MRF Timing System Overview

Timing Workshop EPICS Meeting Fall 2018

Jukka Pietarinen
Micro-Research Finland Oy
Micro-Research Finland Oy -
Company Introduction

• MRF founded in 1985
• Hardware development (and firmware/software)
• History of
  – VME bus products
  – Data acquisition
  – Fibre Optic data transmission
  – Embedded microcontroller systems
    • Motor control/Control system for a lab printing machine
  – Sensor to analyse printing quality for paper
• Current focus on timing systems for particle acceleratos
MRF Experience and Capabilities

• Fibre Optics
  – TAXI chips (125 Mbit/s)
  – Gigabit transceivers
  – Multigigabit transceivers embedded in FPGAs
  – Especially time deterministic applications and low jitter

• FPGAs
  – Xilinx (since the first XC2064)
  – Lattice ECP2M/ECP3
  – VHDL
  – Limited experience Altera

• HW form factors
  – VME / VME64x
  – CompactPCI / PXI
  – PMC
  – CompactRIO
  – PCI Express
  – MicroTCA.4
MRF Timing System Development

- The MRF Timing system originally developed for the Swiss Light Source (SLS), Paul-Scherrer Institut in 1999
- First commercially available timing system for particle accelerators
- Current users include
  - SLS, PSI
  - Diamond Light Source Ltd., U.K.
  - SSRF, Shanghai, China
  - Australian Synchrotron
  - ALBA, Spain (Tango)
  - Elettra, Trieste, Italy (Tango)
  - BEPCII, Institute for High Energy Physics, Beijing, China
  - LCLS, Stanford Linear Accelerator Center, USA
  - KEK, Japan
  - FERMI, Trieste, Italy (Tango)
  - TLS, Taiwan
  - NSLS II, Brookhaven, USA
  - PAL-XFEL, Pohang, South Korea
  - LANSCE upgrade, Los Alamos, USA
  - FRIB, Michigan, USA
  - SwissFEL, PSI
  - STFC, Daresbury
- Future users
  - ESS, Sweden
- Most of the sites listed above are using EPICS
MRF Timing System History

- Functionality based on the APS, Argonne timing system which came from Fermilab
- Redesigned for SLS → Series 100
- Improved performance for Diamond → Series 200
- Timing signals needed for synchronisation of subsystems are applied to Event Generator (EVG) or generated by EVG
- Timing information is converted to 8-bit event codes and distributed to Event Receivers (EVR) as an optical signal
- Event clock rate determines timing resolution:
  - Minimum clock rate 50 MHz, 20 ns resolution
  - Maximum clock rate 125 MHz, 8 ns resolution
- 8-bit distributed bus running in parallel and independent of timing events allows distribution of eight signals updated with the event clock rate
- Further improved performance (7 ns resolution) and Delay Compensation for SwissFEL → DC
What do you need a timing system for?

- Phase and frequency locked clocks at different locations
- Trigger something at different locations at exactly the same time
- Make something happen in sequence with predefined time intervals
- Synchronize the local time at different locations with high precision
- Timestamp events at different locations to analyze what happened first
Operating Principle of the MRF Timing System

- Event based system
  - 256 Event codes, one event code/cycle (7 ns - 20 ns or 50 MHz to 142 MHz)
- In addition to events there are eight signals (distributed bus) that are updated at half the rate (14 ns - 40 ns or 25 MHz to 71 MHz)
- Synchronous data transfers
- Transmission medium: optical fiber
  - typically multi-mode, OM3 or better up to 550 m
  - single-mode, up to 10 km
- Line code or “event stream” based on 8B10B encoding
Event Clock

- Base clock of MRF timing system
- Typically driven from external source e.g. accelerator RF signal
- Has to be in range 50 MHz to 142 MHz (7 ns to 20 ns cycle)
- Has to be provided to the Timing System Master, all other nodes regenerate this clock from the event stream received through the fiber
- The Timing Master has an input divider to divide the provided RF clock by 1, 2, 3, ..., 12, 14, 15, 16, ..., 32.
Acronyms etc. timing jargon

