Timing System

LCLS Controls Timing

Older Timing Documents Web Page

Sharepoint timing documents

Beam Synchronous Acquisition (BSA)

Using Facility Agnostic Databases

evrmaKernel Driver / evrManager / evrClient modules

SC Timing

SC Timing triggers: User Guide

EPICS EVR HowTo

The information below can also be found in the event module README file at our CVS site

Instructions for EVR users

These instructions are for EVENT module event-R4-1-0 or later.
The basic steps when configuring a new IOC are in parts I through IV.

Contents:
I - Adding event module support to your IOC application
II - Adding event databases and EVR configuration to your IOC startup file
IIIa - Standard EDM Screens and Their Macros
IIIb - Adding Timing Display autogen to your application
IV - Hardware Setup
V - Special One-Time PMC Setup
VI - Checkout Instructions
VII - Updating from versions older than event-R3-5-0, for IOC Engineers
VIII - R3-14-12 Migration (migration from pre-event-R3-2-2-br_generaltime branch) issues
IX - Using event module C functions and global variables in your IOC application
X - Consideration for the multiple beam program
XI - Consideration to use the event invariant delay
XII - Consideration for Autosave
XIII - Consideration for the Linux PMC EVR
XIV - General BSA template
XV - Library and DBD files for EVG applicaton
XVI - Library and DBD files for EVR application

If you are still using an older EVENT module please use the README for that specific module version

I - Adding the event module support to your IOC application:

---

(1) Add EVENT and AUTOSAVE to configure/RELEASE and clean/rebuild configure.

(2) Link event libraries into your app by adding to xxxApp/src/Makefile:

```
xxx_LIBS += evrSupport
xxx_LIBS += devMrfEr
xxx_LIBS += mrfVme64x (VME systems only)
xxx_LIBS += drvMrf (Linux systems only)
xxx_LIBS += autosave
```

Note that the order of the above libraries is important.

(3) Add the following .dbd files to xxxApp/src/Makefile
or to xxxApp/src/xxxInclude.dbd:

```
xxx_DBD += evrSupport.dbd
xxx_DBD += devMrfEr.dbd
xxx_DBD += autosave.dbd
```

(4) In most cases, you can load these facility-agnostic db files directly in st.cmd. They have reasonable defaults and standardized names. You can then skip steps 5-8 below.
To do that, add to xxxApp/Db/Makefile:

```
DB_INSTALLS += EvrPmc.db
DB_INSTALLS += Pattern.db
DB_INSTALLS += PMC-trig.db (if EVR is PMC type)
DB_INSTALLS += VME-trig.db (if EVR is VME type)
DB_INSTALLS += VME-TTB-trig.db (for a rear transition or breakout board)
DB_INSTALLS += Bsa.db (if BSA will be used)
```

Details are here

(5) If you aren't using facility-agnostic dbs, add EVR databases to your application. For an example, see:
event/<release>/evrIoc/Db/evrXL05.substitutions

(6) If you aren't using facility-agnostic dbs, add pattern-related databases to your application. For an example, see:
event/<release>/evrIoc/Db/evrPatternXL05.substitutions

(7) If you aren't using facility-agnostic dbs, for IOCs with triggered devices, add trigger databases to your application.
For an example, see:
event/<release>/evrIoc/Db/trigXL05.substitutions. Also add event EDM display auto-generation to your trigger database file. See section 'Adding Timing Display autogen to your application'.

(8) If you aren't using facility-agnostic dbs, for IOCs with beam-synchronous acquisition (BSA), add BSA databases to your application. For an example, see:
event/<release>/evrIoc/Db/bsaXL05.substitutions

(9) For each record that is an input to BSA, add a forward-link (FLNK) to record processing that occurs at beam or trigger time:
field(FLNK, "<dev>:EF<secn>")
Consult with the HLA group to determine the proper values of <dev> and <secn> for your application.

(10) For IOCs with triggered devices, see below section 'Consideration for Autosave' for instructions on setting up autosave of your trigger save/restore setpoints.

