For: wpm
Printed on: Tue, Sep 14, 1993 14:53:26

Document: CaMath
Last saved on: Wed, Jan 27, 1993 16:27:37

CaMath User's Guide
(by Ben–chin Cha, Bob Daly)

CaMath is an external Mathematica package which can be loaded into Mathematica by a user. CaMath consists of a special set of channel access functions which provides the Mathematica users with easy and flexible access of channel information across the IOC networks. It also provides a completes set of process variable event monitoring functions.

The available functions for CaMath, their functionality, and their syntax are described in following sections.

This document also gives examples how a Mathematica user can interface to channel access devices. It is assumed that the user is already familiar with using Mathematica[1]. Few examples of Mathematica module of using CaMath functions are also given in this document.

1. External MathLink Functions

The available CaMath link functions can be grouped into four categoriess: functions operating on a single channel, functions operating on a list of channels, monitoring functions operating on a single or multiple channels, and general functions for debugging or error handling. A user must make sure that the channel name entered is defined in the database and the IOC is operating normally when using values returned by these functions in his control program.

The single channel functions include **CaGet**, **CaPut**, **CaGetString**, **CaPutString**, **CaGetValue, CaInfo**, **CaGetCount, CaGetWF**, and **CaPutWF**. They can be used interactively to get a value back for a single channel or to set the value of a specified channel name.

The functions operating on a list of multiple channels include **CaGetList, CaPutList**, **CaGetStringList**, **CaPutStringList**, **CaInfoList**, and **CaGetValueList**. These functions can be used to get a list of values back for specified devices or to set values for a specified list of devices.

The value change event monitoring functions include **CaAddMonitor, CaWaitEventMonitor, CaEventMonitor, CaGetMonitor,** and **CaClearMonitor** for operating on a single channel and **CaAddMonitorList, CaWaitEventMonitorList, CaEventMonitorList, CaGetMonitorList,** and **CaClearMonitorList** for operating on a list of multiple channel names. They gives the user control of adding / removing / getting the value change status of the channels to be monitored.

The general functions include **CaHelp, CaSleep, CaPendIO, CaPendIOLIST, CaPendEvent, CaError**, **CaErrorList**, **CaSearchList, CaStatus,** and **CaStatusLis*t.***

The functions defined in CaMath conform to the following naming convention: all function names begins with the prefix "**Ca**", and functions operating on a list of channels names ends with the string "**List**". Normally functions operating on a single channel return a single value and functions operating on a list of channel names return a list of values.

The channel name referred to in this document is the same as the process variable name defined in the DCT database. The input channel name should be an ASCII string with a maximum string length less than 28.

When channel access cannot locate the process variable requested by a user, the device not found warning message is always displayed on the user's terminal screen. Any channel not found from the IOC network is automatically cleared from the channel access search queue.

The channel access default pend IO timeout used is 0.3 second for a single channel, and 30 seconds for a list of channel names; and the default pend EVENT timeout used is 0.01 second. These timeouts can be reset by the user.

The functionality of each CaMath function is described below.

1.1 General / Debug / Error Handling Functions

CaHelp[ ]

This function makes a complete list of the currently available external functions defined for CaMath. It returns nothing.

CaSleep[time]

This function causes the process to sleep for the specified time period. Time is measured in real seconds. Time can be any real number, e.g. 0.5. It returns nothing.

CaPendIO[time]

This function allows the user to reset the TIMEOUT seconds for *ca_pend_io* function[2] when getting single channel information. The default timeout is 0.3 second; a TIMEOUT of zero is forever. It returns nothing.

CaPendIOLIST[time]

This function allows the user to reset the TIMEOUT seconds for processing a list of channels. The default timeout is 30 seconds; a TIMEOUT of zero is forever. It returns nothing.

CaPendEvent[time]

This function allows the user to reset the TIMEOUT seconds for *ca_pend_event* function. The default TIMEOUT is 0.01 second; a TIMEOUT of zero is forever. It returns nothing.

CaDebug[i]

This function sets the runtime printing flag. The default value is initially set to 0 when CaMath is installed. If **i** is set to a positive integer, more debug information about the channel access function is printed. To debug the value change event, the value of **i** should be set to 2. The printing flag can be reset to 0 when no more debug information is desired. It returns nothing.

CaError[ ]

This function querries the error status of the last execution of any channel access command. **CaError** takes no argument and returns 0 if no channel access error encountered in the last function call; otherwise it returns –1.

