RAON Control-Document Series
Revision : v1.0, Release : 2014-12-24 fixed date

Timing System을 위한 RTEMS Realtime OS 구성

이상일*
Rare Isotope Science Project
Institute for Basic Science, Daejeon, South Korea

December 17, 2014

Abstract

RAON accelerator는 대형 실험 장치로 많은 실험 장치들과 부대시설 장치로 구성되어 운영된다. 이러한 많은 실험 장치들을 제어하기 위한 제어 시스템들은 넓은 범위로 분산되어 구성되어 있으며 이러한 분산 환경에서의 많은 제어시스템들을 전체의 하나의 제어시스템으로 운영하기 위하여 RAON control system은 EPICS software framework을 사용한다. 또한 이러한 분산 시스템이 하나의 정밀한 제어 시스템으로 운영되기 위하여 가장 필요한 요소는 timing 시스템이다. 현재 timing 시스템에서 사용되는 VME system 상에서 운영되는 realtime OS로는 vxWorks 및 RTEMS가 사용되며, 본 문서에서는 VME system 상에서 운영될 RTEMS realtime OS에 대한 개발 환경 구성에 대하여 상세히 설명한다.

Timing 시스템은 크게 두 가지의 목적으로 사용된다. 하나는 분산된 제어 장치들의 정밀한 동기화 운전을 위하여 동기화된 event trigger 신호를 발생하기 위한 것이고, 다른 하나는 분산 제어 장치들에 대한 시각 동기화를 위한 것이다. Timing 시스템에 대한 자세한 내용은 참고 문헌을 활용하기로 하며, 본 문서에서는 Timing 시스템을 운영하기 위한 또다른 실시간 운영체제인 RTEMS를 사용하기 위한 환경 구성 및 RTEMS 상에서 EPICS IOC 개발 환경에 대하여 다루도록 한다.

1 Timing System

1.1 개요

RAON 제어 시스템에서 사용되는 timing system은 MRF(Micro Research Finland Oy) 사의 EVG/EVR[1] 시스템을 사용한다. 이 시스템은 VME 또는 PCI interface를 지원한

*silee7103@ibs.re.kr

다. RAON 제어 시스템은 VME interface 상에서 EVG/EVR 타이밍 시스템을 사용한다. VME controller는 Emerson사의 MVME6100[2] 또는 MVME3100[3] 사용하며, VME crate를 사용하여 구성한다. Timing 시스템은 Timing 모듈과 네트워크로 구성되어 있다.

1.2 이슈 사항

Timing 시스템에서 사용하는 realtime OS는 VxWorks를 사용한다. 이는 commercial realtime OS로 robust한 응답성을 요구하는 시스템에 주로 사용된다. VxWorks는 고속철도, 국방, 항공과 관련한 분야에서 많이 사용되고 있다. 이러한 많은 사용층 및 레퍼런스가 있음에도 자유롭게 사용하지 못하는 이유는 소프트웨어 사용에 대한 라이센스 문제이다. VxWorks는 라이센스 비용이 고가이며, 해당 소프트웨어 유지보수 비용을 고려해야만 기술 지원을 받을 수 있다. 현재 RAON 제어 시스템에서는 두개의 Node-locked 라이센스를 보유하고 있다. 이러한 부분에서 좀더 자유로울 수 있는 방법으로 open source 기반의 실시간 운영체제인 RTEMS를 고려할 수 있다.

2 Timing 시스템에서의 RTEMS

RTEMS[4]는 open source 기반의 realtime 운영체제 중 하나이다. 실시간 시스템의 정의는 여러형태로 기술되고 있지만 일반적인 형태로는 시스템 동작의 정확성이 논리적 정확성뿐만 아니라 시간적 정확성에서도 좌우되는 시스템으로 정의한다. 또한, 시스템의 수행 결과가 기능적으로 정확해야 할 뿐만 아니라, 결과가 도출되는 시간 역시 주어진 제약 조건을 만족시켜야 하는 시스템을 말한다. 예를 들어 제어 시스템에서 사용되는 실시간 시스템은 sensor로 부터 입력을 받아들여 이를 정해진 시간 내에 처리하여 actuator로 출력하며 극히 작은 시간적 오차를 허용한다.