(11) For IOCs with triggered devices which use event codes that are generated at a "slow" rate (rate slower than beam full rate like a 10Hz event code), if records for your devices process or finish processing 8.3 msec beyond the time of the trigger (ie, profile monitors), set the TSE field of those records to the event code. The records will then be timestamped at the time that the event code was received instead of the time of the last fiducial. The records can then be correlated with other records on other IOCs (ie, BPMs) that were acquired on the same pulse. Make sure that the event code IRQ for all possible TSE values is enabled in your EVR database.

(12) Add each EVR to the following displays:
$EDM/event/evnt_<loca>_main.edl
and one of the following:
$EDM/event/evnt_all_controllers.edl
$EDM/event/evnt_all_beamDiag1.edl
$EDM/event/evnt_all_beamDiag2.edl
$EDM/event/evnt_all_laser.edl
$EDM/event/evnt_all_profileMonitor.edl

(13) Add the following PVs to ALH under the "IOC" subgroup
of the "Event" subgroup for the applicable machine areas:
For each IOC:
<iocname>:PATTERNSTATE
<iocname>:FIDUCIALRATE
<iocname>:NTPSTATE
For each EVR:
<evrname>:LINK
<evrname>:LINKERR

(14) Add event code sequence PVs to your application by installing the
event code sequence db file...

Db/Makefile:
DB_INSTALLS += $(EVENT)/db/eventCodeSequenceAll.db

Doing so provides the following PVs
$(DEVICE):ECS_$(EVENT_CODE)EV
$(DEVICE):ECS_$(EVENT_CODE)CNT
$(DEVICE):ECS_$(EVENT_CODE)RATE
$(DEVICE):ECS_$(EVENT_CODE)NAME

for 67 <= $(EVENT_CODE) <= 98

II - Adding event databases and EVR configuration to your IOC startup file

---

(1) If your IOC supports beam-synchronous acquisition, it will need to support
large waveforms for channel access. Add the following command
before iocInit:

epicsEnvSet("EPICS_CA_MAX_ARRAY_BYTES", "32000")

If your IOC already supports waveforms larger than 32000 bytes for other
reasons, then this command is not needed.

(2) Load databases created in the previous section (before iocInit).
If using the general purpose db files from step 4 above, define macros
as documented at the top of the corresponding substitutions files, e.g.

```
dbLoadRecords("db/EvrPmc.db","EVR=EVR:B34:EV05,CRD=0, SYS=SYS0")
dbLoadRecords("db/Pattern.db","IOC=IOC:B34:EV05, SYS=SYS0")
dbLoadRecords("db/Bsa.db","DEVICE=TRIG:B34:EV05, ATRB=COUNTER")
dbLoadRecords("db/PMC-trig.db", "LOCA=B34,UNIT=EV05,SYS=SYS0,IOC=IOC:B34:EV05")
dbLoadRecords("db/VME-TTB-trig.db", "LOCA=B34,UNIT=EV05,SYS=SYS0,IOC=IOC:B34:EV05")
```

(3) If any of the EVRs on your IOC is a VME model 200 and you are running
RTEMS 4.9.1 on an MVME6100, add the following lines before ErConfigure
(do NOT add these lines if your EVR is PMC or VME model 230 or higher):

```
# From Till Straumann (for RTEMS 4.9.1 upgrade):
# This should set the VME chip into a mode where it
# surrenders the VME bus after every transaction.
# This means that the master has to rearbitrate for the bus for every cycle
# (which slows things down).
#
# The faster setting I had enabled would let the master hold on to
# until some other master requests it.
*(long*)(vmeTsi148RegBase + 0x234) &= ~ 0x18
```

(4) Add ErConfigure for each EVR before iocInit.

```
VME:      ErConfigure(<instance>,<address>,<vector>,<level>,0)
PMC:      ErConfigure(<instance>,    0    ,    0   ,   0   ,1)
Embedded: ErConfigure(<instance>,    0    ,<vector>,<level>,2)

PCI-E:    ErConfigure(<instance>,    0    ,    0   ,   0   ,4)
```

where instance = EVR instance, starting from 0, incrementing by 1
for each subsequent card. Only 1 EVR instance
is allowed for Embedded EVRs.
and address = VME card address, starting from 0x300000,
incrementing by 0x100000 for each subsequent card
(0 for PMC and Embedded)
and vector = VME or Embedded interrupt vector.
For VME, start from 0x60 and increment by 0x02 for
each subsequent card.
(0 for PMC)
and level = VME or Embedded interrupt level.
For VME, set to 4. Can be the same for all EVRs.
(0 for PMC)
and 0 = VME
or 1 = PMC
or 2 = Embedded

or 4 = PCI-E

(5) Add evrInitialize (after ErConfigure) if a fiducial routine will be
registered before iocInit driver initialization:
evrInitialize()

