MRF Timing System IOC Status

M Davidsaver[1],J Shah[1],E Bjorklund[2]

NSLSII Brookhaven National Lab[1]
LANCSE Los Alamos National Lab[2]

EPICS Collaboration Meeting Spring '12

Outline

Timing Background

Current Developments

In Depth

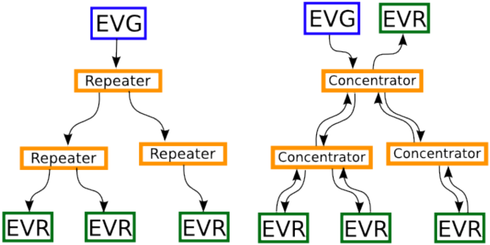
Terms

- Event
  - A point in time. Often defined in relation to another point.
- Code
  - An 8-bit number used to identify an event
- EVG
  - Event Generator - Broadcasts event codes
- EVR
  - Event Receiver - Decodes events and takes local actions
- MRF
  - Micro Research Finland Oy - http://www.mrf.fi/

Architecture

Components

- EVG
- EVR
- Repeater
  - Hub
- Concentrator
  - Switch



Configuration 1

Configuration 2

Synchronization

- Generator (EVG) accepts input from external RF clock (no PLL)
- 8b10 encoding (16-bit frame)
  - Event link bit rate 20x event code rate
  - $500\text{ MHz RF} \div 4 = 125\text{ MHz event} \times 20 = 2.5\text{ GHz link}$
- 8-bit event code, 8-bit data (Distributed Bus)
- Each Receiver (EVR) has a PLL tuned ± 20 *ppm*(10 kHz @ 500MHz)
- Dynamic tuning possible

Global Time Distribution

- Timestamp in two parts: seconds+counter
- Seconds distrubuted as 32-bit unsigned integer
- Counter driven by Event clock, Distributed Bus bit 2, or event code 0x7d
- One event code loads seconds and zeros counter
- Use PPS from GPS receiver

Plans for NSLSII

- EVG in main computer room with fanouts to all 30 cells, RF, and injector buildings.
- All pulls have same length.
- Each cell has additional local fanouts
- VME-EVRRF-230 is standard equipment.
  - TTL for general triggers
  - CML for special cases. Output fill pattern. Trigger kickers.
- cPCI-EVRTG-300 + GUNRC-300 to trigger electron gun.
- PMC-EVR-230 in some Linux servers (softIoc hosts)
  - Use PMC to PCIe carrier board (transparent to software)
  - More precise timestamps
  - One local TTL input

Current Status

- NSLS2 Linac installed, beam commisioning in progress.
- Timing EVG in temporary location in Injector Service Building
- 7 EVRs (3 VME, 3 PMC, 1 EVRTG) in use
- No hardware timestamping
  - Facilities doesn't want temporary hole in roof
- So far no major issues
  - A few bugs (all fixed)
  - Tested recovery from unexpected power outage and RF loss

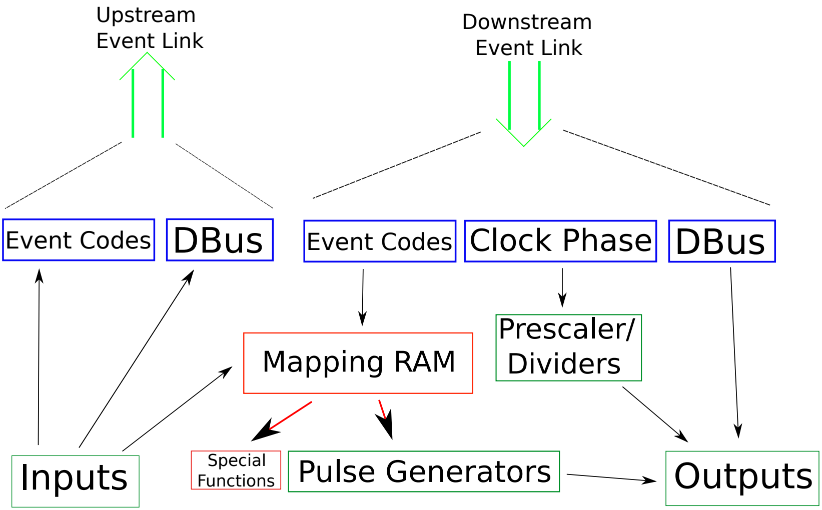
mrfioc2

- Features:
  - Only Base recordtypes
  - As dynamic as possible
  - PCI support via devLib2
- EVR
  - Dynamic mapping (Mapping RAM)
  - Data buffer Tx/Rx (Compatible with 1.x)
- EVG
  - Fully modifible event sequence
  - Timestamp distribution w/o special hardware
- Documentation