일반적으로 실시간 운영체제의 특징은 아래와 같다.

- 선점형 스케쥴링
- 멀티쓰레드 지원 및 쓰레드간 우선순위
- 쓰레드간 동기화
- 인터럽트 지연시간, 시스템 콜 처리시간, 인터럽트 마스킹 시간

실시간 시스템은 주어진 job이 처리되어야 하는 deadline이 주어지고, 주어진 deadline이 지켜져야 하는 엄격성에 따라서 크게 두 가지 종류로 분류된다. 이러한 분류는 결국 실시간 OS의 스케쥴링 알고리즘에 의하여 좌우된다.

- Hard 실시간 시스템
  주어진 deadline을 만족시키지 못한 경우에 막대한 재산적 손실이나 인명의 피해를 주는 경우에 적용되는 시스템

- Soft 실시간 시스템
  시스템 온라인 트랜잭션 시스템과 같이 시간제약 조건을 만족시키지 못하더라도 경성의 경우처럼 치명적이지 않고 마감시간을 넘겨 수행을 마쳐도 계산의 결과가 의미가 있는 경우에 적용되는 시스템

2.1 특성

Embedded Systems 실시간 시스템은 시스템 특성상 시스템을 구성하기 위한 H/W, S/W의 특성이 존재한다. 하드웨어의 경우에는 고신뢰성을 제공하기 위해 결함 허용, 확장성과 유연성, 코드의 ROM화, 신뢰도가 높은 부품을 사용해야 한다. 또한, 소프트웨어의 경우에는 실시간 운영체제 또는 실시간 실행체제 등이 요구되고, 태스크의 스케줄링, 태스크 간 통신 및 동기화, 인터럽트 처리, 실시간 클록 관리 등의 기능 수행 등이 시스템 특성에 맞도록 구현되어야 한다. 항공기의 제어와 같이 주어진 시간 내에 제어를 하지 못할 경우 인명 피해와 같은 손실이 유발되는 경우를 Hard RTOS라 하고 그렇지 않은 경우를 Soft RTOS라 한다.

Timing 시스템을 위한 또다른 RTOS인 RTEMS의 사용을 위하여 아래와 같은 사항을 확인해야 한다.

- RTEMS 가 지원하는 BSP(Board Support Package)에 따른 CPU 타겟 보드 선정
  - MVME 6100 및 MVME 3100 VME Controller CPU Board
- 개발 Tool chains
  - GNU Cross compile 개발 환경
  - POSIX 호환성
- EPICS와의 통합 요부
  - 해당 CPU board 상에서 EPICS 환경 사용 가능 여부
- 현재 사용되고 있는 Reference site를 통하여 안정성 검증

2.2 설치 및 구성

RTEMS 개발 환경[5]이 구성된 최종 디렉토리 리스트는 아래와 같다.

```
silee@silee:~/rtems$ tree -L 2 .
|-- 4.10.2
|   |-- RTEMS -> rtems-4.10.2
|   |-- bld
|   |-- rtems-4.10.2
|   |-- rtems-libbspext-master
|   |-- rtems-libbspext-master.zip
|   |-- src -> rtems-4.10.2
|   `-- tgt
|-- archive
```

```
|   |-- autoconf-2.68.tar.bz2
|   |-- automake-1.11.1.tar.bz2
|   |-- binutils-2.20.1.tar.bz2
|   |-- gcc-core-4.4.5-rtems4.10-20101003.diff
|   |-- gcc-core-4.4.5.tar.bz2
|   |-- gcc-g++-4.4.5.tar.bz2
|   |-- gdb-7.1-rtems4.10-20100812.diff
|   |-- gdb-7.1.tar.bz2
|   |-- help2man-1.36.4.tar.gz
|   |-- mpfr-2.4.2.tar.bz2
|   |-- newlib-1.18.0-rtems4.10-20110114.diff
|   |-- newlib-1.18.0.tar.gz
|   |-- rtems-4.10-repo-conf-0.20101202.1.tar.bz2
|   |-- rtems-4.10-repo-conf-0.20101202.1.tar.xz
|   |-- rtems-4.10-repo-conf-0.22.tar.bz2
|   `-- rtems-4.10.2.tar.bz2
|-- gnu
|   |-- autoconf-2.68
|   |-- automake-1.11
|   `-- rtems-4.10
`-- tools
    |-- autoconf-2.68
    |-- automake-1.11.1
    |-- binutils-2.20.1
    |-- gcc-4.4.5
    |-- gdb-7.1
    |-- gnuTools.lis
    `-- newlib-1.18.0
```

