LCLS EVG Upgrade Design (DRAFT-5)

Till Straumann

2/3/2011

1 Introduction

This document describes a proposed interface and design of the software which shall implement the EVG Upgrade Requirements [TH10].

Due to the limited resources available for this project and the complexity of the existing LCLS timing system it is desirable to keep modifications to the existing system minimal.

The new software is intended to remove most of the dependency on information that is currently distributed on PNET and allow the LCLS EVG to support all mission-critical timing patterns and events etc. even if the SLC-MPG is absent.

The solution should be kept simple, modular and extensible so that at least the most critical features can be made available within a few months. Yet, the design must be generic enough to give the user sufficient flexibility for creating the timing-patterns he/she needs.

2 Architecture

We propose to create a "PNET emulation module" or "Pattern Bit Generator" (PABIG) which essentially would replace (or live alongside) the PNET receiver hardware and driver. Most parts of the existing EVG software would remain in place without any modification and it would be completely transparent whether a PNET packet originates in the emulation or comes from real PNET.

This means that most of the semantics of the SLC system, most notably the notorious "3-timeslot delay" feature are preserved and that (as called for by the [TH10]), optionally, the PABIG can be synchronized to the SLC MPG.

The PABIG is driven by 360Hz interrupts from the Dusatko VME card and computes a new PNET packet from a variety of input sources:

1. Time-slot information from the Dusatko module. It is an open question how it is ensured that this information is in synch with the SLC-MPG (wiring to the correct AC-phase and slope etc.).

2. Internal state machine (for PulseID, .5Hz markers). This state machine can be synchronized to the SLC-MPG.

   Note

   Note: Recent email from Tony G. suggests that the SLC-MPG does *not* synchronize the slow features (pulse-id and .5Hz marker) with a particular time-slot. This subject might need more attention in order to avoid subtle issues.
   The current LCLS EVG implementation does propagate the SLC pulseID (albeit 1 cycle off) and MOD720 bits but synchronizes the LCLS specific base rates to TS1.
   The PABIG shall – while synchronized to PNET/MPG (see [TH10], req. 9.) – propagate the pulseID and MOD720 "verbatim" to the existing LCLS EVG software. If the PNET/MPG is either absent or synchronization disabled so that the PABIG produces the pulseID and MOD720 then the PABIG shall synchronize these bits with TS1 and the period of its own pattern generator.

3. User input buttons (hard- and software).
4. Hardware signals and/or EPICS PVs (e.g., should BCS be "wired" to the EVG or additional inputs on the Dusatko card?).
5. User-configurable pattern configuration database (see below).
6. "Real" PNET.

Note that MPS is explicitly *not* included since any MPS information going into PABIG would inevitably pass through the "pattern pipeline" and thus be unnecessarily delayed.

The MPS information is already propagated in MOD6 which bypasses the pipeline and is thus available faster. Applications can read MOD6 or incorporate it into any desired event code's inclusion/exclusion masks.

2.1 Pattern Configuration

For the following discussion we shall define a few terms:

Pattern-rate A "pattern-rate" or "rate" is a sequence in time of 128-bit PNET patterns. The sequence has a finite length (*RSI_MAX*, see below) but repeats periodically.

Rate-group selector PABIG shall support a small but extensible number of user-programmable rates. These rates are logically grouped into (*GSEL_MAX*) "rate-groups" with *RSEL_MAX* group members.

A "rate-group selector" is a small number (0..*GSEL_MAX*) that identifies a particular rate-group.

Rate-selector A rate-selector is a small number (0..*RSEL_MAX*) that is associated with a user-input element (hard- and/or software "button"). The association is user-programmable.

A rate-selector is used to pick a particular rate from a rate-group.

Rate-sequence index ("RSI") A counter which is synchronized to a defined feature (e.g., first time-slot marked TS1 which follows or coincides with *pulseID* modulo *RSI_MAX* == 0)[1]. The RSI counter wraps after *RSI_MAX* cycles, e.g., 2s.

The central parts of the PABIG are a number of user-programmable structures:

- An array of 6 rate-group selectors defining a periodic sequence of rate-groups assigned to each individual 360Hz time-slot. Each array element defines the rate-group to be active on the current time-slot.
- An array holding the currently active rate-selector for each rate-group. The array has the length *GSEL_MAX*.

