RMT & CCE porting to *NIX

by A.Kazakov, July-August 2007

Current State

The original RMT & CCE source code has been ported to epics' OSI environment. And virtually would compile (and hopefully work) under any epics supported system.
Main functionality proved to work on **vxWorks, Darwin (Mac OS X), Linux.**
Also it was successfully built on **Solaris 5.8** and run. But no further testing.
Some new functions were added to libCom/osi. Please refer to **EPICS base modifications** section for details.

Present problems

It seems that there are might be some problems. At least on Linux. It was seen to crash a few times. Though some modification was made since then. And no crush report yet.
Anyway it seems that there might be some critical error in the code. I've been keeping the system running for days (a week at most) without a crash. In my tests there were around 100 records in the database and one client (dm2k) constantly connected to the system. I suppose more "real world" testing is needed.
One more problem on Linux exists: from time to time there is a connectivity problem on the initial stage between Master and Slave. You could see "BAD DESCRIPTOR ERROR". The origin of the error is still unknown. It was seen only on Linux. After the connection has been successfully established I've never seen this error. So it may appear only on the phase of the start-up of one of the servers (or after a major network problem probably, when they reopen sockets etc).
I hope that no new bugs were introduced during the porting process.

When testing the system with relatively large database (~3000+ records) I've seen error like this:

```
Redundant Update fallen behind by 42 ticks.
```

Which means that CCE could not finish regular update cycle within 60 ticks (this parameter DEFAULTCCETHROTTLE could be changed in cce.c). It is not a critical error, more like informational message. Though it shows that system is working around its speed limit (~5000 records/second for 2 linux machines with 3GHz P4 1core, 2x 100Mbit ethernet cards; doing nothing but redundant IOC). The database I used is 1x30 calc records, and 100x30 ai records, connected to these corresponding calc records. All the records are "scan 1 second" and calc record is changing from 0 to 9 every second.

Problems fixed during porting

There were several NULL pointer references in the code and breaking the array boundaries errors. These errors led to crash on Darwin and Linux machines, but surprisingly seemed to work on vxWorks.
CCE's "CACHE FULL" Error behavior was changed. Now when such an error occurs the CceTraverse thread is suspended, and CCE driver is put into INVALID_ERROR mode. This error could be seen if you have more than 10000 records. Previous behavior just printed an error indicating that, and than the update function was called with the wrong pointer value equal -1. This caused the application to crash (on *unix, I have not tested big database on vxWorks).

Source files location

I've divided the original RMT & CCE Test application (called cceApp) into

- rmtLib - rmt basic functionality. Consists from **rmt.c** and headers.
- cceLib - cce functionality and mxt and other xml stuff (sorry not familiar with it). **cce.c mxt.c xml*.c** and headers
- everything else you can put into some other dirctory. And edit Makefile to plug in rmt and cce libraries. See next section for details.

How to build an application using RMT (and CCE) libs?

nothing is easier now.

1. create your application directory structure (using **makeBaseApp.pl** script or by hands if you are such a guru :)) Let's say you call it **rmtHello**. So inside your newly created directory you should have **rmtHelloApp** directory.
2. put rmtLib (and cceLib) into this directory
3. add these lines to the **Makefile**
   - `DIRS := $(DIRS) rmtLib`
   - `DIRS := $(DIRS) cceLib`
4. Now your **Makefiles** should look like this:

```
#Makefile at top of application tree
TOP = .
include $(TOP)/configure/CONFIG
DIRS := $(DIRS) $(filter-out $(DIRS), configure)
DIRS := $(DIRS) rmtLib
DIRS := $(DIRS) cceLib
DIRS := $(DIRS) $(filter-out $(DIRS), $(wildcard *App))
DIRS := $(DIRS) $(filter-out $(DIRS), $(wildcard *app))
DIRS := $(DIRS) $(filter-out $(DIRS), $(wildcard iocBoot))
DIRS := $(DIRS) $(filter-out $(DIRS), $(wildcard iocboot))
include $(TOP)/configure/RULES_TOP
```

5. Now edit **rmtHello/(src)/Makefile:**
   - add **rmt.dbd (and cce.dbd)** to *rmtHello_DBD* definition
   - add **rmt** and **cce** to *rmtHello_LIBS*
6. That is it! Now build and run an RMT (and CCE) aware application.