RTEMS 환경 구성에서 사용되는 환경변수는 아래와 같다.

```
export GNU_DIR=${HOME}/rtems/gnu
export PATH=$GNU_DIR/autoconf-2.68/bin:$PATH
export PATH=$GNU_DIR/automake-1.11/bin:$PATH
export PATH=${GNU_DIR}/rtems-4.10/bin:/usr/local/bin:$PATH
export LD_LIBRARY_PATH=${GNU_DIR}/rtems-4.10/lib:$LD_LIBRARY_PATH
export RTEMS_ROOT=${HOME}/rtems/4.10.2
export TARGET_DIR=${RTEMS_ROOT}/tgt
export CC_FOR_HOST=gcc44
export CFLAGS_FOR_HOST="-g -O2 -std=gnu99"
export RTEMS_MAKEFILE_PATH=${TARGET_DIR}/powerpc-rtems4.10/mvme3100
```

설치 디렉토리 구성

```
$>mkdir rtems
$>cd rtems
$~/rtems>mkdir archive gnu tools
$~/rtems>ls
archive gnu tools
```

RTEMS 개발 환경 구성을 위하여 GNU tool chain[4][6]을 아래와 같이 구성한다.

Create tools/gnuTools.lis

```
--gnuTools.lis--
autoconf-2.68.tar.bz2
automake-1.11.1.tar.bz2
binutils-2.20.1.tar.bz2
gcc-core-4.4.5-rtems4.10-20101003.diff
gcc-core-4.4.5.tar.bz2
```

```
gcc-g++-4.4.5.tar.bz2
gdb-7.1-rtems4.10-20100812.diff
gdb-7.1.tar.bz2
gmp-4.3.2.tar.bz2
mpc-0.8.1.tar.gz
mpfr-2.4.2.tar.bz2
help2man-1.36.4.tar.gz
newlib-1.18.0-rtems4.10-20110114.diff
newlib-1.18.0.tar.gz
rtems-4.10-repo-conf-0.22.tar.bz2
rtems-4.10-repo-conf-0.20101202.1.tar.bz2
rtems-4.10-repo-conf-0.20101202.1.tar.xz
```

Download archive and/or patch files

```
$~/rtems>cd archive
$~/rtems/archive>wget --base ftp://ftp.rtems.com/pub/rtems/SOURCES/4.10/ -i ../tools/gnuTools.lis
```