CaErrorList[{nm1, nm2, . . . }]

The **CaErrorList** expects a list of channel names as input. It checks for channel access errors for each name entered in the input list. It returns a list of the error codes corresponding to the requested

channel names. The error code can be zero or –2. An error code of 0 indicates that the last channel access call completed successfully. An error code of –2 indicates either that the channel device name was not found across the IOC network or another type of channel access error has occurred (most likely the timeout).

CaSearchList[{nm1, nm2, . . . }]

The **CaSearchList** requires a list of channel names as input. It returns a single value: 0 if all channels in the list were found or –1 if any one of the channels was not found on the IOC network. If –1 is returned, the user can immediately call **CaErrorList** to find out which channel was not found on the IOC network.

This function can be called at the very beginning to make sure every channel name in user's program exists across the IOC network.

1.2 Single Channel Access Functions

CaGetValue[nm]

To query the numerical value of a channel use the **CaGetValue** function. The **CaGetValue** function expects a string containing an IOC channel name defined in the database and returns to Mathematica the current value of the channel.

CaGet[nm]

Use the **CaGet** function to query the value, status, and severity of a channel. **CaGet** expects an IOC channel name defined in the database from the user and returns a list to Mathematica. The list consists of value, status, and severity of the channel.

CaPut[nm, rl]

To set the numerical value of a channel use the **CaPut** function. This function expects two input arguments, channel name and new value, and returns none. The set value can be real or integer. A synonymous function name **CaSet** can be used for **CaPut**.

CaGetString[nm]

For some channels the value field is defined in the database as an ASCII string rather than a numerical value. In these cases the function **CaGetString** should be used to get string value for these channels. The CaGetString function expects an input channel name as an argument and returns a string value for the channel.

CaPutString[nm, "string_value"]

For those channels where the value field is defined as an ASCII string rather than a numerical value, function **CaPutString** should be used to put the string value to the record field. **CaPutString** expects two input arguments: channel name and new string value. It returns the new string value.

CaInfo[nm]

The **CaInfo** function expects a channel name as an input argument. It returns a string of text for the querying device. In addition, the channel name, value, status, severity, operating ranges, and units used are displayed on the user's terminal.

CaGetWF[nm]

The **CaGetWF** function expects a channel name as an input argument. It returns a list of real values of the waveform record. The size of the returned list is the same as the size of the waveform record.

CaPutWF[nm, {vl1, vl2, . . .}]

The **CaPutWF** function expects input consisting of channel name and a list of values which should not exceed the size of the waveform record. It writes the input list to the specifed waveform record to the IOC and returns the new values of the waveform record.

CaGetCount[nm]

The **CaGetCount** function expects a channel name as an input argument. It returns the native element count for the specified channel from the IOC network.

CaStatus[nm]

The **CaStatus** function expects a channel name as an input argument. It returns the status of the specified channel from the current internal data structure which is populated from the previous channel access call.

1.3 Multiple Channel Access Functions

CaGetValueList[{nm1, nm2, . . . }]

The **CaGetValueList** function expects a list of channel names as input. It returns a list of the values corresponding to the requested channels.

CaGetList[{nm1, nm2, . . . }]

The **CaGetList** function expects a list of channel names as input. It returns a list of lists which contain {{value1, status1, severity1}, {value2, status2, severity2}, . . . } corresponding to the requested channels.

CaPutList[{nm1, nm2, . . . }, {vl1, vl2, . . . }]

CaPutList expects two lists as input: a list of channel names, and a list of new values to be assigned to the channels. **CaPutList** returns the list of the set values corresponding to the requested channels. The input values can be real or integer.

CaGetStringList[{nm1, nm2, . . . }]

For some channels the value field is defined as an ASCII string rather than a numerical value. Function **CaGetStringList** is used to get a string value for these channels. **CaGetStringList** expects a list of channel names as input and returns a list of string values for the requested channel names.

CaPutStringList[{nm1, nm2, . . . }, {vl1, vl2, . . . }]

For some channels the value field is defined as an ASCII string rather than a numerical value. Function **CaPutStringList** is used to assign a string value to the record field for these channels.

CaPutStringList expects two input arguments: channel name list and new string value list. It returns the list of strings of the new values written to the IOC.

CaInfoList[{nm1, nm2, . . . }]

The **CaInfoList** function expects a list of channel names as the input argument and returns a list of values corresponding to the requested channel names. In addition, the channel name, value, status, severity, units, operating ranges, and units are displayed on the user's terminal for all the requested channel names.

