Simple CA Client

- **Defines and includes**

```
/* Simple CA client */

#define TIMEOUT 1.0
#define SCA_OK 1
#define SCA_ERR 0
#define MAX_STRING 40

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

Simple CA Client

- **Function prototypes and global variables**

```
/* Function prototypes */
int main(int argc, char **argv);
static int parseCommand(int argc, char **argv);
static void usage(void);

/* Global variables */
int pvSpecified=0;
char name[MAX_STRING];
char value[MAX_STRING];
double timeout=TIMEOUT;
```

Simple CA Client

- Parse the command line

```
int main(int argc, char **argv)
{
    int stat;
    chid pCh;

  /* Parse the command line */
    if(parseCommand(argc,argv) != SCA_OK) exit(1);
    if(!pvSpecified) {
        printf("No PV specified\n");
        exit(1);
    }
```

Simple CA Client

- **Initialize Channel Access**

```
/* Initialize */
  stat=ca_context_create(ca_disable_preemptive_callback);
  if(stat != ECA_NORMAL) {
      printf("ca_context_createfailed:\n%s\n",
        ca_message(stat));
      exit(1);
  }
```

Simple CA Client

- **Request the search**

```
/* Search */
  stat=ca_create_channel(name,NULL,NULL,
    CA_PRIORITY_DEFAULT,&pCh);
  if(stat != ECA_NORMAL) {
      printf("ca_create_channel failed:\n%s\n",
        ca_message(stat));
      exit(1);
  }
```

Simple CA Client

- **Call ca-pend_io to process the search**

```
/* Process search */
  stat=ca_pend_io(timeout);
  if(stat != ECA_NORMAL) {
      printf(“search timed out after %g sec\n",
        timeout);
      exit(1);
  }
```

Simple CA Client

- **Request the get**

```
/* Get the value */
  stat=ca_array_get(DBR_STRING,1,pCh,&value);
  if(stat != ECA_NORMAL) {
      printf("ca_array_get:\n%s\n",
        ca_message(stat));
      exit(1);
  }
```

Simple CA Client

- **Call ca-pend_io to process the get**

```
/* Process get */
  stat=ca_pend_io(timeout);
  if(stat != ECA_NORMAL) {
      printf(“get timed out after %g sec\n",
        timeout);
      exit(1);
  }
  printf("The value of %s is %s\n",name,value)
```

Simple CA Client

- Clean up

```
  /* Clear the channel */
    stat=ca_clear_channel(pCh);
    if(stat != ECA_NORMAL) {
       printf("ca_clear_channel failed:\n%s\n",
         ca_message(stat));
    }

  /* Exit */
    ca_context_destroy();
    return(0);
}
```

Simple CA Client

- **Output**

```
simplecaget evans:calc
The value of evans:calc is 6
```

Simple CA Client with Callbacks

- **Defines and includes**

```
/* Simple CA client with Callbacks */

#define TIMEOUT 1.0
#define SCA_OK 1
#define SCA_ERR 0
#define MAX_STRING 40

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>
#include <cadef.h>
```

Simple CA Client with Callbacks

- **Function prototypes**

```
/* Function prototypes */
int main(int argc, char **argv);
static void connectionChangedCB(struct
  connection_handler_args args);
static void valueChangedCB(struct event_handler_args
  args);
static char *timeStamp(void);
static int parseCommand(int argc, char **argv);
static void usage(void);
```

Simple CA Client with Callbacks

- **Global variables**

```
/* Global variables */
int pvSpecified=0;
char name[MAX_STRING];
time_t curTime, startTime;
double timeout=TIMEOUT;
```

Simple CA Client with Callbacks

- Parse the command line

```
int main(int argc, char **argv)
{
    int stat;
    chid pCh;

  /* Parse the command line */
    if(parseCommand(argc,argv) != SCA_OK) exit(1);
    if(!pvSpecified) {
        printf("No PV specified\n");
        exit(1);
    }
```

Simple CA Client with Callbacks

- **Initialize Channel Access**