(6) For IOCs with triggers, add a line to restore the trigger setpoints
before iocInit. See the ChannelWatcher README file.

IIIa - Standard EDM Screens and Their Macros

At the top of most of the timing db files you will find an edl file listed that
is associated with that db file. Here are the standard ones, the macros that
should be passed to each, and how to define those macros.

Trigger settings
Db file: *trig.db
EDL file: evnt_trig_*.edl to match hardware (PMC/VME, with/without transition board variations) evnt_trig_PMC_TTB.edl evnt_trig_PMC_noTTB.edl
evnt_trig_VME_TTB.edl evnt_trig_VME_noTTB.edl
Macros: DEV, LOCA, UNIT, IOC, SYS
Macros are the same ones used with *trig.db. The default value for DEV is
TRIG:$(LOCA):$(UNIT). If the triggers in your EVR don't all have the same prefix
(TRIG:LOCA:UNIT or DEV), you can't use this screen. Copy the appropriate screen
and customize PV names, or use the display autogen approach.

The following will have all macros derived from the trigger screen macros when
launched from trigger screen using the "Experts" button:
evrPatternDiags.edl (Pattern.db)
evrDiags.edl (EvrPmc.db)
evrTriggerDiags.edl (EvrPmc.db)

BSA diagnostics
Db file: Bsa.db
EDL file: evnt_bsa_dev_edefs.edl
Macros:
DEV_BASE - the BSA data source prefix, e.g. BPMS:IN10:371, TRIG:B34:EV05
BSA - the name of the edef specific screen, e.g. evnt_bsa_dev, evnt_bsa_llrf...
The other macros depend on the screen specified as $(BSA). These are for evnt_bsa_dev:
DEV - prefix for the data source, generally $(DEV_BASE):
SECN - part of PV name defined as SECN, ATTR, or ATRB in the bsa db file, e.g. COUNTER
SYS - SYS0 etc.

IIIb - Adding Timing Display autogen to your application

IOC Engineer steps:
1) Add the following comment tags to your *trig*.substitutions file:

```
      <snip>
  file evrDevTrig.db
  {
  #EVR EVR:XT01:IM01
  #FILE evnt_xt01_im01
  #CONTROLPV1 SIOC:SYS6:AL00:MODE
  #CONTROLPV2 SIOC:SYS6:AL00:TOD
      <snip>
```

where #EVR is your EVR device name, #FILE is the output evnt*.edl filename,
and CONTROLPV* indicates the PV names for mode and time of day at screen bottom,
(indicating PRODUCTION or DEVELOPMENT, and Time of Day) from approproate IOC.
Note that the ":SYS*:" area will change depending on the machine.
Note that if CONTROLPV* tags are omitted, the PVs default to "SYS0" for LCLS
Note that FACET maps to SYS1 and acctest maps to SYS6

2) Add to your Db/Makefile

```
  gen_trig_edl:
    <TAB>create_edm_event_msi.py <*trig*.substitutions1 file>
    <TAB>create_edm_event_msi.py <*trig*.substitutions2 file>
          ....
```

Note:
$TOOLS/script contains the create_edm_event_msi.py script used
$EDM/templates contain the edm template files used by create_edm_event_msi.py
$EDM/install is the area to which generated screens are copied

Example: See /afs/slac/g/acctest/epics/iocTop/R3-14-12/BeamCharge/MAIN_TRUNK/BeamChargeApp/Db/xta
inspect IOC-XT01-IM01trig.substitutions and Makefile