MISC notes on configure/*

To build modified epicBase and RMT on Linux you need:

```
OSITHREAD_USE_DEFAULT_STACK = YES
OP_SYS_CFLAGS += -rdynamic
OP_SYS_LDFLAGS += -export-dynamic
OP_SYS_LDLIBS += -ldl
```

For darwin:

```
OSITHREAD_USE_DEFAULT_STACK = YES
```

For solaris:

```
OP_SYS_LDLIBS += -ldl
```

For vxWorks no special configuration (except needed by default like compiler paths etc)

EPICS base modifications

introduced by A.Kazakov while porting RMT to *unix platform.

libCom/osi

several new functions were introduced. Only C-interfaces were implemented. Here they are:

- **epicsMutex.h:** epicsMutexLockStatus epicsShareAPI epicsMutexLockWithTimeout (epicsMutexId id, double timeout);
- **epicsMutex.h:** epicsMutexLockStatus epicsMutexOsdLockWithTimeout(struct epicsMutexOSD *, double);
- **epicsThread.h:** epicsShareFunc int epicsShareAPI epicsThreadDelete(epicsThreadId id);
- **epicsTime.h:** epicsShareFunc clock_t epicsShareAPI epicsTimeGetTicks ();

Implementation Details epicsMutexLockWithTimeout(epicsMutexId id, double timeout): on **posix** systems there are two implementations of epicsMutexes. One is actually using pthread_mutexes, and the other is epics' self-made approach, though it still uses pthread_mutexes. If PTHREAD_MUTEX_RECURSIVE is supported then the former approach is used, otherwise the latter one. As far as Mac OS X is claimed to support PTHREAD_MUTEX_RECURSIVE, I modified the line where this checking is done to reflect PTHREAD_MUTEX_RECURSIVE ability:
#if defined(_XOPEN_SOURCE) && (_XOPEN_SOURCE)>=500 ll defined(darwin)

And as far as Linux and darwin (after my modification) use the "pthread only" approach I did not bother with the other one and added support only for "pthread only" version.

The main idea was kindly presented by Andrew Johnson. Basically we use pthread_condition_t variable, which we signal on MutexUnlock. When we try to lock mutex in MutexLockWithTimeout, first we do tryMutexLock and if it fails we wait on condition variable until it is signaled or timeout expires.

epicsMutexOSD structure was extended with these two variables:

```
pthread_cond_t      cond;
pthread_mutex_t     timeLock;
```

on epicsMutexOsdCreate we initialize them:

```
status = pthread_mutex_init(&pmutex->timeLock,0);
checkStatusQuit(status,"pthread_mutex_init","epicsMutexOsdCreate");
status = pthread_cond_init(&pmutex->cond,0);
checkStatusQuit(status,"pthread_cond_init","epicsMutexOsdCreate");
```

on epicsMutexOsdLockWithTimeout we do:

```
int status, unlockStatus;
pthread_mutex_t *id = &pmutex->lock;
struct timespec  wakeTime,now;
convertDoubleToWakeTime(timeout,&wakeTime);

while(1) {
status = pthread_mutex_trylock(id);
if(status == 0) return (epicsMutexLockOK);
if (status == EBUSY) {
 status = mutexLock(&pmutex->timeLock);
```

```
  checkStatusQuit( status, "phtread_mutex_lock",
                           "epictMutexOsdLockWithTimeout");
  status = pthread_cond_timedwait(&pmutex->cond,
                           &pmutex->timeLock,&wakeTime);
  unlockStatus = pthread_mutex_unlock(&pmutex->timeLock);
  checkStatusQuit(unlockStatus,"pthread_mutex_unlock",
                           "epicsMutexOsdLockWithTimeout");
  if(status == ETIMEDOUT) break;
  checkStatusQuit(status,"pthread_cond_timedwait",
                           "epicsMutexOsdLockWithTimeout");
 } else
  checkStatusQuit(status,"pthread_mutex_trylock",
                           "epicsMutexOsdLockWithTimeout");
}
return epicsMutexLockTimeout;
```

