ARCSDLL version 2.1.2

(for Linux)

April 6, 2001

ARCSdll provides a way to directly access the functionality of the ARCS Lightning Board through user programs or third-party software. This document describes the installation procedure and the basic functionality.

Installation

1. Copy everything in the arcsdll subdirectory into a temporary directory.
2. In this temporary directory, examine the `Makefile`. If necessary, modify the home directory of the ARCS installation, $(ARCSHOME) (default: `/usr/local/arcs`).
3. Type `make`. The following files are copies into the appropriate directories:
   - **libarcscmd.so**: This file invokes the appropriate JAVA methods. It is placed under <ARCS_ROOT>/arcsdll
   - **arcscmd.jar** : This contains the ARCScmd application classes. It will be placed in <ARCS_ROOT>/lib
   - **demonstration files**: These files are placed in <ARCS_ROOT>/arcsdll:
     - arcsdll.h: Sample header file for functions that can be called in the user C program.
     - arcserror.h: Sample header file for error messages (for user C program).
     - testdll.c: An example C program.
     - sinewave.out: An example sinewave program that is called in testdll.c.

The following system variables are needed by ARCSdll (see the included bash script pathdef.sh in <ARCS_ROOT>/arcsdll for an example):

- ARCS_ROOT: This should be set to the ARCS root directory (default: `/usr/local/arcs`).
- JAVA_HOME: This should be set to the home directory of JRE or JDK (default: `/usr/local/jre118_v3`).
- LD_LIBRARY_PATH: This should include the following directories <JAVA_HOME>/lib/i686/native_threads:<ARCS_ROOT>/arcsdll:<ARCS_ROOT>/lib

As an example, see the included C program, `testdll.c`, at the end of this file. The header file arcserror.h and arcsdll.h would need to be modified for other languages. To build the example program, copy all files in <ARCS_ROOT>/arcsdll into a working directory, delete testdll.o, and type

```
make –ftestdll.mk
```

Function Description

Function name	Description
ArcsInitializeLocalHost ();	Initialize the local host
ArcsInitializeRemoteHost(char *hostName);	Initialize the remote host (remote host must have the HdwHostApp running)
ArcsLoad (const char *app);	Load an application
ArcsStart ();	Start running of application which is downloaded already
ArcsStop ();	Stop running of current application
ArcsReboot();	Reboot ARCS lightning board, also allows reconnection with a different Hardware Host
ArcsGetModuleNames(char *modName[], int *modLen);	Return module names which are available in current application
ArcsGetSymbolNames(char *mod, char *symName[], int *symLen);	Get symbol names in a module
ArcsSetSymbolValue(char *name,float *value, int *offset);	Set the value of the specified symbol. When the symbol is an array, the offset may be specified.
ArcsGetSymbolValue(char *name,int *offset, double *value);	Return a value of specified symbol
ArcsAddSymbolIntoDAQ(char *name, int *offset);	Add a specified symbol into list for data acquisition. The sequence of adding the symbol corresponds to the column number which is returned with ArcsGetDAQ command.
ArcsDelSymbolFromDAQ (char *name,int *offset);	Delete a specified symbol from the data acquisition list.
ArcsDelAllSymbolsFromDAQ ();	Delete all symbols from the data acquisition list.
ArcsGetDAQ(int *period, int *noPoints, int *channel, int *numT, int *numP, double value[]);	Start the data acquisition and return its result. period = # of sampling period between data acquisition noPoints = total # of points of data to collect channel = Interrupt channel (0 or 1) numT= numP= value =collected data
ArcsGetSymbolValueWithModule(char *name,char *module,int *offset, double *value);arcsinitremote	Display the available remote symbols which will be linked with local symbols.
ArcsFinish ();	Terminates the connection to ARCS DSP Board.
ArcsInitRemoteConnection(char *rName,char *rObj);	Initialize the remote node
ArcsStartRemoteConnection();	Start the remote connection
ArcsAddRemoteConnection(char *remoteSym,char *hostSym, int *offset);	Establish linkage between remote symbol and local symbol with optional offset.