```
/* Initialize */
  stat=ca_context_create(ca_disable_preemptive_callback);
  if(stat != ECA_NORMAL) {
      printf("ca_context_createfailed:\n%s\n",
        ca_message(stat));
      exit(1);
  }
```

Simple CA Client with Callbacks

- Search

```
/* Search */
  stat=ca_create_channel(name,connectionChangedCB,NULL,
    CA_PRIORITY_DEFAULT,&pCh);
  if(stat != ECA_NORMAL) {
      printf("ca_create_channel failed:\n%s\n",
        ca_message(stat));
      exit(1);
  }
  printf("%s Search started for %s\n",timeStamp(),name);
```

Simple CA Client with Callbacks

- **Wait in ca_pend_event for the callbacks to occur**

```
/* Wait */
  startTime=curTime;
  ca_pend_event(timeout);
  printf("%s ca_pend_event timed out after %g sec\n",
    timeStamp(),timeout);
```

Simple CA Client with Callbacks

- Clean up

```
  /* Clear the channel */
    stat=ca_clear_channel(pCh);
    if(stat != ECA_NORMAL) {
       printf("ca_clear_channel failed:\n%s\n",
         ca_message(stat));
    }

  /* Exit */
    ca_context_destroy();
    return(0);
}
```

Simple CA Client with Callbacks

- **Connection callback implementation**

```
static void connectionChangedCB(struct
   connection_handler_args args)
{
    chid pCh=args.chid;
    int stat;

  /* Branch depending on the state */
    switch(ca_state(pCh)) {
```

Simple CA Client with Callbacks

- **Connection callback implementation**

```
case cs_conn:
    printf("%s Connection successful\n",timeStamp());
    stat=ca_array_get_callback(DBR_STRING,1,pCh,
      valueChangedCB,NULL);
    if(stat != ECA_NORMAL) {
        printf("ca_array_get_callback:\n%s\n",
          ca_message(stat));
        exit(1);
    }
    break;
```

Simple CA Client with Callbacks

- **Connection callback implementation**

```
    case cs_never_conn:
        printf("%s Cannot connect\n",timeStamp());
        break;
    case cs_prev_conn:
        printf("%s Lost connection\n",timeStamp());
        break;
    case cs_closed:
        printf("%s Connection closed\n",timeStamp());
        break;
    }
}
```

Simple CA Client with Callbacks

- Value changed callback implementation

```
static void valueChangedCB(struct event_handler_args args)
{
  /* Print the value */
    if(args.status == ECA_NORMAL && args.dbr) {
        printf("%s Value is: %s\n",timeStamp(),
          (char *)args.dbr);
        printf("Elapsed time: %ld sec\n",
          curTime-startTime);
    }
}
```

Simple CA Client with Callbacks

- **Output**