- **EVG** - Event Generator, produces event stream
- **EVR** - Event Receiver, decodes event stream and performs different hardware & software functions
- **Fan-Out** - Splits event stream to several ports
- **Concentrator** - Combines events from several sources into one port
- **EVM** - Event Master, EVG and Fan-Out/Concentrator
Event link protocol carries:
- 8 bit event codes @ event rate max. 142 MHz
- 8 bit distributed data bits @ 1/2 event rate
- Data buffer transfers up to 2 kbyte
- Segmented data buffers up to 128 x 16 bytes
- Max. data transfer rate 71 Mbytes/s
EVG Basic Building Blocks

- Trigger Events
- Multiplexed Counters
- Sequencers
- Event Analyzer
- Software Event
- Time distribution logic
- Configurable Size Data buffer
- Segmented Data buffer
Trigger Event

- A Trigger Event has one event code assigned to it
- Upon a trigger this event code is sent out
- Sources for triggers:
  - External signal
  - Multiplexed Counter edge
  - AC main synchronization logic
Multiplexed Counter

- 32 bit counters running at the event clock rate
- Can produce frequencies from event clock rate/(2^{32} - 1) to event clock rate/2
- Counters can be reset simultaneously by software
- Configurable polarity
Sequencer

- A sequencer sends out events in predefined order with predefined timing
- 2k dual-port memory for 32-bit timestamp, event code and mask
- Operating modes: single, normal (trigger) and recycle
Sequencer triggering

- Multiplexed counter
- AC Synchronization logic
- External trigger
- Software trigger
Sequencer operating modes

- Single mode
  - Plays back sequence once until end sequence event
- Normal mode
  - Plays back sequence and waits for next trigger after end sequence event
- Recycle mode
  - Plays back sequence and immediately restarts sequence from beginning after end sequence event
Event Analyzer

- Tools to record all events sent out by the EVG
- 64-bit counter counting event clock cycles
- 2k memory to store event codes, 64-timestamp and distributed bus state
Sequencer and Event Analyser Example

<table>
<thead>
<tr>
<th>address</th>
<th>timestamp</th>
<th>event</th>
<th>Event</th>
<th>Event Analyser with 64-bit time counter</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0x20</td>
<td>1. Event</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>85168*66</td>
<td>0x2a</td>
<td>2. Event</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>94537*66</td>
<td>0x24</td>
<td>3. Event</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>94538*66</td>
<td>0x25</td>
<td>4. Event</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>94821*66</td>
<td>0x2c</td>
<td>5. Event</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>282002*66</td>
<td>0x30</td>
<td>6. Event</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>283895*66</td>
<td>0x3c</td>
<td>7. Event</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>284000*66</td>
<td>0x7f</td>
<td>End sequence</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Analyser time (s)</th>
<th>Offset (us)</th>
<th>code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0,190072274</td>
<td>-1,07</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>0,190073347</td>
<td>0,00</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>0,235073191</td>
<td>44999,84</td>
<td>2a</td>
<td></td>
</tr>
<tr>
<td>0,240023448</td>
<td>49950,10</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>0,240023977</td>
<td>49950,63</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>0,240173504</td>
<td>50100,16</td>
<td>2c</td>
<td></td>
</tr>
<tr>
<td>0,339073511</td>
<td>149000,16</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>0,340073707</td>
<td>150000,36</td>
<td>3c</td>
<td></td>
</tr>
<tr>
<td>0,390076629</td>
<td>-1,07</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>0,390077702</td>
<td>0,00</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>0,435077546</td>
<td>44999,84</td>
<td>2a</td>
<td></td>
</tr>
<tr>
<td>0,440027803</td>
<td>49950,10</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>0,440028332</td>
<td>49950,63</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>0,440177859</td>
<td>50100,16</td>
<td>2c</td>
<td></td>
</tr>
<tr>
<td>0,539077866</td>
<td>149000,16</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>0,540078062</td>
<td>150000,36</td>
<td>3c</td>
<td></td>
</tr>
</tbody>
</table>