Current Status

- EVR
  - Working with prerelease firmware
  - Tested with VME64x, cPCI, and PMC
- EVG
  - VME model working
  - cPCI model not supported (no access to hardware)
- Deployed at BNL for NSLSII teststands (LINAC, BPMs, and PS controllers)
- Version 2.0.1 released 23 April 2012

Receiver Hardware



Receiver Hardware

- Programmable pulse generator
  - Triggered by event code(s)
- Phase locked frequency source (F_{evt}/i)
- Global timestamp receiver
  - Wall clock
  - Event code # received
  - Local input
- Local inputs create timestamps or send upstream
  - Available as: VME, cPCI, and PMC

EVR Mapping Ram

- Many-to-many mapping of event code to function
  - Trigger pulse generator
  - Reset prescalers
  - Timestamp functions
- Most cases 1-to-1 (code 17 triggers pulse gen. 4)
- Some are small-to-small
- Few are many-to-1 (FIFO, Forwarding)

Mapping Records

- One record per pairing
- Default DB maps 3 events

```
record(longout, "pul4:trig1") {
field(DTYP, "EVR␣Pulser␣Mapping")
field( OUT, "@OBJ=EVR1:Pul0,Func=Trig")
field( VAL, "0x40")
}

record(longout, "blk1") {
field(DTYP, "EVR␣Mapping")
field( OUT, "@OBJ=EVR1,Func=Blink")
field( VAL, "0x40")
}
```

Data Buffer

- Buffer reception in two stage. High priority thread reads from hardware places in FIFO. Lower priority thread takes from FIFO and runs callback list.
- Waveform device support to receive. Does endian conversion for multibyte types.
- Plan to use this to distribute fill pattern for NSLSII.

Event FIFO Buffer

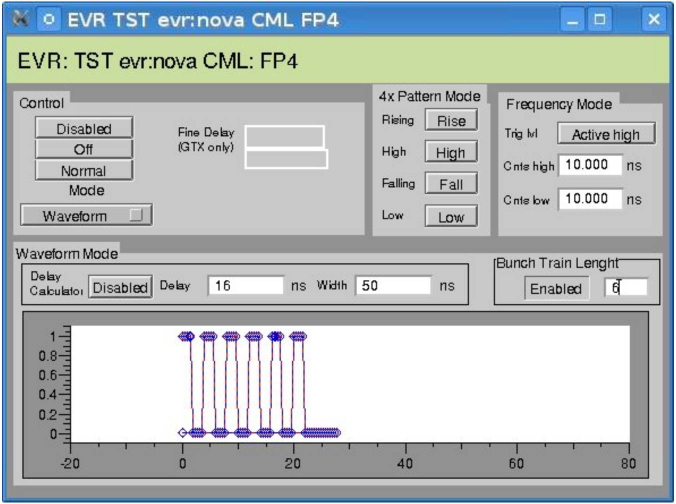
- Arrival of an “interesting” event is recorded in a hardware FIFO buffer.
- I/O Intr scan and callback list.
- longin device support to process on event reception.
- Throttling to prevent too fast events from taking 100% of CPU. Limit buffered events to a given rate. Also, do not run callback list until all previous processing is complete.
  - Can be disabled

Timestamp Validation

- Must prevent invalid timestamps from propogating into generalTime.
- Several times a misconfiguration caused one second tick to be sent too often, or out of sync.
- Firmware bug (now fixed) caused occasional invalid reads.
- EVR must receive 5 sequential updates before it will start using time. Invalid if out of order time is received.