CaStatusList[{nm1, nm2, . . . }]

CaStatusList expects a list of channel names as input and returns a list of the current status codes corresponding to the requested channel device names from the internal data structure populated from the previous channel access call.

1.4 Channel Access Monitoring Functions

CaAddMonitor[nm]

The **CaAddMonitor** function expects a channel name as the input argument. It adds the specifed channel name to the monitored list and automatically adds a change of connection event for the monitored channel.

A new event is generated immediately after adding the event, after each writing of a value to the channel, and after each new connecton with the channel's current value.

CaWaitEventMonitor[wtime, nm]

The **CaWaitEventMonitor** function expects two input arguments: the specified time in seconds for waiting for events to happen, and the channel name of the interested monitored channel. This function returns 0 if any event of the value change happened within the specified time interval and –1 if no value change event happened within the specified time interval since the last call of **CaGetMonitor**.

CaEventMonitor[nm]

CaEventMonitor expects a channel name as the input argument. It queris the status of the value change event for the specifed channel name. It returns 1 if the specifed channel has never been processed and has undergone at least one value change, and returns 0 if there is no new value change event or if the status of changed value already been processed by either **CaEventMonitor** or **CaEventMonitorList**.

CaGetMonitorValue[nm]

The **CaGetMonitorValue** function expects a channel name as the input argument. It gets the current value from the data structure for the specified monitored channel name. The status of the changed event is set to 0 if **CaEventMonitor** was not called before calling **CaGetMonitor**.

CaGetMonitor[nm]

The **CaGetMonitor** function expects a channel name as the input argument. It gets the current {value, status, severity} from the data structure for the specified monitored channel name. The status of the changed event is set to 0 if **CaEventMonitor** was not called before calling **CaGetMonitor**.

CaClearMonitor[nm]

The **CaClearMonitor** function expects a channel name as the input argument. It removes the specified channel name from the monitored list.

CaAddMonitorList[{nm1, nm2, . . . }]

The **CaAddMonitorList** function expects a list of channel names as the input argument. It adds the specified channel names to the monitored list and automatically adds change of connection events for the monitored channels.

For each newly–added monitored channel, a new event is generated immediately after adding the event, after each writing of a value to the channel, and after each new connecton with the channel's current value.

CaWaitEventMonitorList[wtime, {nm1, nm2, . . . }]

CaWaitEventMonitorList expects two input arguments: the specified time in seconds for waiting for events to happen, and the list of channel names of the interested monitored list. This function returns 0 if any event of the value change happened within the specified time interval and –1 if no value change event happened within the specified time interval.

CaEventMonitorList[{nm1, nm2, . . . }]

The **CaEventMonitorList** function expects a list of monitored channel names as the input argument. It queries for the event status of the specifed channel names. It returns a list of event statuses corresponding to the specified list of channel names. The elements of the returned list are either 1 or 0. The returned element is 1 if the value of the specifed channel has changed and it has not been processed by either **CaEventMonitor** or **CaEventMonitorList**. The returned element is 0 if either the value change has been processed by **CaMonitor** or **CaMonitorList** or the value has not changed.

CaGetMonitorValueList[{nm1, nm2, . . . }]

The **CaGetMonitorValueList** function expects a list of monitored channel names as the input argument. It returns a list of current values from the data structure for the specifed channel names. The status of the changed event is set to 0 if **CaEventMonitorList** was not called before calling **CaGetMonitorList**.

CaGetMonitorList[{nm1, nm2, . . . }]

The **CaGetMonitorList** function expects a list of monitored channel names as the input argument. It returns a list of current {{value1, status1, severity1}, {value2, status2, severity2}, ... } from the data structure for the specifed channel names. The status of the changed event is set to 0 if **CaEventMonitorList** was not called before calling **CaGetMonitorList**.

CaClearMonitorList[{nm1, nm2, . . . }]

CaClearMonitorList expects a list of channel names as the input argument. It removes the specifed channel names from the monitored list.

CaConnect[nm]

The **CaConnect** function expects a channel name as the input argument. It checks for the reconnection event of a specified channel. When reconnection happens, a user must wait for reestablishment of channel access in order to guarantee the normal operation of channel access functions.