- 264/4 = 66, 528 us cycles
- Line sync. divider 10
- 50 Hz applied to line sync. Input
- Trigger event enabled to send 0x11 on seq. trigger
Time Distribution Logic

- Support for 32 bit Seconds register
- 32-bit sub second register - e.g. 1 us precision can be used
- Seconds value is send out with event codes
- Sub-second clock is send out as an event or on the distributed bus
- 1 PPS and faster clock e.g. 1 MHz have to be supplied externally and these two clock must be rising edge aligned
Configurable Size Data Buffer

- 2k byte dual-port memory
- Transfer size 4 bytes to 2k in 4 byte increments
- Controlled by software
Segmented Data Buffer

- Introduced with Delay Compensation hardware
- 2k byte dual-port memory
- 128 16-byte segments
- Can send a single or multiple segments at a time
- Controlled by software
- Last segment #127 is reserved for delay compensation
  - delay compensation data
  - Topology ID
EVR Basic Building Blocks

- Event Mapping RAM
- Pulse Generators
- Prescalers
- Timestamping logic
- Event FIFO
- Event Log
- Data buffer
- Segmented Data buffer
Event Mapping RAM

- 256 x 128 bit dual-port RAM
- Each event code has 128 bits assigned and these bits define what functions are to be performed

<table>
<thead>
<tr>
<th>Map bit</th>
<th>Default event code</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>127</td>
<td></td>
<td>Save event in FIFO</td>
</tr>
<tr>
<td>126</td>
<td></td>
<td>Latch timestamp</td>
</tr>
<tr>
<td>125</td>
<td></td>
<td>Led Event</td>
</tr>
<tr>
<td>124</td>
<td></td>
<td>Forward event</td>
</tr>
<tr>
<td>123</td>
<td>0x79</td>
<td>Stop log</td>
</tr>
<tr>
<td>122</td>
<td></td>
<td>Log event</td>
</tr>
<tr>
<td>101</td>
<td>0x7A</td>
<td>Heartbeat</td>
</tr>
<tr>
<td>100</td>
<td>0x7B</td>
<td>Reset Prescalers</td>
</tr>
<tr>
<td>99</td>
<td>0x7D</td>
<td>Timestamp reset event</td>
</tr>
<tr>
<td>98</td>
<td>0x7C</td>
<td>Timestamp clock event</td>
</tr>
<tr>
<td>97</td>
<td>0x71</td>
<td>Timestamp Seconds ‘1’</td>
</tr>
<tr>
<td>96</td>
<td>0x70</td>
<td>Timestamp Seconds ‘0’</td>
</tr>
<tr>
<td>95 - 64</td>
<td></td>
<td>Trigger pulse generator 31 - 0</td>
</tr>
<tr>
<td>63 - 32</td>
<td></td>
<td>Set pulse generator 31 - 0 output high</td>
</tr>
<tr>
<td>31 - 0</td>
<td></td>
<td>Reset pulse generator 31 - 0 output low</td>
</tr>
</tbody>
</table>
Pulse Generator
Prescaler

- Programmable counter that can produce different frequencies
- All prescalers are reset by one event code, thus all prescalers on all EVRs can be synchronized
- Can be output directly on I/O or used to trigger pulse generators
Timestamping Logic and Event FIFO

- Load bit '0' event code $70
- Load bit '1' event code $71
- TS event counter reset event code $7D
- Latch timestamp MAP14
- Event FIFO write MAP15
- Event FIFO

<table>
<thead>
<tr>
<th>32-bit Seconds</th>
<th>32-bit Timestamp</th>
<th>event</th>
</tr>
</thead>
<tbody>
<tr>
<td>32-bit Seconds</td>
<td>32-bit Timestamp</td>
<td>event</td>
</tr>
<tr>
<td>32-bit Seconds</td>
<td>32-bit Timestamp</td>
<td>event</td>
</tr>
<tr>
<td>32-bit Seconds</td>
<td>32-bit Timestamp</td>
<td>event</td>
</tr>
<tr>
<td>32-bit Seconds</td>
<td>32-bit Timestamp</td>
<td>event</td>
</tr>
</tbody>
</table>
Event Log