CML/GTX Pattern Outputs

- Higher resolution. 20x EVRRF, 40x EVRTG (effective 8x)
- Output multi-bit patterns



EVR TST evr:nova CML FP4

EVR: TST evr:nova CML: FP4

Control

Disabled

Off

Normal

Mode: Waveform

Fine Delay (GTX only)

4x Pattern Mode

Rising: Rise

High: High

Falling: Fall

Low: Low

Frequency Mode

Trig lvl: Active high

Cnts high: 10.000 ns

Cnts low: 10.000 ns

Waveform Mode

Delay Calculator: Disabled

Delay: 16 ns

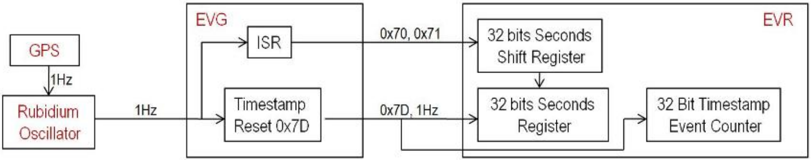
Width: 50 ns

Bunch Train Lenght: Enabled 6

Generator Hardware

- Send periodic event and/or data
- Send event sequences
  - Preset list of times and codes (eg. linac shot or booster ramp)
- Currently VME only, in future cPCI only.

Timestamp



- Synchronize to GPS without custom electronics.
- Off the shelf GPS receiver with NTP server and 1Hz TTL output.
- Buffered with Rubidium oscillator for high precision. Continues running if GPS 1Hz is lost.
- 1Hz send special event code and interrupts CPU
  - Special event code 0x7D marks start of a second (hardware only)
  - Interrupt sends next second bit by bit. POSIX time by default.

EVG Sequences

- Example. Timeline for injection/top off
  - Start insertion kicker ramp up
    - wait 100us
  - Trigger Klystron modulators
    - wait 20us
  - Trigger Klystron
    - wait 500ns
  - trigger e^{-}gun
    - wait 10us
  - Start insertion kicker ramp down

Delay	Code
0	0x10
12500	0x20
2500	0x25
61	0x40
1250	0x12

Note: This is how it looks in hardware

Sequence Use Cases

- NSLSII Booster is $\frac{1}{5}$ diameter or Storage ring.
- Filling/top off process involves multiple injections
- Need to control how many bunches and where they go
- Use timing system to select which sector to fill
  - “Fill Manager” process sets booster extraction delay
  - Move ≥ 1 events
- Allow programatic manipulation w/o complicating client(s)
  - Probably aSub records in a seperate softIOC

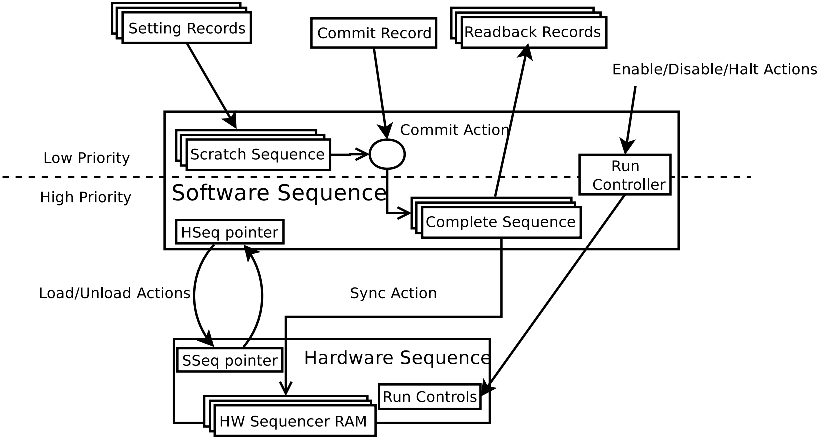
Sequence Representation

- 2 waveforms (codes and times)
    - Clients have to know array index
    - Ordering
- Trigger source/mode
- Control (commit, (un)load, enable/disable)

Sequence Management

- Manage user interactions with sequence ram
- Current hardware supports two independent sequences.
- Single shot or repeating
- Don't modify while running

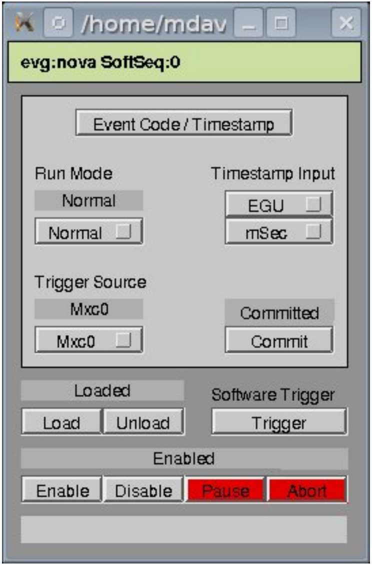
Model


Setting Records
Commit Record
Readback Records
Enable/Disable/Halt Actions
Commit Action
Low Priority
Scratch Sequence
Run Controller
High Priority
Software Sequence
Complete Sequence
HSeq pointer
Load/Unload Actions
Sync Action
SSeq pointer
Hardware Sequence
Run Controls
HW Sequencer RAM