2. Setup Requirement for CaMath

- Create a subdirectory, **math**, for storing Mathematica–related program files, and make a soft link to **CaMath**, and **CaMath.m**

```
mkdir  math
cd  math
ln  –s  /home/phebos2/CHA/math/CaMath  CaMath
ln  –s  /home/phebos2/CHA/math/CaMath.m  CaMath.m
```

This is a one–time setup requirement for all procedures. It assumes that a user always runs Mathematica in the **math** subdirectory.

3. Procedure for Running CaMath

- Invoke a **math** session
- Load the file **CaMath.m** into **math**
- Enter the channel access functions as described above or load other macro files which consist of functions written in **math** and **CaMath**

Example

An example of running **CaMath** functions interactively under the Open Look window manager is given below. The boldfaced words are user–supplied command syntax, the italicized face words are comments, and the remaining lines are output automatically generated by **CaMath** and Mathematica (**math**), respectively. The Unix host name in the following example is 'pan'.

| *Mathematica session*

pan 292: **math**

```
Mathematica 2.1 for SPARC
Copyright 1988–92 Wolfram Research, Inc.
 — OPEN LOOk graphics initialized —

In[1]:= <<CaMath.m                          Load CaMath.m into math

Uninstall::unlink: External package CaMath has not been installed.
CaMath Installed.
ca_task_initialize

In[2]:= CaHelp[]                            List available CaMath functions
Available CaMath functions include following:
    CaGet["name"],          CaPut["name",value]
    CaGetString["name"],    CaPutString["name","string"]
    CaGetValue["name"],     CaStatus["name"]
    CaGetWF["name"],        CaPutWF["name",{vals}]
    CaGetCount["name"]
    CaInfo["name"]

    CaGetList[ {names}],          CaPutList[{names},{val}]
    CaGetStringList[{names}],     CaPutStringList[{names},{val_string}]
    CaGetValueList[{names}],      CaStatusList[{names}]
    CaInfoList[{names}]

    CaAddMonitor["name"]
    CaWaitEventMonitor[wtime,"name"]
    CaEventMonitor["name"]
    CaGetMonitor["name"]
    CaClearMonitor["name"]

    CaAddMonitorList[{names}]
    CaWaitEventMonitorList[wtime, {names}]
    CaEventMonitorList[{names}]
    CaGetMonitorList[{names}]
    CaClearMonitorList[{names}]

    CaError[]
    CaErrorList[{names}]
    CaSearchList[{names}]
    CaConnect["name"]
    CaDebug[]
    CaSleep[ time ]
    CaPendIO[ time ]
    CaPendIOLIST[time]
    CaPendEvent[ time ]
    CaHelp[]
```

In[3]:= **CaGet[”chademobi1”]** *Get value, status, severity of “chademobi1”*

Out[3]= {0, 0, 0}

In[4]:= **CaPut[”chademobi1”]** *Put value to “chademobi1” w/o given value*

Out[4]= CaPut[chademobi1] *Math evaluation of last command*

In[5]:= **CaPut[”chademobi1”,1]** *Put value of 1 to device “chademobi1”*

In[6]:= **x = ”chademoai1”** *Assign x to “chademoai1”*

Out[6]= chademoai1

In[7]:= **CaGetValue[x]** *Get value for device x*

Out[7]= –7.

In[8]:= **u = {”chademoai1”,”chademomask1”,”chademobi1”}** *Set u to a list of device string*

Out[8]= {chademoai1, chademomask1, chademobi1}

In[9]:= **v = CaGetValueList[u]** *Set v to the return list from caGetValueList*

Out[9]= {17., 10., 1.}

In[10]:= **v** *List contents of list v*

Out[10]= {17., 10., 1.}

In[11]:= **v[[2]]=100** *Reset the 2nd element of v to 101*

Out[11]= 100

In[12]:= **v** *List contents of list v again*

Out[12]= {17., 100., 1.}

In[13]:= **CaPutList[u,v]** *Put new values v to devices specified by u*

Out[13]= {17., 100., 1.}

In[14]:= **CaGetValue[u[[2]]]** *Get the value of the 2nd element of u*

Out[14]= 101.