```
simplecagetcb evans:calc
Sep 14 18:31:55 Search started for evans:calc
Sep 14 18:31:55 Connection successful
Sep 14 18:31:55 Value is: 5
Elapsed time: 0 sec
Sep 14 18:31:56 ca_pend_event timed out after 1 sec
```

- **Time for this operation is typically a few ms**

Source files for Simple Get Clients

- **Some of the code that is not related to Channel Access has not been shown**
- **All the files necessary to build a project as an EPICS Extension should be available with the presentation**
  - Makefile
  - Makefile.Host
  - simplecaget.c
  - simplecagetcb.c
  - LICENSE
- **Stored as simpleCA.tar.gz**

EPICS Build System

- **Supports both native and GNU compilers**
- **Builds multiple types of components**
  - libraries, executables, headers, scripts, java classes, …
- **Supports multiple host and target operating systems**
- **Builds for all hosts and targets in a single <top> tree**
  - epics/base
  - epics/extensions
- **Allows sharing of components across <top> trees**
- **Has different rules and syntax for 3.13 and 3.14**

System Requirements

- **Required software**
  - Perl version 5 or greater
  - GNU make, version 3.78.1 or greater
  - C++ compiler and linker (GNU or host vendor's compiler)
- **Optional software**
  - Tornado II and board support packages
  - RTEMS development tools and libraries
  - Motif, X11, JAVA, TK/TCL…

User Requirements

- **Set an environment variable to specify the architecture**
  - EPICS_HOST_ARCH for 3.14
    - *solaris-sparc, linux-x86, win32-x86, darwin-ppc, etc.*
  - HOST_ARCH for 3.13
    - *solaris, Linux, WIN32, etc.*
- **Set the PATH so the required components can be found**
  - Perl, GNU make, C and C++ compilers
  - System commands (e.q. cp, rm, mkdir)

Typical Extensions Build Tree

epics/base	<top> for base
epics/extensions	<top> for extensions
config	3.13 configuration
configure	3.14 configuration
bin	Binaries by architecture
solaris	
solaris-sparc	
lib	Libraries by architecture
solaris	
solaris-sparc	
src	Sources by application
simpleCA	Application source files
O.solaris	Binaries for this application
O.solaris-sparc	

Getting Started with an Extension

- **Make a directory structure for base**
  - E.g. epics/base
- **Obtain base and build it**
  - Set COMPAT_TOOLS_313 first if necessary (see later)
- **Make a directory structure for extensions**
  - E.g. epics/extensions
- **Get extensions/config and configure from the EPICS pages**
  - http://www.aps.anl.gov/epics/extensions/index.php
- **Set EPICS_BASE to your desired version of base**
  - In extensions/config/RELEASE for 3.13
  - In extensions/configure/RELEASE for 3.14
- **Type gnumake (or make) in extensions**
- **Get an extension and put it under extensions/src**
- **Type gnumake (or make) in your application directory**

Using the 3.13 Build Rules for Extensions

- **Most existing extensions are still set up for 3.13 builds**
  - There is a Makefile and a Makefile.Host
  - Makefile.Host is most important and has 3.13 syntax
  - Can still use a 3.14 base
- **Set HOST_ARCH for your platform**
  - Solaris, Linux, WIN32, etc.
- **Set EPICS_HOST_ARCH for your platform**
  - solaris-sparc, linux-x86, win32-x86, darwin-ppc, etc.
- **Configuration is in extensions/config**
  - RELEASE (Specifies what base to use, can be 3.14)
  - CONFIG_SITE_xxx (Specifies local changes for xxx arch)
- **Before building a 3.14 base**
  - Modify base/configure/CONFIG_SITE
    - *COMPAT_TOOLS_313 = YES*

Using the 3.14 Build Rules for Extensions

- **Go to the the EPICS page for your version of base**
  - http://www.aps.anl.gov/epics/base/index.php
- **Read the README**
  - It is very extensive
  - Should tell you everything you need to know
- **There is a only a Makefile and it uses 3.14 syntax**
- **Set EPICS_HOST_ARCH for your platform**
  - solaris-sparc, linux-x86, win32-x86, darwin-ppc, etc.
- **Configuration is in extensions/configure**
  - RELEASE (Specifies what base)
  - os/CONFIG_SITE_xxx (Specifies local changes for xxx arch)

Makefile for Simple Get Clients

```
TOP = ../..
include $(TOP)/config/CONFIG_EXTENSIONS
include $(TOP)/config/RULES_ARCHS
```

Makefile.Host for Simple Get Clients

```
TOP = ../../..
include $(TOP)/config/CONFIG_EXTENSIONS

HOST_OPT = NO
CMPLR = STRICT

PROD = simplecaget simplecagetcb

PROD_LIBS = ca Com
ca_DIR = $(EPICS_BASE_LIB)
Com_DIR = $(EPICS_BASE_LIB)

simplecaget_SRCS += simplecaget.c
simplecagetcb_SRCS += simplecagetcb.c

include $(TOP)/config/RULES.Host
```

Acknowledgements

- **Jeff Hill [LANL] is responsible for EPICS Channel Access and has developed almost all of it himself**
- **Janet Anderson [ANL] is responsible for and has developed most of the EPICS Build System**

Thank You

This has been an

APS Controls Presentation