Sequencer Workflow

1. Modify scratch sequence
   - DB/CA operations of individual records (synchronous device supports)
   - CA put w/ callback
2. Commit
   - Single DB/CA operation
   - Updates complete sequence
3. Sync
   - When loaded, or at end of run if already loaded
   - Automatic

Interface



/home/mdav

evg:nova SoftSeq:0

Run Mode

Normal

Timestamp Input

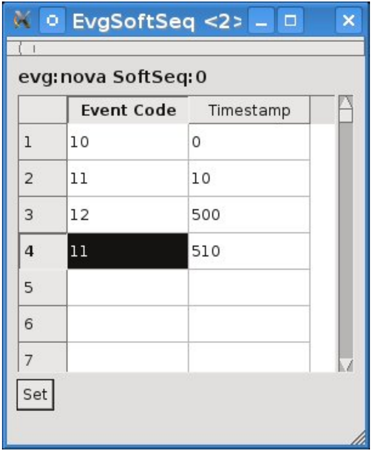
Trigger Source

Mxc0

Committed

Loaded

Enabled

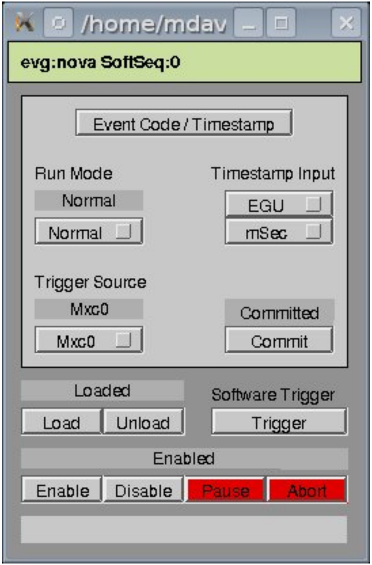


EvgSoftSeq <2>

evg:nova SoftSeq:0

	Event Code	Timestamp
1	10	0
2	11	10
3	12	500
4	11	510
5		
6		
7		

Sequence Control



/home/mdav

evg:nova SoftSeq:0

Run Mode
Normal

Timestamp Input

Trigger Source
Mxc0

Committed

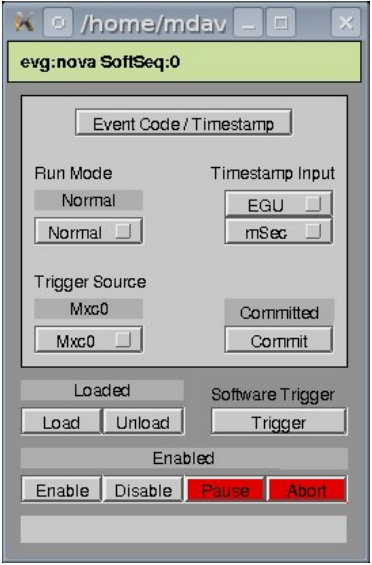
Loaded

Software Trigger

Enabled

- Run Mode
  - **Single**
  - Disarm after one run
  - **Normal**
  - rearm after each run
  - **Automatic**
  - continuous run
- Trigger Source
  - For Single and Normal
- Units
  - Meaning of time delay
- Commit
  - Propogate changes to hardware

Sequence Control (2)



/home/mdav

evg:nova SoftSeq:0

Run Mode: Normal

Timestamp Input: EGU, mSec

Trigger Source: Mxc0

Committed

Loaded

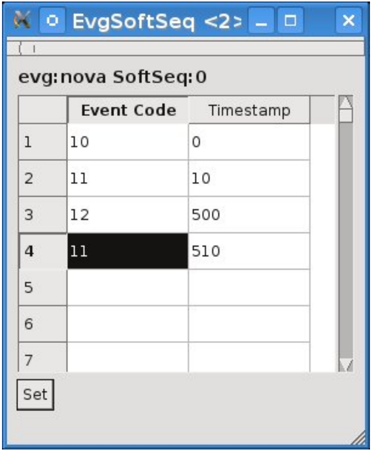
Software Trigger

Enabled

- Load/Unload
  - (De)Allocate hardware resources to run this sequence
- Enabled
  - Trigger permit
- Disable
  - Prevent further triggers. If already triggered, run to completion
- Pause
  - Stop running sequence w/o reset.
- Abort
  - Immediately halt

Interface

Specify sequence. Units of Timestamp defined for each sequence.
Note: Pictured is a small PyQt+cothreads script to allow editing sequence waveforms in a table.