and on epicsMutexOsdUnlock we wake one of the waiting threads:

```
    status = pthread_cond_signal(&pmutex->cond);
    checkStatusQuit(status,"pthread_cond_signal","epicsMutexOsdUnlock");
```

on **vxWorks** simply:

```
epicsMutexOsdLockWithTimeout(struct epicsMutexOSD * id,
                          double timeout)
{
 int status;
 status  = semTake((SEM_ID)id, timeout*sysClkRateGet());
 if(status==OK) return (epicsMutexLockOK);
 if(errno==S_objLib_OBJ_UNAVAILABLE) return(epicsMutexLockTimeout);
 return(epicsMutexLockError);
}
```

Implementation details epicsThreadDelete(epicsThreadId id):
posix:

```
 return pthread_cancel(id->tid);
```

vxWorks:

```
 return (taskDelete(id));
```

Implementation details epicsTimeGetTicks():
posix:

```
 struct tms tp;
 return (times(&tp));
```

vxWorks:

```
return (tickGet());
```

changes for CCE

introduced by G.Liu and others

with red color I selected the code being added.

- **base/src/db/dbAccess.c**

```
long epicsShareAPI dbPutSpecial(DBADDR *paddr,int pass)[1]
{
long        status;
status = putSpecial(paddr,pass);
```

---

1 with red color I highlighted the code being added

```
return status;
}
```

- **base/src/db/dbScan.c**

typdef and prototypes:

```
typedef struct {
        BOOL            run;            /* scan task is running */
        BOOL            activeFlag;     /* flag to check if loop is running */
        const char*     instanceName;
        epicsThreadId             tid;            /* task id of driver instance
*/
} scanPrivateType;

static STATUS scanStart(scanPrivateType *ppvt);
static STATUS scanStop(scanPrivateType *ppvt);
static STATUS scanGetStatus(scanPrivateType *ppvt, drvStatusType *pstatus);
static STATUS scanGetInfo(scanPrivateType *ppvt, char *pString, short *psize,
char *prequestString);
```

register “scan driver in rmt”:

```
    /* added by Gongfa Liu */
    /* ------------------------------------------------*/
    rmtEntryTabType   scanRegistry;
    scanPrivateType   scanPvt;

    const rmtInfoTabType  *info;
    long (*prmtRegister)() = NULL;
    long status;

    scanRegistry.type                = "ScanPeriodic";
    scanRegistry.instanceName        = epicsThreadGetNameSelf();
    scanRegistry.testTimeTypical     = 0;
    scanRegistry.pPrvt               = (void *)&scanPvt;
    scanRegistry.pStart              = (RMTSUPFUN)scanStart;
    scanRegistry.pStop               = (RMTSUPFUN)scanStop;
    scanRegistry.pTestIO             = NULL;
    scanRegistry.pGetStatus          = (RMTSUPFUN)scanGetStatus;
    scanRegistry.pShutdown           = NULL;
    scanRegistry.pGetInfo            = (RMTSUPFUN)scanGetInfo;
    scanRegistry.pGetUpdate          = NULL;
    scanRegistry.pStartUpdate        = NULL;
    scanRegistry.pStopUpdate         = NULL;

    scanPvt.instanceName        = scanRegistry.instanceName;
    scanPvt.tid                 = epicsThreadGetIdSelf();
    scanPvt.run                 = FALSE;
    scanPvt.activeFlag          = TRUE; /* always TRUE when no testIO */

    /*rmtRegister(&scanRegistry, NULL); */

    prmtRegister = epicsFindSymbol("rmtRegister");
    if(prmtRegister == NULL)
    {
      printf("Not redundance IOC!!!\n");
      scanPvt.run = TRUE; /* let scan task be active */
    }
    else
    {
      status = (*prmtRegister)(&scanRegistry, NULL);
    }
    /* ------------------------------------------------*/
```