Tool Chain Build

```
Ensure that your path variables are clean. In particular, make sure that . is not in your PATH.
Define an environment variable pointing to where the packages will be installed:

export GNU_DIR=${HOME}/rtems/gnu
cd ../tools
Build autoconf:
tar xjf archive/autoconf-2.68.tar.bz2
mkdir b-autoconf
cd b-autoconf/
../autoconf-2.68/configure --prefix=$GNU_DIR/autoconf-2.68
make all
make info
make install
cd ..
rm -rf b-autoconf/
--------
Add autoconf to the path in front of any other autoconf:
export PATH=$GNU_DIR/autoconf-2.68/bin:$PATH
Build automake:
tar xjf archive/automake-1.11.1.tar.bz2
mkdir b-automake
cd b-automake/
../automake-1.11.1/configure --prefix=$GNU_DIR/automake-1.11
make all
make info
make install
cd ..
rm -rf b-automake/

----------------

Add automake to the path in front of any other automake:
export PATH=$GNU_DIR/automake-1.11/bin:$PATH
Unpack the tools:
tar xjf archive/binutils-2.20.1.tar.bz2
tar xjf archive/gcc-core-4.4.5.tar.bz2
tar xjf archive/gcc-g++-4.4.5.tar.bz2
tar xzf archive/newlib-1.18.0.tar.gz
tar xjf archive/gdb-7.1.tar.bz2
tar xjf archive/gmp-4.3.2.tar.bz2 -> instead of this line, use atpitude install libgmp-dev
tar xzf archive/mpc-0.8.1.tar.gz -> instead of this line, use atpitude install libmpc2
```

```
tar xjf archive/mpfr-2.4.2.tar.bz2-> instead of this line, use atpitude install libmpfr-dev

Apply patches:
cd gdb-7.1/
cat ../archive/gdb-7.1-rtems4.10-20100812.diff | patch -p1 --dry-run
cat ../archive/gdb-7.1-rtems4.10-20100812.diff | patch -p1
cd ../gcc-4.5.1/
cat ../archive/gcc-core-4.4.5-rtems4.10-20101003.diff | patch -p1 --dry-run
cat ../archive/gcc-core-4.4.5-rtems4.10-20101003.diff | patch -p1
cd ../newlib-1.18.0/
cat ../archive/newlib-1.18.0-rtems4.10-20110114.diff | patch -p1 --dry-run
cat ../archive/newlib-1.18.0-rtems4.10-20110114.diff | patch -p1
cd ..

------------------------------
Build binutils:
mkdir b-binutils
cd b-binutils
../binutils-2.21/configure --target=powerpc-rtems4.10 --prefix=$GNU_DIR/rtems-4.10
make all
make info
make install
cd ..
rm -rf b-binutils
------------------------------
Add the just-built executables to the paths:
export PATH=${GNU_DIR}/rtems-4.10/bin:$PATH
export LD_LIBRARY_PATH=${GNU_DIR}/rtems-4.10/lib:$LD_LIBRARY_PATH
Build gcc. The make alltakes some time:
cd gcc-4.4.5/
ln -s ../newlib-1.18.0/newlib .
ln -s ../gmp-4.3.2 ./gmp //-> no need.
ln -s ../mpc-0.8.1 ./mpc //-> no need.
ln -s ../mpfr-2.4.2 ./mpfr //-> no need.
cd ..

mkdir b-gcc
cd b-gcc/
../gcc-4.4.5/configure --target=powerpc-rtems4.10 --with-gnu-as --with-gnu-ld --with-newlib --
    verbose --enable-threads --enable-languages="c,c++" --prefix=$GNU_DIR/rtems-4.10
make all
make info
make install
cd ..
rm -rf b-gcc
------------------------------------------------------

Build gdb. The make alltakes some time:
cd gdb-7.1/
ln -s ../newlib-1.18.0/newlib .
ln -s ../gmp-4.3.2 ./gmp // -> no need.
ln -s ../mpc-0.8.1 ./mpc // -> no need.
ln -s ../mpfr-2.4.2 ./mpfr // -> no need.
cd ..
mkdir b-gdb
cd b-gdb
../gdb-7.1/configure --target=powerpc-rtems4.10 --enable-sim --enable-sim-powerpc --enable-sim-
    timebase --enable-sim-hardware --prefix=$GNU_DIR/rtems-4.10
make all
make info
make install
cd ..
```

```
rm -rf b-gdb

//note
1. if you see the compile constant error "-Werror=unused-but-set-variable", add new compile flag
     CFLAGS = -Wno-error=unused-but-set-variable".
if you see the compile constant error "-Werror=unused-but-set-parameter", add new compile flag
     CFLAGS = -Wno-error=unused-but-set-parameter".
or add "CFLAGS = -Wno-error=unused-but-set-variable" line in config.status file

2. modify (sim/ppc/Makefile), delete $(LIBS)
  -psim: $(TARGETLIB) main.o $(LIBIBERTY_LIB) $(BFD_LIB) $(LIBS) $(LIBINTL_DEP)
  +psim: $(TARGETLIB) main.o $(LIBIBERTY_LIB) $(BFD_LIB) $(LIBINTL_DEP)
```

RTEMS 버전 4.10.2 다운로드 및 환경변수 설정

```
$~/rtems>mkdir 10.4.2
$~/rtems>cd 10.4.2
$~/rtems/10.4.2>wget ftp://ftp.rtems.org/pub/rtems/4.10.2/rtems-4.10.2.tar.bz2
$~/rtems/10.4.2>ls
rtems-4.10.2.tar.bz2
$~/rtems/10.4.2>export RTEMS_ROOT=${HOME}/rtems/4.10.2
$~/rtems/10.4.2>tar xjf rtems-4.10.2.tar.bz2
```