ArcsGetRemoteConnection(char *remoteSyms[], int *remoteSymLen);	Get the remote symbols that are available for remote connection
ArcsGetHostSymbolForRemoteConnection(char *hostSyms[], int *hostSymLen);	Get the host symbols that are available for remote connection

Error Code

Number	Error Message	MACRO
0	No Error	NOERROR
1	Initialization has already been performed	ERR_REINITIALIZED
2	Java Virtual Machine cannot be created	ERR_NOTCREATE_JVM
3	Cannot find ARCSCMD.jar	ERR_NOTFIND_ARCSCMD
4	Initialization has failed (check arcsware.props file)	ERR_FAIL_INITIALIZATION
5	Hardware host has not been started	ERR_NOINITIALIZED
6	Error occurred during loading an application	ERR_LOADING
7	Cannot find specified application file	ERR_NOTFIND_APP
8	Application has not been loaded yet	ERR_NOTLOAD_APP
9	Application has already started	ERR_RESTART_APP
10	No application has been started yet	ERR_NOTSTART_APP
11	Symbol name cannot be found	ERR_WRONG_SYMBOL
12	Error occurred on setting value of symbol	ERR_SET_SYMBOL_VALUE
13	Error occurred on getting value of symbol	ERR_GET_SYMBOL_VALUE
14	Error occurred on getting module names	ERR_GET_MODULE_NAME
15	Error occurred on getting symbol names	ERR_GET_SYMBOL_NAME
16	Error occurred on adding symbol for data aquisition	ERR_ADD_SYMBOL_DATAACQ
17	Error occurred on removing symbol from data aquisition	ERR_REM_SYMBOL_DATAACQ
18	Too many points specified for data acquisition	ERR_TOO_MANY_POINTS
19	No Symbol is added for data acquisition	ERR_NO_SYMBOL_DATAACQ

Example Program (testdll.c)

```
#include <stdio.h>
#include <stdlib.h>

#include "arcsdll.h"
#include "arcserror.h"

int main ()
{
   int i,status;
   char sName[]   = "A";
   char sName1[]   = "sinewave.out";
   double value;
   double wave;
   double time;
   int offset = 0;
   float amp = 1.0;
   char *modName[50];
   char *symName[50];
   int  modLen,symLen;
      int ch;

   // System initialization.
   status = ArcsInitializeLocalHost ();
   if(status !=0)  { printf("   <<ARCS>>   %s \n",errorMsg[status]);
      exit(1);}
   status = ArcsLoad ("sinewave");
   if(status !=0)  { printf("   <<ARCS>>   %s \n",errorMsg[status]);
      exit(1);}


   // Illustration of getting application modules & symbols
   printf ("Available Modules:\n");
   status = ArcsGetModuleNames(modName,&modLen);
   if(status ==0)
   {
            printf("   <<ARCS>>   The available Modules are as
follows...\n");
            for(i=0;i<modLen;i++)
            {
                  if( (i%2) == 0) printf("\n              ");
                  printf("%-20s     ",modName[i]);
            }
            printf("\n\n");
      }
      else
      {
            printf("   <<ARCS>>   %s \n",errorMsg[status]);
            exit(1);
      }
      status = ArcsGetSymbolNames(modName[0],symName,&symLen);
      if(status==0)
      {
            printf("   <<ARCS>>   The available Symbols in <%s> are as
follows...\n",modName[0]);
```

```
            for(i=0;i<symLen;i++)
            {
                  if( (i%2) == 0) printf("\n              ");
                  printf("%-20s     ",symName[i]);
            }
            printf("\n\n");
      }
      else
      {
            printf("   <<ARCS>>   %s\n",errorMsg[status]);
            exit(1);
      }
   // Start the application now!!
      status = ArcsStart();
      if(status ==0)
            printf("   <<ARCS>>   current application is started...\n");
      else
            printf("   <<ARCS>>   %s\n",errorMsg[status]);


   ArcsStop();
   printf ("   Quitting!   ...  Bye!!\n\n");
}
```