  User input elements ("buttons" or "menus") deposit the desired rate-selector for each rate-group into this array using a "priority-based" scheme: Among multiple elements that may control a particular rate-group (UI elements, MPS or BCS inputs etc.) the one requesting the (numerically) smallest rate-selector prevails.
- A user-programmable, three-dimensional array of 128-bit PNET patterns which is addressed by
  1. Rate-sequence index
  2. Rate-group selector
  3. Rate-selector

2.2 Basic Algorithm

These data structures allow PABIG to produce PNET patterns according to the following (simplified) algorithm:

1. Obtain times-slot information from VME module.
2. Try to read a pattern from the PNET card.
3. Compute new pulseID and .5Hz marker and verify against PNET data.

   Note

   Note: If – as reported by Steph – there is currently indeed a 1-cycle offset between SLC and LCLS pulse IDs then we plan to fix this as part of the project

   .

   Resolve possible conflicts according to user "MPG resync policy" selection.
4. Compute new RSI.
5. Fetch rate-group selector for current time-slot. If the rate-group selector matches a special "NULL"-group then omit go to step 7.
6. Read rate-selector from "active rate" array by indexing with rate-group selector.

[1] This makes sense under the assumption that *RSI_MAX* devides the maximal *pulseID*.

7. If the rate-group or rate selector matches the "NULL"-group or "NULL"-rate, respectively, then use an all-zero PNET pattern and go to step 9 Otherwise, fetch PNET pattern for RSI, rate-group selector and rate-selector.
8. Merge hardware and software (read via EPICS Channel-Access) bits (e.g., BCS input) to user-definable position with user-definable polarity into pattern. For each bit as many mappings as supported rate-groups are available so that the mapping may be rate-group specific. There is a finite number of such bits supported (0..*INP_MAX*).

   There are two modes for mapping hardware inputs into a pattern.

   Copy Mode clears or sets a (programmable) bitmask in the pattern if the input is inactive or active, respectively.

   Mask Mode clears a (programmable) bitmask in the pattern if the input is inactive but uses the unaltered bits from the rate while active.
9. If no valid PNET information is available then skip to step 13.
10. Verify that all bits selected by a user-programmable mask ([COMMON_MASK]) are identical in both, the original PNET pattern and the one provided by PABIG. If a mismatch is detected then flag the state as "LOST" or "WAIT" (depending on the synchronization mode; in "AUT" mode use the latter, otherwise the former) and proceed to step 13.
11. Check the beam-codes provided by PNET and the PABIG, respectively. If both beam-codes are nonzero but are different then flag the state as "LOST" or "WAIT" (depending on the synchronization mode; in "AUT" mode use the latter, otherwise the former) and proceed to step 13.

    If PNET provides a non-zero beam-code and PABIG a zero beam-code then the code provided by PNET is inserted into the pattern.
12. Merge "real PNET" bits into pattern using a user-definable mask (0: use bit from PABIG, 1: use bit from "real PNET").

    If no valid PNET information is available then this step is skipped and therefore the original bits from PABIG are used. The user should consider this when programming rates.
13. Merge time-slot, pulseID synchronization bits, .5Hz marker into pattern. Note that the algorithm described here performs this step even for the "NULL" rate.

It should be kept in mind that although the user may configure all 128 bits of the PNET patterns, PABIG and the EVG software implicitly *override* some specific bits.

2.3 Example

As a simple example we illustrate the setup of the relevant arrays for the case of two rate groups:

LCLS The "LCLS" rate-group shall use beam-code 1 and feature three rates 0Hz, 10Hz and 120 Hz.

NLCTA featuring two rates: 0Hz and 60Hz using beam-code 5.

The "beam-code" bits in the pattern are not treated differently from other "modifier" bits and can be seen as an example for more generic patterns.

The "LCLS" group shall be effective on time-slots 1 and 4 and "NLCTA" on time-slot 2.

We assign "LCLS" and "NLCTA" the rate-group selectors 1 and 2, respectively. Rate-group selector 0 shall be defined as the "NULL" group.

Hence, the 6-element array associating a rate-group with each time-slot would be set to $1, 2, 0, 1, 0, 0$ so that group "LCLS" is effective on TS1 and TS4 whereas "NLCTA" is effective on TS2. The pattern emitted on all other time-slots is all-zeroes ("NULL"-group).