Note that the following example output from create_edm_event_msi.py is normal:

```
[drogind@cdlx08 xta]$ make gen_trig_edl
create_edm_event_msi.py IOC-XT01-IM01trig.substitutions
channel count =  1
System as determined from CONTROLPV1  SYS6
Done generating  evnt_xt01_im01_msi.substitution
EDM templates from /afs/slac/g/acctest/tools/edm/display/templates
Generating EDL file /afs/slac/g/acctest/tools/edm/display/install/evnt_xt01_im01.edl
Warning: unexpanded macros in ouput
```

For any other errors, please consult with Murali or Debbie

IV - Hardware Setup

---

(1) Installation instructions for the PMC EVR are here:
http://www.slac.stanford.edu/grp/lcls/controls/global/subsystems/timing/PMC-EVR_install_inst_v1d0.pdf

(2) For VME EVRs, at times, the white tab in the IEEE handle does not pop out all the way and the module blue light remains on and the module will not configure. Make sure the tab is out when seating the module.

(3) Connect fiber from a nearby timing fiber fanout module to each EVR. These fanout modules are identified in the timing system block diagram by Mike Browne:
https://sharepoint.slac.stanford.edu/sites/LCLS%20Document%20Storage/01%20-%20LCLS%20Systems/electronbeamsys/controls/Shared%20Documents/Timing/TIMING%20System%20BD.pdf

If the fiber has just the receive cable, connect the single cable to the top spigot of the transceiver (bottom when the black lock is on the right).

V - Special One-Time PMC Setup

The PMC EVRs, as delivered, have incorrect device and vendor IDs in their EEPROMs. These IDs must be corrected before the module can be configured by the software. Use the following one-time procedure to burn in the correct values for each PMC EVR.

WARNING - WARNING - WARNING:
IF THERE ARE OTHER PMCS CONNECTED TO YOUR CPU, IT'S POSSIBLE THIS PROCEDURE WILL WRITE TO THE WRONG PMC AND RUIN IT!! REMOVE NON-EVR PMCS BEFORE RUNNING THIS PROCEDURE TO BE SAFE.

(1) Boot the CPU with an IOC application that includes the event package. The "ErConfigure" command for each PMC EVR will fail if it hasn't yet been corrected.

(2) Type the following command to read the vendor and device ID for PMC instance 0 or 1:

```
Cexp>evrEEPROMFixup(<0 or 1>,0)

Look for these results:

PLX 9030 found at 0xe110<0or1>000, EEPROM present
Subsytem vendor ID 0x10b5, device ID 0x9030

> Sanity check passed...
> Informational mode only; EEPROM contents unmodified.
==> Call again with nonzero 'doit' argument to apply change.
0x00000000
```

(3) If the results are as expected, type the following command to correct the vendor and device ID:

```
Cexp> evrEEPROMFixup(<0 or 1>,1)

Look for these results:

PLX 9030 found at 0xe110<0or1>000, EEPROM present
Subsytem vendor ID 0x10b5, device ID 0x9030

> Sanity check passed...
> Writing MRF id (0x1a3e) to EEPROM SSVID
> Writing EVR id (0x10c8) to EEPROM SSDID
==> EEPROM successfully fixed
0x00000000 (0)
```

(4) Reboot the CPU and check that the ErConfigure commands are successful.

