# 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 Spr '10

# Outline

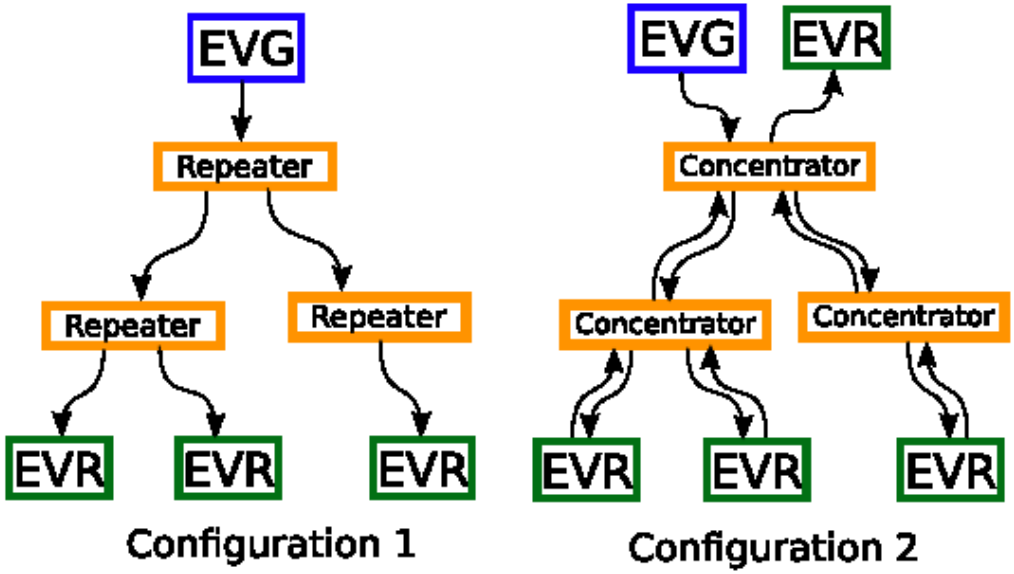
# Terms

- 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



# Synchronization

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

# Recent History

- EPICS support for MRF hardware in use at SLS, Diamond, etc.
- Designed around custom records (eg, egevent, er, erevent)
  - Specific to current register map
  - Not dynamic
  - EVG event sequences inflexible
- MRF “cleans up” register map (make room for future growth)
- Opportunity to “do it right this time”

# Rev. 2

- Goals:
  - Only Base recordtypes
  - Be Dynamic
  - PCI/VME64x support for devLib
- EVR
  - Take advantage of new hardware features (Mapping Ram)
- EVG
  - Fully modifible event sequence
- Write Documentation!

# Current Status

- EVR
  - Works with prerelease firmware
  - Tested VME64x and PMC (cPCI not available)
  - TODO
    - Timestamp distribution (generalTime)
    - Data buffer Rx/Tx
    - Bumpless reboot
- EVG
  - Starting driver development
    - Found some HW bugs
  - Starting on sequence compilation
  - TODO
    - create/load a sequence

# 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-many (FIFO, Event log)

# Mapping Records

- One record per pairing
- Default DB maps 3 events

```
record(longout, "pul4:trig1") {
field(DTYP, "EVR Pulser Mapping")
field( OUT, "@C=1,I=4,Func=Trig")
field( VAL, "0x40")
}

record(longout, "blk1") {
field(DTYP, "EVR Mapping")
field( OUT, "@C=1,Func=Blink")
field( VAL, "0x40")
}
```

# 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.

# 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 |

# 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 $\geq 1$ events
- How to allow programatic manipulation w/o complicating client(s)

# Sequence Representation

- 2 waveforms (codes and times)
  - Clients have to know too much (array index)
  - Ordering

# Sequence Representation

- 2 waveforms (codes and times)
  - Clients have to know too much (array index)
  - Ordering
- egevent record
  - Each event has one record
  - Code and Time are fields
  - Usual problems with record w/ several value fields
  - Hard to implement coarse+fine delay transparently

# Sequence Representation

- 2 waveforms (codes and times)
  - Clients have to know too much (array index)
  - Ordering
- egevent record
  - Each event has one record
  - Code and Time are fields
  - Usual problems with record w/ several value fields
  - Hard to implement coarse+fine delay transparently
- Properties
  - Two records: Code and Time
  - All the benefits of egevent
  - Only drawback is lots of records

# Sequence Management

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