In[15]:= **CaGetStringList[u]** *Get the string values of the elements of u*

Out[15]= {"–11","100","State 1"}

In[16]:= **u = {"chademobi1","chademomask1"}** *Reset the elements of the list u*

Out[16]= {chademobi1, chademomask1}

In[17]:= **CaAddMonitorList[u]** *Add channels of u to the monitor list*

In[18]:= **CaDebug[2]** *Turn on run–time debug option*

In[19]:= **CaGetList[u]** *Get the value of List u*
Old: nmae=chademobi1, value=1.000000, stat=7, sevr=1, error=–2 *Debug output*
New: nmae=chademobi1, value=1.000000, stat=7, sevr=1, error=0 "
Old: nmae=chademomask1, value=100.000000, stat=0, sevr=0, error=–2 "
New: nmae=chademomask1, value=100.000000, stat=0, sevr=0, error=0 "
GetAllDoubleArray:
chademobi1 value=1.000000, stat= 7, sevr=1
chademomask1 value=100.000000, stat= 0, sevr=0

Out[19]= {{1., 7, 1}, {100., 0, 0}} *Return values from CaGetList[u]*

In[20]:= **CaEventMonitorList[u]** *Check event status of the monitoring list* u
caGetEventList: name=chademobi1, value=1.000000, status=7, event=1 *Debug output*
caGetEventList: name=chademomask1, value=100.000000, status=0, event=1 "

Out[20]= {1, 1}

In[21]:= **CaEventMonitorList[u]** *Recheck the event status of the u again*

Out[21]= {0, 0} *No value changes since last call*

In[22]:= **CaPutList[u,{0,22}]** *Put new values to the monitored channels*
Old: nmae=chademobi1, value=1.000000, stat=7, sevr=1, event=1 *Debug output*
New: nmae=chademobi1, value=0.000000, stat=0, sevr=0, event=1 "
Old: nmae=chademomask1, value=100.000000, stat=0, sevr=0, event=1 "
New: nmae=chademomask1, value=22.000000, stat=0, sevr=0, event=1 "
Old: nmae=chademobi1, value=0.000000, stat=0, sevr=0, error=–2 *Debug output*
New: nmae=chademobi1, value=0.000000, stat=0, sevr=0, error=0 "
Old: nmae=chademomask1, value=22.000000, stat=0, sevr=0, error=–2 "
New: nmae=chademomask1, value=22.000000, stat=0, sevr=0, error=0 "

Out[22]= {0, 22.} *Return list of CaPutList*

In[23]= **CaClearMonitorList[u]** *Remove list u from the monitoring list*

In[24]:= **CaDebug[0]** *Turn off run–time debug*

In[25]:= **CaInfoList[u]** *Get info about all devices in u*

```
DEVICE                    TYPE  VALUE  STATUS  SEVR  UNITS      UOPR     LOPR
chademobi1                 3      0.000000   0  0                0.000000    0.000000
chademomask11              6     22.000000   0  0                0.000000    0.000000

Out[25]= {0, 22. }


In[26]:= ??CaGetWF
CaGetWF[name] – returns the array of waveform of the specified record–name
    string name.

CaGetWF[x_String] :=
  ExternalCall[LinkObject[”CaMath”, 1, 1], CallPacket[31, {x}]]

In[26]:= CaGetWF[”chademowf1”]                    Get waveform record

Out[27]= {0.308825, 0.308825, 0.308825, 0.808898, 1., 0.809137, 0.309211,
>   –0.308822, –0.808896, –1., –0.809138, –0.309214, 0.30882, 0.808895, 1.,
>   0.80914, 0.309216, –0.308817, –0.808893, –1., –0.809141, –0.309219,
>   0.308815, 0.308815, 0.308815, 0.808892, 1., 0.809143, 0.309221,
>   –0.308812, –0.80889, –1., –0.809144, –0.309224, 0.30881, 0.808889, 1.,
>   0.809146, 0.309226, –0.308807}

In[27]:= Quit                                     Quit math session
ca_task_exit                                      Uninstall CaMath functions
pan  293 :                                        Return to Unix prompt
```

In the above example, Out[4] echos Mathematica's evaluation of In[4], because an error in the number of arguments is found in **CaPut**. **CaPut** expects two arguments: device name and set value. Legitimate device names and values must be entered for normal operation of the channel access functions. To ensure the right data is obtained, a user should check the status returned by channel access functions by calling **CaError[]**.