VI - Checkout Instructions

---

(1) Errors when loading event databases, contact khkim.
(2) Errors from ErConfigure, contact khkim.
(3) ErConfigure is successful but EVR has red LEDs, contact jedu to make
sure your fiber is connected properly from EVG through fanout modules to
your EVR.
(4) Unexpected or ominous messages at your console, contact saa.
(5) Once the IOC is up and configured successfully, type "dbcar()" at the
Cexp prompt and make sure there are no disconnected channels. Contact saa
if there are.
(6) From the home LCLS EDM display (lclshome) on a production machine, select
"Event". Under "EVR IOCs", select the button containing your IOC. Select
each display next to your IOC name (all channels must be connected):
  * EVR display - look for non-zero FPGA and "ON" for "Rx Link".
  * "Triggers..." display - check that all the event codes that you expect
    to use on your IOC are set. Check that the proper channels are enabled
    for the proper event code(s). If any records on your IOC use a non-zero
    TSE, check that the event codes used by your TSEs have the IRQ enabled.
  * "Events..." display - check that the rate for event code 1 is exactly
    360Hz. For event codes that have the IRQ set, check that their rates
    are as expected.
  * "Pattern..." display - check that data is changing every 2 seconds. From
    the main Event display, select "EVG Diags...".
    The data on your EVR pattern display and the EVG pattern display must
    match. If the pattern data is all magenta, the fiber to your EVR is
    probably not connected properly. Select "General Time" and check that
    the "Best Time Event Provider" is evrTimeGet.
  * "Devices..." display - check that all your triggered devices are
    available and the polarity, pulse width, delay (TDES), and control (TCTL)
    are properly set.
  * Contact saa if any problem.
(7) If the timestamps of your records are not reasonable, contact saa.
(8) Check that the polarity, pulse width, delay, and control of your devices
    are being monitored and updated by your IOC's ChannelWatcher and are
    properly restored on reboot.
(9) Check that the Channel Archiver is recording all your trigger delays and
    that there are no disconnected channels.

VII - Updating from versions older than event-R3-5-0, for IOC Engineers

---

Please, remove both macros TEC (Trigger Event Code Name) and ACTV (Activate Event Code Invariant Delay)
from your trig.substitution file. (there is no more trigger event code name.
The forward/backward propagation and event code invariant delay is a default mode)

IOC engineers need to put new macro SYS to describe the beam program in the trig.substitutions and evr.substitutions files

Please, follow the following rule.

SYS=SYS0: LCLS
SYS=SYS1: FACET
SYS=SYS2: LCLS II
SYS=SYS6: NLCTA

Note)
If you really want to disable the event code invariant delay,
you can put ACTV macro and set it to ZERO.
Then the feature will be disabled. The intrinsic delay (EVG delay)
will be set to 13004 as a default.
If you want to change the default value, you can use TEC macro.
Please, put your default instrinsic delay value for the TEC macro.

VIII - R3-14-12 Migration (migration from pre-event-R3-2-2-br_generaltime branch) issues

---

Before starting please read this section and
 'Consideration for Autosave', and
 'Adding Timing Display autogen to your application'
which are included near the end of this README

(1) Update modules

 If this is a move to epics-R3-14-12 you will need to remove the GENERALTIME and RESTORE modules, update all other modules in your RELEASE file to those built for epics-R3-14-12, and use AUTOSAVE instead of RESTORE. Before proceeding please read the AUTOSAVE module README file, and the sections of this README

 'Consideration for Autosave'

which are included near the end of this README file.
   Edit configure/RELEASE file to use the latest event module
   Please, check up the module dependency also.

(2) Edit *evr.substitutions file for the er record instance
   We can use the following database templates which have been used from old version of event module
   for the er record instance.
   . evr.db, evrWithDelays.db, evrWithExtDelays.db, evrWithFrontPanel.db

   The new event module also provides new template for the Linux PMC platform
   . evrPmc.db, evrWithDelayPmc.db, evrWithExtDelaysPmc.db, evrWithFronPanelPmc.db
   If you are planing to use the Linux PMC EVR, please choose these template

   (Remark) Please, find more details in the slides: 'How to use Tming System as a Client'.

(3) Edit *evr.substitutions file for the trigger/event configuration

   Please, use evrEventCtrl.db template for the trigger/event configuration
   Need to need to describe a list of event number which will be used for the trigger,
   and also need to put initial trigger configuration.
   But, it can be changed at runtime if we need.

   (Remark) the trigger/event cofniguration part was not changed from the previous version,
   but, need to consider for each beam program. Each beam program has different event list.

(4) Edit *pattern.substitutions file to choose proper timeslots, and propoer event list for each beam program

   Put correct active timeslots for the macors: ST1ST, and TS2ND for the evrPatternAll.db.
   The evrPatternAll.db is a common template for all of beam programs.

   Please, put a correct evrEvent*.db template. Each beam program has own template

   (Remark) Please, look at 'consideration for the multiple beam program' section in this file
         and also look at the slies: 'How to use Timing System as a Client'.