scan driver “rmt interface” function implementations:

```
/* added by Gongfa Liu */
/* ------------------------------------------------------*/
```

```
static STATUS scanStart(scanPrivateType *ppvt)
{
        ppvt->run = TRUE;
printf("scanStart: ppvt->run=%d\n", ppvt->run);
        return (OK);
}

static STATUS scanStop(scanPrivateType *ppvt)
{
        ppvt->run=FALSE;

printf("scanStop: ppvt->run=%d\n", ppvt->run);
        return (OK);
}

static STATUS scanGetStatus(scanPrivateType *ppvt, drvStatusType *pstatus)
{
        pstatus->mode = ppvt->run ? MODE_run : MODE_stop;
        pstatus->testResult = TEST_undefined;

        if ( epicsThreadIsSuspended(ppvt->tid) )
        {
            pstatus->error = INVALID_ERROR;
            /*pstatus->mode = MODE_stop;* set again when the task is suspended
*/
        }
        else
            pstatus->error = NO_ERROR;

        pstatus->updateBusy = FALSE;
        pstatus->inSync = TRUE;
        pstatus->activeFlag = ppvt->activeFlag;
        /* ppvt->activeFlag = FALSE;  */ /* activeFlag shuold be kept TRUE when
no TestIO */

        return OK;
}

static STATUS scanGetInfo(scanPrivateType *ppvt, char *pString, short *psize,
char *prequestString)
{
        char scanTaskInfo[100];

        sprintf(scanTaskInfo,"<XML><NAME>%s</NAME><STATUS>%s</STATUS></XML>",
          ppvt->instanceName, ppvt->run ? "active" : "inactive");

        if(prequestString && strlen(prequestString))
                printf("%s: scanGetInfo with request\n", ppvt->instanceName);
        else
                printf("%s: scanGetInfo without request\n", ppvt->instanceName);

        if (psize != NULL && pString != NULL)
  {
                if (*psize > sizeof(scanTaskInfo))
                        strcpy (pString, scanTaskInfo);
                else
                        *psize = sizeof(scanTaskInfo)+1;

                printf("%s\n", scanTaskInfo);
                return OK;
        }
        else
                return ERROR;
}
/* ----------------------------------------------------------------------*/
```

- **base/src/dbStatic/dbBase.h**

```
/* redundant ioc update mode */
typedef enum {RED_NO_UPDATE,RED_INIT_COPY,RED_CONT_UPDATE} redUp;

typedef struct dbFldDes{
...
        redUp   red_update;     /*redundant ioc update mode    */
...
}dbFldDes;
```

- **base/src/dbStatic/dbLexRoutines.c**

```
static void dbRecordtypeFieldHead(char *name,char *type){
...
    pdbFldDes->red_update = RED_CONT_UPDATE;
...
}

static void dbRecordtypeFieldItem(char *name,char *value)
{
...
if(strcmp(name,"red_update")==0) {
            if(strcmp(value,"NO_UPDATE")==0) {
                    pdbFldDes->red_update = RED_NO_UPDATE;
            } else if(strcmp(value,"INIT_COPY")==0) {
                    pdbFldDes->red_update = RED_INIT_COPY;
            } else if(strcmp(value,"CONT_UPDATE")==0) {
                    pdbFldDes->red_update = RED_CONT_UPDATE;
            } else {
                    yyerror("Illegal value. Must be NO_UPDATE, INIT_COPY or
CONT_UPDATE");
            }
    }
}
```