Next, we assign the "LCLS-10Hz" and "LCLS-120Hz" rates the rate-selectors 1 and 2, respectively. We use the "NULL" rate selector 0 for the 0Hz rate. We now show the "LCLS" rate-group as a two-dimensional matrix of beam-code values (representing the full 128-bit pattern) which is indexed by rate (rows) and RSI (columns). Only a few columns of this matrix are shown. Note that *RSI* modulo 6 + 1 = *time_slot*.

RSI	0	1	2	3	4	5	6	7	...	36	37	38	...
TS	1	2	3	4	5	6	1	2	...	1	2	3	...
Rate	Beam-Code/Pattern												
10Hz	1	0	0	0	0	0	0	0	...	1	0	0	...
120Hz	1	0	0	1	0	0	1	0	...	1	0	0	...

In the same way we use rate-selector 1 for "NLCTA-60Hz" and 0 for the zero rate. The rate-group matrix for "NLCTA" then looks like this:

RSI	0	1	2	3	4	5	6	7	...	36	37	38	...
TS	1	2	3	4	5	6	1	2	...	1	2	3	...
Rate	Beam-Code/Pattern												
60Hz	0	5	0	0	0	0	0	5	...	0	5	0	...

3 Interface

The PABIG is controlled (and monitored) by means of EPICS PVs. There are static, compile-time limits for *RSI_MAX* and the number of rate-groups (*GSEL_MAX*) and rate-selectors (*RSEL_MAX*) as well as for the number of input bits (0..*INP_MAX*).

The format of one 128-bit PNET pattern is fixed in order to ensure compatibility with the existing event-system and SLC software. 128-bit PNET patterns and bitmasks are represented by a sequence of four 32-bit words starting with *MOD1*.

The following PVs shall be available. Note that the names given here specify only the "attribute" part according to the LCLS PV naming convention.

All PVs which define 128-bit patterns (except for arrays of such patterns, e.g., proper pattern-rates) also have four associated "longout" records as a convenience for EDM screen designers. Since no widget is available to set a 128-bit entity these associated records allow for

writing individual 32-bit words into the 128-bit pattern. The "longout" records use the same name as the original PV but shall have a suffix "_MOD1"..."_MOD4" appended, e.g., "COMMON_MASK_MOD1"..."COMMON_MASK_MOD4". All these five PVs are to be kept in sync, i.e., when one is changed by the user the others are automatically updated by the software.

3.1 Programming Rate-Groups

For each PV named 'RG01_xxx' in the list below there are in fact *GSEL_MAX* instances ('RG02_xxx', ...), one for each supported rate-group – except for group 0 (the 'NULL' rate-group) which has no associated controls for modifying the all-zero pattern (non-group specific items such as real-PNET bits, pulseID, timeslot etc. are still generated; see 2.2).

For each PV named 'RG01_I0_xxx' in the list below there are in fact $GSEL\_MAX * INP\_MAX$ instances ('RG01_I1_xxx', ..., 'RG_02_I0_xxx', ...). The attributes supported for each input can be programmed for each rate-group individually.

The user is assumed to appropriately program all PVs belonging to a particular rate-group while that group is *unused*, i.e., it's associated selector is *not present* in **TS_RG**.

3.1.1 List of PV-attributes

TS_RG A waveform holding 6 rate-group selectors. In addition, six individual PVs **TS1_RG**..**TS6_RG** are available which provide access to individual elements of the waveform. Logic is in place which ensures coherence of the seven PVs.

RG01_NAME An informational name that can be assigned to a rate-group. This PV can be used by edm screens to make them more "operator-friendly".