(5) Edit *trig.substitutions file for the event code invariant delay and, the automatic generating EDM screen

   Put new macros: TOUT and ACTV for the evrDevTrig.db template. It is for the event code invariant delay.
   Please, find more detialed information in the 'consideration to use the event invariant delay' section in this file.
   Also, can find an example in the slides: 'How to use Timing System as a Client'.

   Put new comment tags in the evrDevTrig.db section in the substitutions file. It is for the automatic generating EDM screen.
   Please read below section Adding Timing Display autogen to your application'.
   Also, please find more detailed information and example in the slides: 'How to use Timing System as a Client'.

(6) Edit Makefile in the src/Db directory for the automatic generating EDM screen

   Need to put new target to generate the EDM screen from the *trig.substitutions file

   example)

      gen_trig_edl:
      <TAB>   create_edm_event_msi.py   IOC-XT01-IM01trig.substitutions

   The *trig.substitutions file name should be matched with yours.

(7) Misc.

   - Please, look at the section for the 'Consideration for the autusave' in this file
   - Recommend to use high level delay and width PVs for the save-restore
   - Need to use save-restore for the er record setting
   - Please, do not use the pre-scaler for the extended delay
     . the event code invariant delay does not work properly with the pre-scaler
     . EVR provides enough range for the delay without the pre-scaler

IX - Using event module functions and global variables in your IOC application:

---

(1) If you want a routine to run after the fiducial at 360hz by the evrTask,
   (which runs at epicsThreadPriorityHigh+1), register the routine in your
   initialization code:

   #include "evrTime.h"

```
static void *optionalInput = 0;
static void fiducialRoutine(void * optionalInput) {
... your 360hz logic here ...
... to get the current or the one of the next 2 360hz patterns...
   evrModifier_ta modifier_a;
   epicsTimeStamp time_s;
   unsigned long  patternStatus; /* see evrPattern.h for values */
   int status = evrTimeGetFromPipeline(&time_s,
   <evrTimeCurrent, evrTimeNext1, evrTimeNext2, evrTimeActive>,
   modifier_a, &patternStatus, 0,0,0);
... check status - status is non-zero when same/missing pulses ...
...   in the pipeline or bad timestamp.
... check patternStatus - status is non-zero when there is a timeout,
...   unsynchronized pulse ID, and other PATTERN error listed in
...   evrPattern.h.
... use BEAMCODE, TIMESLOT macros defined in evrPattern.h ...
...   to parse out beam code and time slot from modifier_a
...   TIMESLOT is zero if pattern is bad.
... use PULSEID macro defined in evrTime.h ...
...   to parse out pulse ID from time_s
...   PULSEID is set to PULSEID_INVALID when pattern is bad.
... do your 360hz logic requiring pattern here ...
}
... in your init code ...
... must be called after ErConfigure and evrInitialize ...
... set optionalInput as needed for your application ...
evrTimeRegister((FIDUCIALFUNCTION)fiducialRoutine, optionalInput);
...
```

See event/<release>/evrIoc/src/mpsEvrProc.c for an example.

(2) If you prefer not to use the FLNK to provide input to BSA, use the API directly:

```
#include "bsa.h"
... your initialization logic - do at or before iocInit ....
... for every value that BSA needs (ie, X,Y,TMIT for each device) ...
char[PVNAME_STRINGSZ] bsaName = <BSA name for the data>
void                  *dpvt = 0; /* unique per value */
status = bsaSecnInit(bsaName, 0, &dpvt);

... your data processing logic here ...
... to update BSA after data is ready ...
... for every value that BSA needs (ie, X,Y,TMIT for each device) ...
epicsTimeStamp dataTimeStamp = <timestamp of your data>
double         dataValue = <value of your data>
epicsEnum16    dataStat = <EPICS alarm status of your data>
epicsEnum16    dataSevr = <EPICS alarm severity of your data>
void          *dpvt = <value of dpvt from bsaSecnInit>
status = bsaSecnAvg(&dataTimeStamp, dataValue, dataStat, dataSevr, dpvt);
```

(3) Two global variables providing the high resolution time of the fiducial are available for diagnostics purposes:

evrFiducialTime - high resolution time of the last fiducial (360hz update)

evrActiveFiducialTime - high resolution time of the last timeslot 1 or 4 fiducial (120hz update)

These values are also provided in <ioc>:FIDUCIAL.A and B, respectively.

