('DMC32', 'DMCCCommand', hDmc,
sCommand, szBuffer, length(szBuffer));
SerialNumber = sResponse

% in the end you should close communications
rc = calllib('DMC32', 'DMCClose', hDmc);
% and unload the library:
unloadlibrary DMC32

```

The output of the program above will be a simple text display like the following;

```

Controller =
DMC1846 Rev 1.0

:
SerialNumber =
45.0000

:
>>

```

3.2 DMC32 DLL function list

A complete list of all function declared in the DMC32.dll can be viewed by invoking the function:

```
libfunctionsview DMC32
```

The dialog box (shown in Figure 5 as a partial list) will help define the required arguments and returned values as supported by MATLAB. This will be helpful since MATLAB only supports return values as opposed to the syntax denoted in the Galil DMCWin32 documentation which implies that the arguments can be passed back by reference.

Return Type	Name	Arguments
[int32, s_GALILREGISTRYPtr, uint16Ptr]	DMCAddGalilRegistry	(s_GALILREGISTRYPtr, uint16Ptr)
[int32, s_GALILREGISTRY2Ptr, uint16Ptr]	DMCAddGalilRegistry2	(s_GALILREGISTRY2Ptr, uint16Ptr)
[int32, s_GALILREGISTRY3Ptr, uint16Ptr]	DMCAddGalilRegistry3	(s_GALILREGISTRY3Ptr, uint16Ptr)
[int32, cstring, cstring, uint32Ptr]	DMCArrayDownload	(int32, cstring, uint16, uint16, cstring, uint32, uint32Ptr)
[int32, cstring, cstring, uint32Ptr]	DMCArrayUpload	(int32, cstring, uint16, uint16, cstring, uint32, uint32Ptr, int16)
[int32, voidPtr, s_GALILREGISTRY2Ptr]	DMCAssignIPAddress	(voidPtr, s_GALILREGISTRY2Ptr)
[int32, uint8Ptr, cstring]	DMCBinaryCommand	(int32, uint8Ptr, uint32, cstring, uint32)
int32	DMCChangeInterruptNotification	(int32, int32)
int32	DMCClear	(int32)
int32	DMCClose	(int32)
[int32, cstring, cstring]	DMCCommand	(int32, cstring, cstring, uint32)
[int32, cstring, uint8Ptr, uint32Ptr]	DMCCommand_AsciiToBinary	(int32, cstring, uint32, uint8Ptr, uint32, uint32Ptr)
[int32, uint8Ptr, cstring, uint32Ptr]	DMCCommand_BinaryToAscii	(int32, uint8Ptr, uint32, cstring, uint32, uint32Ptr)
[int32, cstring, cstring, uint16Ptr]	DMCCompressFile	(cstring, cstring, uint16, uint16Ptr)
[int32, voidPtr]	DMCCopyDataRecord	(int32, voidPtr)
int32	DMCDeleteGalilRegistry	(int16)
int32	DMCDiagnosticsOff	(int32)
[int32, cstring]	DMCDiagnosticsOn	(int32, cstring, int16)
[int32, cstring, cstring]	DMCDownloadFile	(int32, cstring, cstring)
[int32, cstring]	DMCDownloadFirmwareFile	(int32, cstring, int16)
[int32, cstring, cstring]	DMCDownloadFromBuffer	(int32, cstring, cstring)
[lib.pointer, voidPtr]	DMCEditRegistry	(voidPtr)
[int32, uint16Ptr, s_GALILREGISTRYPtr]	DMCEnumGalilRegistry	(uint16Ptr, s_GALILREGISTRYPtr)
[int32, uint16Ptr, s_GALILREGISTRY2Ptr]	DMCEnumGalilRegistry2	(uint16Ptr, s_GALILREGISTRY2Ptr)
[int32, uint16Ptr, s_GALILREGISTRY3Ptr]	DMCEnumGalilRegistry3	(uint16Ptr, s_GALILREGISTRY3Ptr)
[int32, cstring]	DMCError	(int32, int32, cstring, uint32)
[int32, cstring]	DMCFastCommand	(int32, cstring)
[int32, cstring, cstring]	DMCFile_AsciiToBinary	(int32, cstring, cstring)
[int32, cstring, cstring]	DMCFile_BinaryToAscii	(int32, cstring, cstring)
[int32, cstring]	DMCGetAdditionalResponse	(int32, cstring, uint32)
[int32, uint32Ptr]	DMCGetAdditionalResponseLen	(int32, uint32Ptr)
[int32, cstring]	DMCGetControllerDesc	(uint16, cstring, uint32)
[int32, uint16Ptr, int32Ptr]	DMCGetDataRecord	(int32, uint16, uint16, uint16Ptr, int32Ptr)
[int32, uint16Ptr, int32Ptr]	DMCGetDataRecordByItemId	(int32, uint16, uint16, uint16Ptr, int32Ptr)
[int32, stringPtrPtr]	DMCGetDataRecordConstPointer	(int32, stringPtrPtr)
[int32, stringPtrPtr, uint16Ptr]	DMCGetDataRecordConstPointerArray	(int32, stringPtrPtr, uint16Ptr)
[int32, uint16Ptr, uint16Ptr]	DMCGetDataRecordItemOffsetById	(int32, uint16, uint16, uint16Ptr, uint16Ptr)
[int32, uint16Ptr]	DMCGetDataRecordRevision	(int32, uint16Ptr)
[int32, uint16Ptr]	DMCGetDataRecordSize	(int32, uint16Ptr)
[int32, int32Ptr]	DMCGetDelay	(int32, int32Ptr)
[int32, s_GALILREGISTRYPtr]	DMCGetGalilRegistryInfo	(uint16, s_GALILREGISTRYPtr)
[int32, s_GALILREGISTRY2Ptr]	DMCGetGalilRegistryInfo2	(uint16, s_GALILREGISTRY2Ptr)
[int32, s_GALILREGISTRY3Ptr]	DMCGetGalilRegistryInfo3	(uint16, s_GALILREGISTRY3Ptr)
[int32, int32Ptr]	DMCGetHandle	(uint16, int32Ptr)
[int32, int32Ptr]	DMCGetTimeout	(int32, int32Ptr)

Figure 5 – Partial List of DMC32 functions