Build and install RTEMS

```
 Place the source directory in an appropriate spot and create some links to it
$~/rtems/10.4.2>ln -s rtems-4.10.2 src
$~/rtems/10.4.2>ln -s rtems-4.10.2 RTEMS

There are additional instructions at Quick Start but distribution probably contains more up-to-
     date instructions

See the release notes
See ${RTEMS_ROOT}/src/README
See ${RTEMS_ROOT}/src/INSTALL
See ${RTEMS_ROOT}/src/README.configure

Apply patches. Some may fail because the target files have changed.

In order to support Gen 1 RCEs, a DHCP patch by Matt Weaver must be applied. It is in$\SVNROOT}/
     rtems/patches/dhcp.diff. The affected files are${RTEMS_ROOT}/src/cpukit/libnetworking/rtems/
     dhcp.h and rtems_dhcp.c.

In order to support Virtex 5 chips, apply the patches in

${SVNROOT}/rtems/patches/ppc440.diff

Apply the patch in ${RTEMS_ROOT}/src/cpukit/libnetworking/rtems/bsp.diff to update
theacinclude.m4 in the powerpc BSP directory so that the additional BSPs are included
If any of the patches failed, they should be updated and committed to the repository

cd ${RTEMS_ROOT}/src

cat ~/dat/rtems/patches/dhcp.diff | patch -p1 --dry-run
cat ~/dat/rtems/patches/dhcp.diff | patch -p1
cat ~/dat/rtems/patches/ppc440.diff | patch -p1 --dry-run
cat ~/dat/rtems/patches/ppc440.diff | patch -p1
cat ~/dat/rtems/patches/bsp.diff | patch -p1 --dry-run
cat ~/dat/rtems/patches/bsp.diff | patch -p1
```

```
Checkout (or update) the Virtex BSPs from the DAT repository (${SVNROOT}/rtems/bsp) to${RTEMS_ROOT
    }/src/c/src/lib/libbsp/powerpc
Be sure that the prerequisite tools (built above) are in the appropriate paths, e.g.

export GNU_DIR=${HOME}/rtems/gnu
export PATH=${GNU_DIR}/autoconf-2.68/bin:$PATH
export PATH=${GNU_DIR}/automake-1.11/bin:$PATH
export PATH=${GNU_DIR}/rtems-4.10/bin:$PATH
export LD_LIBRARY_PATH=${GNU_DIR}/rtems-4.10/lib:$LD_LIBRARY_PATH


Create the build and target installation directories.
It mightbe useful to have these on local disks for faster access, as shown here. If so, be sure to
     cross soft-link everything:
mkdir ${RTEMS_ROOT}/bld

export TARGET_DIR=/u1/reg/package/rtems/4.10.2/tgt
mkdir -p $TARGET_DIR
ln -s ${RTEMS_ROOT}/RTEMS
ln -s ${RTEMS_ROOT}/src
cd $RTEMS_ROOT
ln -s ${RTEMS_ROOT}/bld
ln -s ${RTEMS_ROOT}/tgt
cd -

You may need to override some portions of the environment. Do configure --help. Note that these
    environment variables are not additive so you need to find the original value in order to add
     another switch. For example, in the RTEMS 4.9 days we needed to do

export CC_FOR_HOST=gcc44
export CFLAGS_FOR_HOST="-g -O2 -std=gnu99"

Build and install RTEMS:
cd ${RTEMS_ROOT}/src
./bootstrap
cd ../bld

../src/configure --target=powerpc-rtems4.10 --enable-posix --enable-networking --enable-rdbg --
    enable-cxx --enable-maintainer-mode --enable-rtemsbsp="mvme3100 mvme5500" --prefix=
    $TARGET_DIR
make -j all
make -j install
cd ..

The results can be found in ${TARGET_DIR}/share. Also see http://www.rtems.com/onlinedocs.html.
Install the rtems-gdb-stub in the new target.
```