RG01_PATTRNS A waveform holding $RSEL\_MAX * RSI\_MAX$ PNET patterns. The patterns are stored so that the pattern for *rate_sel*, *rsi* can be addressed by index $(rate\_sel - 1) * RSI\_MAX + rsi$[2]. Individual patterns are stored as 4 consecutive 32-bit, unsigned numbers representing MOD1..MOD4 in this order. Hence, the linear array layout of this waveform is as follows:

[2] rate-selector zero is special and designates the "NULL" rate which has no user-programmable patterns but uses an all-zero pattern, see 2.2

rate_1_rsi_0_mod1
rate_1_rsi_0_mod2
rate_1_rsi_0_mod3
rate_1_rsi_0_mod4
rate_1_rsi_1_mod1
rate_1_rsi_1_mod2
rate_1_rsi_1_mod3
rate_1_rsi_1_mod4
...
rate_n_rsi_0_mod1
rate_n_rsi_0_mod2
rate_n_rsi_0_mod3
rate_n_rsi_0_mod4
rate_n_rsi_1_mod1
rate_n_rsi_1_mod2
rate_n_rsi_1_mod3
rate_n_rsi_1_mod4

RG01_DESRATE "rate-menu" with *RSEL_MAX* + 1 entries (including an entry for the NULL/zero rate). The menu entry names are programmable[3]. The "active" menu entry defines the "desired" rate for the rate-group associated with the menu.

RG01_ACTRATE Read-only PV holding the rate selector which is currently active. This may be lower than the rate requested by DESRATE due to a request from a hardware/software input to limit the rate. This is also a menu with programmable names. The user should program the same names used with DESRATE.

RG01_RATE01_NAME..RG01_RATE15_NAME Informational names to be assigned to the rates belonging to this rate-group. The names are automatically propagated to all relevant menu/enum PVs (e.g., **RG01_ACTRATE** etc.).

RG01_RATESRC Read-only PV which identifies the requestor of ACTRATE. The numerical value of the PV ranges from −1 (DESRATE) to 0..*INP_MAX* with non-negative values referring to input bits.

RG01_I0_MODE Menu selecting the operation mode of the particular input (I0) for the particular rate group (RG01). The following modes are defined:

NONE Input bit is ignored.

MASK If the input is asserted then no modification to the pattern is made. If the input is de-asserted then the pattern is ANDed with the bitmask programmed into RG01_I0_MASK.

This mode 'blanks' a set of bits from the user-programmed rate while the input it de-asserted.

COPY The pattern is ANDed with the bitmask programmed into RG01_I0_MASK. If the input is asserted then the one's complement of RG01_I0_MASK is ORed into the pattern.

[3]The PV is a standard EPICS "mbbo" record, i.e., the field-names for the menu entries are ".ZRST", ".ONST", and so on.

This mode 'inserts' the value of the input bit 'verbatim' into an arbitrary position (or multiple positions) of the pattern.

RATE When the input bit is active then the associated rate selector (RG01_I0_RATE) is requested. If it is lower than any other rate selector requested by any other input or by RG01_RATESRC then RG01_I0_RATE goes into effect immediately.

RATE_AND_MASK Both, the rate- and mask-modes (as described above) are in effect concurrently.

RATE_AND_COPY Both, the rate- and copy-modes (as described above) are in effect concurrently.

RG01_I0_MASK 128-bit bitmask used to control blanking and insertion in 'MASK', 'COPY', 'RATE_AND_MASK' and 'RATE_AND_COPY' modes. Unused in other modes. [4]

RG01_I0_RATE Rate selector to be requested while the input is asserted and in 'RATE', 'RATE_AND_MASK' or 'RATE_AND_COPY' mode. Unused in other modes.

RG01_I0_POLR Binary variable with values 'NORMAL' and 'INVERT', respectively. The PV allows the user to logically invert the level of the input (for the rate-group in question).

RG01_I0_BYPS Menu variable with values

NOT_BYPASSED Input is operating normally for this rate group

BYPASSED_DEASS Input is bypassed and treated as if it was de-asserted

BYPASSED_ASSRT Input is bypassed and treated as if it was asserted

PNET_MASK 128-bit bitmask that defines which bits are copied from the "real" PNET pattern. If PNET is unavailable or PNET-synchronization disabled then this mask is ignored and the original bits from the user-programmed rate are used. [5]

COMMON_MASK 128-bit bitmask that defines which bits of both, PNET- and PABIG-provided patterns must match up ([TH10], 9.d.iii). [6]

3.2 PNET Synchronization

Several PVs support synchronization of the PABIG to "real" PNET.

3.2.1 List of PV-attributes

PNET_STATUS The PABIG continuously monitors PNET and tries to keep internal counters for time-slot, modulo-720, and pulseID synchronized to PNET. Note that the PABIG maintains two independent copies of each of these counters: one for PNET the other for PABIG itself.