- **base/src/rec/aiRecord.dbd**

```
 recordtype(ai) {
        include "dbCommon.dbd"
        field(VAL,DBF_DOUBLE) {
...
                promptgroup(GUI_INPUTS)
                asl(ASL0)
                pp(TRUE)
                red_update(NO_UPDATE)
        }
```

CA Server

- **base/src/rsrv/camsgtask.c**

```
extern int masterFlagForCAS; /* Added by Gongfa Liu */

 /*
  *  camsgtask()
  */
...
            epicsPrintf ("CAS: forcing disconnect from %s\n", buf);
                break;
        }
```

```
       if(masterFlagForCAS == FALSE) client->disconnect = TRUE; /* added by
Gongfa Liu */
}
```

- **base/src/rsrv/online_notify.c**

```
extern int masterFlagForCAS; /* Added by Gongfa Liu */
void rsrv_online_notify_task(void *pParm)
{
...
      beaconCounter++; /* expected to overflow */

      while (masterFlagForCAS == FALSE) epicsThreadSleep(0.1); /* wait here when
slave, by Gongfa Liu */
      }
}
```

- **base/src/rsrv/cast_server.c**

```
extern int masterFlagForCAS; /* Added by Gongfa Liu */
void cast_server(void *pParm){
...
             cas_send_dg_msg (prsrv_cast_client);
             clean_addrq ();
         }
   while (masterFlagForCAS == FALSE) epicsThreadSleep(0.1); /* wait here when
slave, by Gongfa Liu */
      }
 }
```

- **base/src/rsrv/caservertask.c**

typedefs and prototypes:

```
/* added by Gongfa Liu */
/* ------------------------------------------------*/
#include "epicsFindSymbol.h"
#include "rmtDrvIf.h"
#ifndef vxWorks
#define BOOL int
#define LOCAL static
#define IMPORT extern
#define ERROR -1
#define TRUE 1
#define FALSE 0
#define OK 0
#endif


typedef struct {
        BOOL            run;            /* scan task is running */
        BOOL            activeFlag;     /* flag to check if loop is running */
        const char     *instanceName;
        epicsThreadId             tid;            /* task id of driver instance
*/
} castcpPrivateType;

int masterFlagForCAS = FALSE;

static STATUS castcpStart(castcpPrivateType *ppvt);
static STATUS castcpStop(castcpPrivateType *ppvt);
static STATUS castcpGetStatus(castcpPrivateType *ppvt, drvStatusType *pstatus);
static STATUS castcpGetInfo(castcpPrivateType *ppvt, char *pString, short
*psize, char *prequestString);
```

```
static STATUS castcpStartUpdate(castcpPrivateType *ppvt, updateMode mode);
/* ------------------------------------------------------*/
```