(4) A general-purpose pattern-matching function is available for applications that need to check pattern modifiers against inclusion and exclusion masks, with optional beam code and time slot check. To call:

```
#include "evrPattern.h"

   evrModifier_ta modifier_a;
... get the pattern modifiers (ie, using evrTimeGetFromPipeline)  ...
   evrModifier_ta inclusion_a, exclusion_a;
   unsigned long  beamcode; /* 0 = any beam code */
   unsigned long  timeslot; /* 0 = any time slot */
... fill in inclusion and exclusion masks, beamcode, and timeslot ...
   int matches = evrPatternCheck(beamcode, timeslot,
               inclusion_a, exclusion_a,
               modifier_a);
... check matches - 0 = no match, 1 = match ...
```

X - Consideration for the multiple beam program

Now, we have mulitple beam programs. Each beam program has a different event configuration.
Thus, an ioc engineer needs to choose correct evrEvent database for thier own application.
And, ioc engineer also need to choose correct timeslots for the each beam program.
Please, look at the following tables.

Beam program	evrEvent database	Active Time Slots TS1ST	TS2ND
LCLS	evrEventAll.db	1	4
FACET	evrEventFACET.db	2	5
XTA	evrEventXTA.db	3(or 0)*	6

Please follow the above table in your <application>pattern.substitutions

(Remark *) The XTA beam has maximum rate 10 Hz. If you really don't need 120 Hz timestamps,
you can drop one of the active timeslots which should be timeslot 3,
becuase, the beam runs on the timeslot 6.

Each beam program has different event codes and different concerns about the timeslots.
For example FACET has 3 custom event code, inject-201, scavenger-202, and beam event-203 (**).
These should be refleced on the trigger configuration in the individual application.
Please, update your <application>evr.substitutions file. You can find "file evrEventCtrl.db" section
in the file and look at the "ID" macro which means the event number.

You also need to reflect the different concern about the timeslot.
FACET beam runs on the timeslot 5 instead of the timeslot 4.
So, you may need to change each 4x event numbers to 5x.

Same considerations are required for the XTA. The XTA own event codes are the followings.
RF full rate - 201, Beam full rate -202, Beam rate limited - 203 (**)
You may want to use event number 6x for the XTA due to the beam runs on timeslot 6.

XI - Consideration to use the event invariant delay

We have made changes on the evrDevTrg.db. This database has new macros: TOUT and ACTV.
We need to put those two macros into the section for the evrDevTrig.db
in the "<application>trig.subsititutions" file.

The TOUT macro provides mapping between the output channel and the event invariant logic.
We can use OUT0, OUT1, ..., OUTA, OUTB, OUTC to describe the output channel.
Actually, front panel trigger outputs share the events with the real panel triggers.
So, you can use, OUT0, OUT1, OUT2 for the front panel.

XII - Consideration for Autosave

The event module supports two ways to use autosave.

```
    # autosave/restore settings
    # this session should be located after loading up the database

    save_restoreSet_status_prefix( "<<EVR_NAME>>")
    save_restoreSet_IncompleteSetsOk(1)
    save_restoreSet_DatedBackupFiles(1)

    set_requestfile_path("/data", "autosave-req")
    set_requestfile_path("./")
    set_savefile_path("/data","autosave")

    # To restore from the sav file which is generated by the INFO field
    set_pass0_restoreFile("info_position.sav")
    set_pass0_restoreFile("info_setting.sav")


    # This session should be located very last in your start up script
    # Make request file
    chdir("/data/autosave-req")
    makeAutosaveFiles()
    # Set monitoring for autosave
    create_monitor_set("info_positions.req", 5, "DEV=<<EVR_NAME>>")
    create_monitor_set("info_settings.req", 30, "DEV=<<EVR_NAME>>")
```