Build BSP for mvme3100 on RTEMS

```
Place the BSP in the RTEMS tree at ${RTEMS_ROOT}/src/c/src/lib/libbsp/<CPU>/<BSP>
Not sure whether one can build a BSP outside of the RTEMS tree. It may not even make sense. If
    that desire exists, a private RTEMS tree may be needed.
Ensure the PATH and LD_LIBRARY_PATH environment variables aren't polluted
Set up the appropriate environment variables

export RTEMS_ROOT=${HOME}/rtems/4.10.2
export TARGET_DIR={RTEMS_ROOT}/tgt
export GNU_DIR= ${HOME}/rtems/gnu
export PATH=${GNU_DIR}/autoconf-2.68/bin:$PATH
export PATH=${GNU_DIR}/automake-1.11/bin:$PATH
export PATH=${GNU_DIR}/rtems-4.10/bin:$PATH
```

```
export LD_LIBRARY_PATH=${GNU_DIR}/rtems-4.10/lib:$LD_LIBRARY_PATH

Build RTEMS as above
Description file of compiler switches to be used for this BSP (still needed? May have been a 4.9
     requirement.)

cd ${RTEMS_ROOT}/src/make/custom
ln -s ../../c/src/lib/libbsp/powerpc/mvme3100/make/custom/mvme3100.cfg
ln -s ../../c/src/lib/libbsp/powerpc/mvme5500/make/custom/mvme5500.cfg


Be sure that the following lines are in the list of BSPs by editing${RTEMS_ROOT}/src/c/src/lib/
     libbsp/powerpc/acinclude.m4(nominally done by the bsp.diff patch above)

 //no need, only confirm.

  mvme3100 )
    AC_CONFIG_SUBDIRS([mvme3100]);;

  mvme5500 )
   AC_CONFIG_SUBDIRS([mvme5500]);;


If some configure or Makefile.in must be regenerated, i.e., configure.ac and/or Makefile.am has
     been modified, bootstrapmust be run again. It is sufficient to run it from the level at which
      the change was made. There is no need to run it on the whole tree again (which is arduously
     slow).
First clean out the previously generated files using bootstrap -c. This will also blow away the
     result of theconfigure step, which must thus be redone. See ~claus:Build and install RTEMS.


cd ${RTEMS_ROOT}/src/...
./bootstrap -c
./bootstrap
cd ${RTEMS_ROOT}/bld
../src/configure --target=powerpc-rtems4.10 --enable-posix --enable-networking --enable-rdbg --
     enable-cxx --enable-maintainer-mode --enable-rtemsbsp="mvme3100 mvme5500" --prefix=
     $TARGET_DIR
make all
make install

Take care to use the correct set of build tools by specifying --target correctly, i.e. powerpc-
     rtems4.10vs powerpc-rtems4.11
Nominally one can give multiple instances of --enable-rtemsbsp=BSP to the configure line, but one
     can apparently also give the configure line multiple times with different --enable-rtemsbsp=
     BSP each time.
if need, CFLAGS = ---- -Wno-error=unused-but-set-variable

One can also do:
cd ${RTEMS_DIR}/bld
make RTEMS_BSP="bsp1 bsp2 ..." all
```

BSP Extension RTEMS

```
1. git clone https://github.com/epicsdeb/rtems-libbspext   bspext
2. cd bspext
3. make
4. cd o-optimize
5. confirm libbspExt.a
6. cp libbspExt.a ${TARGET_DIR)/powerpc-rtems4.10/mvme3100/lib/
7.cp bsp/bspExt.h ${TARGET_DIR)/powerpc-rtems4.10/mvme3100/lib/include
```

```
8. rtems recompile
```