4. Example of External Math Functions

Function for Taking Data and Plotting Results

```
Graph::usage = "Graph[\"devname\",points,seconds] – Plots data versus real time\n
            devname – specifies the requested device name\n
            points  – specifies the data points to be taken\n
            seconds – specifies the time interval in seconds"

Graph[in_String,np___Integer,dt___Real|dt___Integer] := (
   t0=SessionTime[];
   npoints = If[np>0, np, 20];
   interval = If[dt>0, dt, 1];
   mplot=Table[0,{npoints}];
   For[i=1,i<=npoints,i++,
     (
       t=SessionTime[]–t0;
       x=N[CaGetValue[in]];
       mplot[[i]]={t,x};
       Print[{t,x}];
       Pause[interval]
      )
   ];

   ListPlot[mplot,PlotJoined–>True,Frame–>True,
      FrameLabel–>{"Time (Sec)", "Value",in," "}]
)
```

Function for Plotting Waveform Record

```
PlotWF::usage = "PlotWF[\"WFrecordname\"] \n
         WFrecordname – specifies the requested waveform \n
                  device name"

PlotWF[in_String] := (
      x= CaGetWF[in];
      Print[x];
      ListPlot[x,PlotJoined–>True]
)
```

Function for Creating and Plotting Waveform Record

```
sine::usage = "sine[\"devname\"] – set and plot the sine wave form\n"

sine[in_String]:=(
```

```
   npoints=CaGetCount[in];
   x=Table[0,{npoints}];
   For[i=1,i<=npoints,i++,
     (  ii=Pi*2/npoints*i;
        x[[i]]=N[Sin[ii]];
     )
   ];
   Print[x];
   CaPutWF[in, x];
   ListPlot[x,PlotJoined–>True,Frame–>True,
                FrameLabel–>{in, "Value","PLOT OF SINE WAVE "," "}]
)

cosine::usage = "cosine[\"devname\"] – set and plot the sine wave form\n"

cosine[in_String]:=(
   npoints=CaGetCount[in];    x=Table[0,{npoints}];
   For[i=1,i<=npoints,i++,
     (  ii=Pi*2/npoints*i;
        x[[i]]=N[Cos[ii]];
      )
   ];
   Print[x];
   CaPutWF[in, x];
   ListPlot[x,PlotJoined–>True,Frame–>True,
                FrameLabel–>{in, "Value","PLOT OF COSINE WAVE "," "}]
)

updown::usage = "updown[\"devname\"] – set and plot the sine wave form\n"

updown[in_String]:=(
   npoints=CaGetCount[in];    x=Table[0,{npoints}];
   For[i=1,i<=npoints,i++,
     (  x[[i]] = If[i>10,20–i,i];
        x[[i]] = If[i>30,i–40,x[[i]]];
      )
   ];
   Print[x];
   CaPutWF[in, x];
   ListPlot[x,PlotJoined–>True,Frame–>True,
                FrameLabel–>{in, "Value","PLOT OF COSINE WAVE "," "}]
)

downup::usage = "downup[\"devname\"] – set and plot the sine wave form\n"

downup[in_String]:=(
```

```
    npoints=CaGetCount[in];    x=Table[0,{npoints}];
    For[i=1,i<=npoints,i++,
      (  x[[i]] = If[i>10,i–20,–i];
         x[[i]] = If[i>30,40–i,x[[i]]];
      )
    ];
    Print[x];
    CaPutWF[in, x];
    ListPlot[x,PlotJoined–>True,Frame–>True,
                  FrameLabel–>{in, "Value","PLOT OF COSINE WAVE "," "}]
)
```

5. Timing of CaGet / CaPut in Mathematica

The average execution times in seconds for **CaGet** , **CaPut**, **CaGetString**, **CaPutString** are listed below.

CaGet	CaGetString	CaPut	CaPutString
0.016	0.017	0.009	0.007

6. CaMath Related Files

Below is a list of the related CaMath files required for generating the mathlink functions which can be linked into Mathematica. Detailed information on generating the mathlink external functions can be found in reference 3 .

CaMath Files

CaMath.c	The C source channel access functions to be linked into Mathematica.
CaMath.tm	The MathLink template file that specifies how functions in the C source code of the external program CaMath.c are to be associated with Mathematica functions.
makefile	The makefile used to create CaMath external link functions.
alarmString.h	The header file containing the alarm string status codes.
chandata.h	The header file containing the definition of data structure used in cafunc.c.
cafunc.o	The object library containing typical channel access function calls.
caevent.o	The object library containing the value monitoring function calls.
hash3.o	The object library containing the memory allocation and hash table generation for any given input device.

References

1 Wolfram, Stephen. *Mathematica: A System for Doing Mathematics by Computer.* Reading, MA: Addison–Wesley, 1991 (2nd edition).

2 Hill, Jeff. *Channel Access Reference Manual.* Technical Note AT8–89SYS001, LANL, September 1991.

3 Wolfram Research. *Mathematica Technical Report : MathLink External Communication in Mathematica.*