[4] See the earlier remarks about four additional PVs for programming individual 32-bit words inside the mask.
[5] See the earlier remarks about four additional PVs for programming individual 32-bit words inside the mask.
[6] See the earlier remarks about four additional PVs for programming individual 32-bit words inside the mask.

Only the latter is authoritative and will be merged into the synthesized pattern. Depending on the synchronization mode the PABIG counters are seeded with the PNET counters thus enforcing synchronization.

The **PNET_STATUS** PV reflects the synchronization status of the PNET "version" of these counters. It may assume the following values:

BAD Counters not synchronized to PNET. Their values are undefined.

OK Counters are valid and synchronized to PNET.

MATCH Counters are valid and synchronized to PNET and in addition they are also synchronized with the corresponding but independent counters maintained by PABIG.

PNET_MODE This is a writeable PV for selecting the PNET synchronization mode. It supports the following "menu" settings:

OFF PNET synchronization disabled. No bits from "real PNET" are merged into the pattern.

MAN Manual PNET synchronization. When synchronization is lost then the PABIG continues to produce patterns but ignores PNET. Re-synchronization requires operator intervention (e.g., writing **PABIG_RESYNC**).

AUTO Automatic PNET synchronization. When synchronization is lost the PABIG continues to produce patterns but simultaneously monitors PNET and once all the relevant PNET information is available (MOD720, TS, PULSEID) then the PABIG is immediately resynchronized to PNET.

PABIG_STATE Synchronization state of the PABIG with respect to PNET. This is a read-back PV for monitoring the current state. In all but 'SYNC' state the PABIG is not synchronized to PNET. This PV may take on the following values:

NONE The PABIG is not synchronized to PNET.

LOST In 'MAN' mode the PABIG goes into 'LOST' state when synchronization is lost. It remains in LOST state (regardless of **PNET_STATUS**) until the mode is switched or a manual resynchronization request is issued (**PABIG_RESYNC**).

WAIT In 'AUT' mode the PABIG goes into 'WAIT' state when synchronization is lost. It remains in WAIT state while **PNET_STATUS** is 'BAD' and automatically transitions to 'SYNC' state as soon as PNET synchronization is gained.

SYNC The PABIG is synchronized to PNET.

Whenever the PABIG transitions into 'SYNC' state (only possible in MAN or AUT mode) the PABIG counters are written with the values of the corresponding PNET counters thus creating a glitch. Note that some state information (e.g., the MOD720 bit in modifier 1) does not produce a glitch immediately but with a considerable delay.

PABIG_RESYNC This is a writeable PV which only is effective in 'MAN' mode. Writing this PV with a non-zero value (or processing it while it holds a non-zero value) issues a resynchronization

request. If PNET_STATUS is not 'BAD' then the PABIG transitions into 'SYNC' state and seeds the internal counters with the PNET values.

3.3 Diagnostics

Several statistics counters provide diagnostic information. The 'xxxERRS' counters are scanned (provided that the record's SCAN field is set to 'I/O Intr') when they change so that the record timestamp information approximately reflects the time of the last change/error. The same applies to the **PABIG_STATE** and **PNET_STATUS** PVs.

3.3.1 List of PV-attributes

PNET_TSMISMS If the PNET timeslot information does not match the timeslot as maintained by PABIG then **PNET_STATUS** is flagged as BAD and **PNET_TSMISMS** is incremented.

PNET_720ERRS If the PNET MOD720 synchronization bit is not observed when the internal mod720 counter wraps around then **PNET_STATUS** is flagged as BAD and **PNET_720ERRS** is incremented.

PNET_TSERRS If the PNET timeslot information does not match the internally maintained counter then **PNET_TSERRS** is incremented and the counter reset to the value shipped on PNET.

PNET_BCERRS If the PNET beam-code information does not match the PABIG beam-code and both are non-zero then **PNET_STATUS** is flagged as BAD and **PNET_BCERRS** is incremented.

PNET_IPERRS If the PNET information has the MPG_IPLING bit set then **PNET_STATUS** is flagged as BAD and **PNET_IPERRS** is incremented.

PNET_SYNLOST Counts the number of occurrences when PNET synchronization was lost.

PNET_PIDERRS If the PNET pulseID resynchronization information does not match the internally maintained pulseID counter then **PNET_STATUS** is flagged as BAD and **PNET_PIDERRS** is incremented.