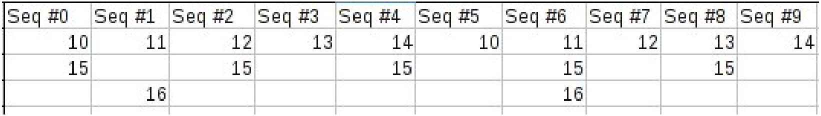


evg:nova SoftSeq:0

	Event Code	Timestamp
1	10	0
2	11	10
3	12	500
4	11	510
5		
6		
7		

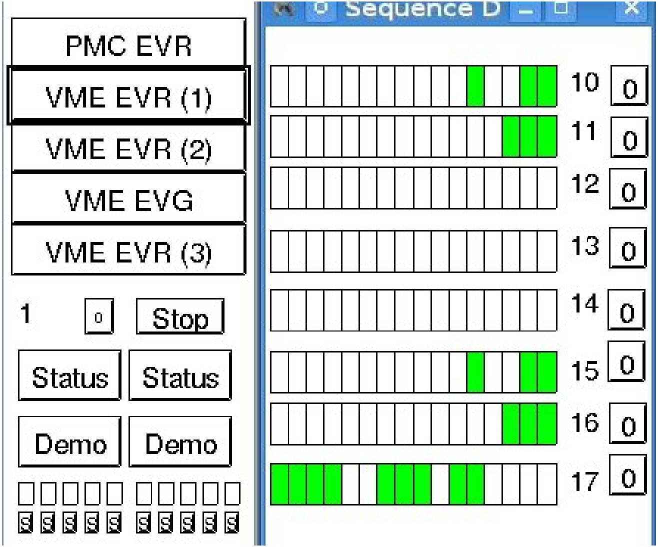
Super Sequencer Demo

- Several Linac FEL machines want to reconfigure between shots.
- Also have repetition rates ~100Hz (10ms period)
- Demo attempts to show how soft sequence interface could be used
- Cycle through 10 predefined sequencer configurations at 100Hz



Seq #0	Seq #1	Seq #2	Seq #3	Seq #4	Seq #5	Seq #6	Seq #7	Seq #8	Seq #9
10	11	12	13	14	10	11	12	13	14
15		15		15		15		15	
	16					16			

Demo Interface



											🟩			🟩	🟩	10	0
													🟩	🟩	🟩	11	0
																12	0
																13	0
																14	0
											🟩			🟩	🟩	15	0
													🟩	🟩	🟩	16	0
🟩	🟩	🟩	🟩			🟩	🟩	🟩		🟩	🟩					17	0

Database Processing

```
record(longout, "$(P)Evnt-SP") {
  field(DTYP, "EVR␣Event")
  field(OUT , "@OBJ=EVR1,Code=17")
  field(SCAN, "I/O␣Intr")
  field(FLNK, "$(P)Cnt-I")
}
record(calcout, "$(P)Cnt-I") {
  field(CALC, "A<10?A+1:1")
  field(INPA, "$(P)Cnt-I␣NPP")
  field(OUT, "$(P)Load-Fout_.SELN␣PP")
  field(FLNK, "$(P)Load-Sel_")
}
record(calcout, "$(P)Load-Sel_")
{
  field(INPA, "$(P)Cnt-I␣NPP")
  field(CALC, "A<=1?10:A-1")
  field(OUT , "$(P)Unload-Fout_.SELN␣PP")
}
```

```
record(seq, "$(P)Load-Fout_")
{
  field(SELM, "Specified")
  field(DOL1, "1")
  field(LNK1, "$(SEQ1)Load-Cmd␣PP"
  ...
}
record(seq, "$(P)Unload-Fout_")
{
  field(SELM, "Specified")
  field(DOL1, "1")
  field(LNK1, "$(SEQ1)Unload-Cmd␣F
  ...
}
```