3 RTEMS를 위한 EPICS base 설치

EPICS base compile for RTEMS

RTEMS 상[7][8]에서 EPICS 사용을 위한 설치에 대한 내용은 아래와 같다.

```
in ${EPICS_BASE}/configure/CONFIG_SITE file

$>cd ${EPICS_BASE}/configure/
$rtems@silee:~/epics/R3.14.12.4/base/configure>vi CONFIG_SITE

CROSS_COMPILER_TARGET_ARCHS=RTEMS-mvme3100

after saving

in ${EPICS_BASE}/configure/os/CONFIG_SITE.Common.RTEMS file

$rtems@silee:~/epics/R3.14.12.4/base/configure>cd os
$rtems@silee:~/epics/R3.14.12.4/base/configure/os>vi CONFIG_SITE.Common.RTEMS

# Site-specific information for all RTEMS targets
#-----------------------------------------------------

# Where to find RTEMS
#
RTEMS_VERSION = 4.10
RTEMS_BASE = ${RTEMS_ROOT}/tgt

# Cross-compile toolchain in $(RTEMS_TOOLS)/bin
#
RTEMS_TOOLS = ${HOME}/rtems/gnu/rtems-$(RTEMS_VERSION)/
COMMANDLINE_LIBRARY = EPICS
```

RTEMS EPICS example

```
# Run the makeBaseApp.pl program to create the example application:
$rtemsioc>makeBaseApp.pl -t example test
$rtemsioc>makeBaseApp.pl -i -t example -a RTEMS-mvme3100 test
$rtemsioc>make

$~/epics/R3.14.12.4/siteApps/rtemsioc$ tree -L 2 .
.
|-- Makefile
|-- bin
|   |-- RTEMS-mvme3100
|   `-- linux-x86_64
|-- configure
|   |-- CONFIG
|   |-- CONFIG_SITE
|   |-- Makefile
|   |-- O.Common
|   |-- O.RTEMS-mvme3100
|   |-- O.linux-x86_64
|   |-- RELEASE
|   |-- RULES
```

```
|   |-- RULES.ioc
|   |-- RULES_DIRS
|   `-- RULES_TOP
|-- db
|   |-- dbExample1.db
|   |-- dbExample2.db
|   |-- dbSubExample.db
|   |-- user.substitutions
|   `-- userHost.substitutions
|-- dbd
|   |-- test.dbd
|   |-- xxxRecord.dbd
|   `-- xxxSupport.dbd
|-- include
|   `-- xxxRecord.h
|-- iocBoot
|   |-- Makefile
|   |-- ioctest
|   `-- nfsCommands
|-- lib
|   |-- RTEMS-mvme3100
|   `-- linux-x86_64
`-- testApp
|-- Db
|-- Makefile
`-- src

$~/epics/R3.14.12.4/siteApps/rtemsioc/bin/RTEMS-mvme3100$ ls
test  test.boot
```

Extension EPICS lib for MRF IOC

```
// devlib2
hg clone http://epics.hg.sourceforge.net:8000/hgroot/epics/devlib2

// mrfioc2
hg clone http://epics.hg.sourceforge.net:8000/hgroot/epics/mrfioc2
1. move devlib2 to siteApps
2. cd devlib2/configure
3. edit RELEASE -> EPICS_BASE=${HOME}/epics/R3.14.12.4/base
4. compile
5. cd mrfioc2/configure
6. edit RELEASE ->
DEVLIB2=${HOME}/epics/R3.14.12.4/siteApps/devlib2/
EPICS_BASE=${HOME}/epics/R3.14.12.4/base
```

Bibliography

[1] MRF Timing System.
http://www.mrf.fi/pdf/presentations/MRF.CERN.Feb2008.pdf.

[2] MVME 6100 Single Board Computer, .
https://www.slac.stanford.edu/grp/lcls/controls/global/standards/hardware/v6100a_ih.pdf.

[3] MVME 3100 Single Board Computer, .
http://www.slac.stanford.edu/grp/ssrl/spear/epics/vme/6806800m28c_mvme3100_iu.pdf.

[4] Realtime OS RTEMS, .
http://www.rtems.com.

[5] RTEMS Installation, .
https://confluence.slac.stanford.edu/display/CCI/RTEMS+Installation#RTEMSInstallation-Selectandsetuptheenvironment.

[6] RTEMS Download, .
ftp://ftp.rtems.com/pub/rtems.

[7] EPICS, Experimental Physics ans Industrial Control System, .
http://www.aps.anl.gov/epics.

[8] EPICS on RTEMS, .
http://www.aps.anl.gov/epics/base/RTEMS/tutorial/.