PNET_AUTERRS In AUT synchronization mode it is theoretically possible that while the PABIG is synchronized to PNET a glitch is detected in the PNET information. Such a glitch would propagate to the PABIG without ever leaving SYNC state. The **PNET_AUTERRS** PV counts these glitches.

PABIG_TSERRS If the time-slot information read from the "Dusatko-module" does not match the internally maintained time-slot counter then the internal counter is set to the value read from the VME module and **PABIG_TSERRS** is incremented.

PABIG_PULSID Pulse-id as maintained by the PABIG.

PNET_PULSID Pulse-id as received from PNET.

PNET_SYNERRS Bitmask of errors that caused PNET synchronization to be lost. Multiple bits may be set when an error is detected.

Note that the bits are 'latched' when synchronization is lost but not updated afterwards. Thus the first event causing the error is recorded and subsequent errors are ignored.

`0x0001` No PNET data received.

`0x0002` PNET packet had IPLING set.

`0x0004` Timeslot reported by PNET doesn't match reading of the VME card.

`0x0008` MOD720 synchronization was lost.

`0x0010` pulseID synchronization was lost.

`0x0020` MOD720 counter is INVALID, i.e., not yet synchronized.

`0x0040` PNET pulseID is INVALID, i.e., not yet synchronized.

`0x0080` Unable to synchronize to PNET (unknown error).

`0x0100` Timeslot received from PNET invalid (not in 1..6).

`0x0200` PABIG- and PNET-provided bits requested by [COMMON_MASK] do not match.

`0x0400` PABIG- and PNET-provided beam-codes are both nonzero but don't match.

PABIG_HW_I0_STATE (Read-only PV). Reflects the state of hardware input I0. Note that *INP_MAX* + 1 of these PVs are available, one for each hardware-input bit.

PABIG_HW_I0_NAME Informational name that can be assigned to a hardware-input bit. It is automatically propagated to all relevant menu/enum PVs.

PABIG_BCx_RATE Rate (in Hz) at which all of the defined beam-codes (x = 0..31) appear. There is one PV per beam-code. Beam-codes less than 10 have no leading zero in the PV name, i.e., they are represented by a single digit.

PATTERN_OUT A diagnostic waveform that stores 720 patterns as sent by the EVG. Note that this PV is strictly not a feature of PABIG but of the EVG since it holds patterns *after the EVG software applied modifications and additions. The waveform holds a sequence of patterns consisting of* 6 *32-bit words, namely MOD1..MOD6.*

Acquisition of this waveform is triggered by writing a non-zero value to **PATTERN_OUT.RARM**. *The software then synchronizes pattern-acquisition with then next arrival of a MOD720 marker. At this point, the waveform's time-stamp is written with the EVG's time-stamp and pattern-acquisition starts.*

The record processes in asynchronous fashion, i.e., it completes "phase two" when the waveform fills up. The record time-stamp (provided that the TSE field is set to −2*) holds the EVG's time-stamp of the first pattern in the waveform.*

There will be further diagnostics PVs but they are mostly useful for low-level debugging and not be incorporated into the standard UI.

3.4 Security

The "EPICS Access Security" facility shall be used to protect critical PVs against inadvertent modifications. Three ASGs are used to classify the aforementioned PVs "into security groups".

PABIGRGMOD comprises PVs which are related to programming patterns and rate-groups: **TS_RG**, **RGxx_PATTRNS**, **PNET_MASK**, **COMMON_MASK**.

PABIGINMOD comprises PVs that define the operational mode of input bits: **RGxx_Iy_MODE**, **RGxx_Iy_MASK**], **RGxx_Iy_RATE**, **RGxx_Iy_POLR**].

PABIGOPS comprise PVs that can be switched by operators during normal operation: **RG01_DESRATE**, **RGxx_Iy_BYPS**], **PNET_MODE**, **PABIG_RESYNC**.

"Helper"-PVs that access individual words in a small array (e.g., **PNET_MASK_MOD3**, **TS1_RG**) shall belong to the same ASG as the associated array PV.

Any PV not listed above belongs to the "DEFAULT" group which is read- and writeable by anyone with network access to the IOC.

References

[TH10] T. Himel, *EVG Upgrade Requirements*, 1/21/2010.