XIII - Consideration for the Linux PMC EVR

---

1) Synchornous signal catch for the EVR interrupt
The linux driver generates signal (SIGIO) when the interrupt occurs.
In the prvious version, the signal is proceeded by _MAIN_ thread with
the registered signal handler. But, the _MAIN_ thread does not have RT priority,
it drags the real-timer performance for the interrupt handling. As a result,
it makes many issues on processing on the evrTask and the evrRecord.
To avoid this issue, we added highest RT priority thread and the new thread handles
the signal synchornousely. To use this new feature, we need to add the followings
in the <application>Main.cpp file to block the signal for the _MAIN_ thread.

```
int main(int argc,char *argv[])
{
    sigset_t set;

    /*
     * if it is possible, proceed the memory locking
     * to avoid real-time peformance dragg do to swapping.
     */
    #if _POSIX_MEMLOCK > 0
    if(mlockall(MCL_CURRENT | MCL_FUTURE) == -1) {
        printf("Fatal error: memory locking fail\n");
        return -1;
    }
    #endif

    /* Blocking SIGIO signal for the _MAIN_ thread */
    sigemptyset(&set);
    sigaddset(&set, SIGIO);
    pthread_sigmask(SIG_BLOCK, &set, NULL);

    if(argc>=2) {
        iocsh(argv[1]);
        epicsThreadSleep(.2);
    }
    iocsh(NULL);
    epicsExit(0);
    return(0);
}
```

2) erapi chages
MRF provides APIs for both EVR and EVG. These APIs are included in the followings files.
    erapi.c

erapi.h

We have put a new thread (irqHandler thread) for linux platform.
This thread receive the SIGIO signal sychronousely from the kernel module, and does the irq handling.
We have modfied the erapi.c for this new thread.

For upgrading the APIs, Please, follow the following procedure.

- copy above files into mrfApp/src directory
- add the following line into very top in the erapi.c

```
    void EvrIrqHandlerThreadCreate(void(*handler)(int));
```

- make chage for the EvrIrqAssignHdler()

```
void EvrIrqAssignHandler(volatile struct MrfErRegs *pEr, int fd,
                         void (*handler)(int))
{
  struct sigaction act;
  int oflags;
  int result;

/*

  act.sa_handler = handler;
  sigemptyset(&act.sa_mask);
  act.sa_flags = 0;

  result = sigaction(SIGIO, &act, NULL);
  printf("sigaction returned %d\n", result);
*/

  EvrIrqHandlerThreadCreate(handler);

  fcntl(fd, F_SETOWN, getpid());
  oflags = fcntl(fd, F_GETFL);
  fcntl(fd, F_SETFL, oflags | FASYNC);
  /* Now enable handler */
  EvrIrqHandled(fd);
}
```

XIV - General BSA template

Exising BSA templates are depended on a specific system.
There is demand to get a new BSA template which is not depend on a specific system,
is also have more flexibility to adjust PV fields.
So, we made a new general BSA template which named "bsaGENEdef.db."
IOC engineer has to set the following macros on his/her subsitutuion file to use the
new template.

```
    $(DEVICE)    device information, prefix of PV name
          ex) TCAV:LI28:800
    $(ATRB)        attibute name for the BSA PV
    $(IN)        PV name for the data source
    $(EGU)        Engineering Unit
    $(HOPR)        High operation range
    $(LOPR)        Low operation range
    $(PREC)        Precision
    $(ADEL)        Archive Deadband
    $(FLNK)        forward link to hook a processing chain after the BSA processing
          if you don't want to use it, Please, put double quote into subsitution file
          to make an empty allocation for the forward link.
```

XV - Library and DBD files for EVG applicaton

RTEMS/VME

Library	DBD
evrSupport	
devMrfEg	devMrfEg.dbd
devMrfEr	
mrfVme64x	

XVI - Library and DBD files for EVR application

RTEMS/VME&PMC

Library	DBD
evrSupport	evrSupport.dbd
devMrfEr	devMrfEr.dbd
mrfVme64x	

LINUX/PMC&PCI

Library	DBD
evrSupport	evrSupport.dbd
devMrfEr	devMrfEr.dbd
drvMrf	