register “CA Server driver” with rmt:

```
LOCAL void req_server (void *pParm)
{
    ...
    /* added by Gongfa Liu */
    /* ------------------------------------------------*/
    rmtEntryTabType   castcpRegistry;
    castcpPrivateType   castcpPvt;

    const rmtInfoTabType  *info;
    long (*prmtRegister)() = NULL;
    /*SYM_TYPE symType;*/
    long status1;

    castcpRegistry.type                = "cas-tcp";
    castcpRegistry.instanceName        = epicsThreadGetNameSelf();
    castcpRegistry.testTimeTypical     = 0;
    castcpRegistry.pPrvt               = (void *)&castcpPvt;
    castcpRegistry.pStart              = (RMTSUPFUN)castcpStart;
    castcpRegistry.pStop               = (RMTSUPFUN)castcpStop;
    castcpRegistry.pTestIO             = NULL;
    castcpRegistry.pGetStatus          = (RMTSUPFUN)castcpGetStatus;
    castcpRegistry.pShutdown           = NULL;
    castcpRegistry.pGetInfo            = (RMTSUPFUN)castcpGetInfo;
    castcpRegistry.pGetUpdate          = NULL;
    castcpRegistry.pStartUpdate        = (RMTSUPFUN)castcpStartUpdate;
 castcpRegistry.pStopUpdate         = NULL;

    castcpPvt.instanceName        = castcpRegistry.instanceName;
    castcpPvt.tid                 = epicsThreadGetIdSelf();
    castcpPvt.run                 = FALSE;
    castcpPvt.activeFlag          = TRUE; /* always TRUE when no testIO */

   /* status1 = symFindByNameEPICS(sysSymTbl,"_rmtRegister",(char **)
&prmtRegister,&symType);
    if(status1 || symType != (SYM_GLOBAL|SYM_TEXT))
    */
    prmtRegister = epicsFindSymbol("rmtRegister");
    if(prmtRegister == NULL)
    {
            printf("Not redundance IOC!!!\n");
            masterFlagForCAS = TRUE; /* let flag is TRUE */
    }
    else
            status1 = (*prmtRegister)(&castcpRegistry, &info);
    /* ------------------------------------------------*/

...
 while (masterFlagForCAS == FALSE) epicsThreadSleep(0.1); /* wait here when
slave, by Gongfa Liu */
    }
}
```

rmt driver interface functions implementation:

```
/* added by Gongfa Liu */
/* ----------------------------------------------------------*/
static STATUS castcpStart(castcpPrivateType *ppvt)
{
        printf("%s: castcpStart\n", ppvt->instanceName);

        ppvt->run = TRUE;
        masterFlagForCAS = TRUE;
```

```
        return(OK);
}

static STATUS castcpStop(castcpPrivateType *ppvt)
{
        printf("%s: castcpStop\n", ppvt->instanceName);

        ppvt->run = FALSE;
        masterFlagForCAS = FALSE;

        return(OK);
}

static STATUS castcpGetStatus(castcpPrivateType *ppvt, drvStatusType *pstatus)
{
        pstatus->mode = ppvt->run ? MODE_run : MODE_stop;
        pstatus->testResult = TEST_undefined;

        if(  epicsThreadIsSuspended(ppvt->tid) )
                pstatus->error = INVALID_ERROR;
        else
                pstatus->error = NO_ERROR;

        pstatus->updateBusy = FALSE;
        pstatus->inSync = TRUE;
        pstatus->activeFlag = ppvt->activeFlag;
        /* ppvt->activeFlag = FALSE;  */ /* activeFlag shuold be kept TRUE when
no TestIO */

        return(OK);
}

static STATUS castcpGetInfo(castcpPrivateType *ppvt, char *pString, short
*psize, char *prequestString)
{
        char castcpTaskInfo[100];

        sprintf(castcpTaskInfo,"<XML><NAME>%s</NAME><STATUS>%s</STATUS></XML>",
                        ppvt->instanceName, ppvt->run ? "active" : "inactive");

        if(prequestString && strlen(prequestString))
                printf("%s: castcpGetInfo with request\n", ppvt->instanceName);
        else
                printf("%s: castcpGetInfo without request\n", ppvt-
>instanceName);

if (psize != NULL && pString != NULL)
        {
                if (*psize > sizeof(castcpTaskInfo))
                        strcpy (pString, castcpTaskInfo);
                else
                        *psize = sizeof(castcpTaskInfo)+1;

                printf("%s\n", castcpTaskInfo);
                return(OK);
        }
        else
                return(ERROR);
}
static STATUS castcpStartUpdate(castcpPrivateType *ppvt, updateMode mode)
{
        printf("%s: castcpStartUpdate\n", ppvt->instanceName);

        masterFlagForCAS = FALSE;
```

```
        if (ppvt->run) return(ERROR);
        else return(OK);
}
/* ------------------------------------------------------------------*/
```