- Similar to Event FIFO, but with ring buffer
- Event FIFO stops when full
- Event Log rolls over and overwrites old events
- Can be used for diagnostics: stop log event e.g. after beam loss, read out log events
Data Buffer

- Receive Data Buffer send by EVG
- Data buffer has to be armed to receive anything
- Data buffer is automatically disable after a received buffer
- Data buffers can be lost if buffer is not read and re-armed before EVG sends next buffer
- Software interrupt upon reception
Segmented Data Buffer

- Receive Segments send by EVG
- Segmented Data buffer is enabled all the time
- Flags for each segment: receive complete, checksum mismatch, overflow flag
- Each segment has a received data bytes counter
- Software interrupt configurable for each segment independently
GTX Outputs

- 40 bits/event clock cycle resolution (200 ps @ 125 MHz event clock)
- Operating modes:
  - pulse
  - pattern
  - frequency
EVM, EVG, EVR

To make it even more confusing...

● The difference between Event Generator and Event Receiver is not that clear anymore as it used to be
● EVG features have been added to EVRs and vice versa
Event Master

- Introduced with the latest series i.e. Delay Compensation (DC) capable hardware
- Can operate as the System Master (Top level) or Fan-Out (lower level)
- Consists of
  - one EVG
  - Fan-Out / Concentrator
  - Two “internal” EVRs
Typical System Layout - Delay Compensation (DC)

Event link protocol carries:
- 8 bit event codes @ event rate max. 142 MHz
- 8 bit distributed data bits @ 1/2 event rate
- Data buffer transfers up to 2 kbyte
- Segmented data buffers up to 128 x 16 bytes
- Max. data transfer rate 71 Mbytes/s
EVM configured as Fan-Out
EVR Simplified Block Diagram
Topology of Simple System mainly for testing purposes without EVG and Delay Compensation
VME-EVM-300

- **Event Generator (EVG)**
  - 2 sequencers with maskable events
    - Max. 2047 events/sequence
    - 32 bit timestamp
  - 8 multiplexed counters
  - data buffer up to 2k bytes
  - segmented data buffer
    - 127 segments, 16 bytes each
- **8-Way Fan-Out/Concentrator**
- **Two Event Receivers (EVR)**
  - 8 internal pulse outputs
  - one sequencer
- **Event rate conversion (with some data buffer related limitations)**
- **Front panel input phase monitoring/select features**
- **Distributed bus phase selection**
- **RF input monitoring**
VME-EVR-300

- Event Receiver (EVR)
- Four Universal I/O slots
  - all compatible with -DLY modules
  - one slot driven by GTX logic
- two differential CML outputs driven by GTX logic
- two TTL inputs for external triggers
- transition board interface for additional eight Universal I/O modules (no support for -DLY modules)
mTCA-EVM-300

- Event Generator (EVG)
  - 2 sequencers with maskable events
    - Max. 2047 events/sequence
    - 32 bit timestamp
  - 8 multiplexed counters
  - data buffer up to 2k bytes
  - segmented data buffer
    - 127 segments, 16 bytes each
- 7-Way Fan-Out/Concentrator
- Two Event Receivers (EVR)
  - 8 internal pulse outputs
  - one sequencer
- Event rate conversion (with some data buffer related limitations)
- Front panel input phase monitoring/select features
- Distributed bus phase selection
- RF input monitoring
mTCA-EVR-300U

- Event Receiver (EVR)
- Two Universal I/O slots
  - both compatible with -DLY modules
- Four LVTTL outputs
- Two TTL inputs for external triggers
- Backplane triggers
- Can drive TCLKA/TCLKB clocks with GTX logic
- RTM interface, no RTM designed yet
mTCA-EVR-300I, PCIe-EVR-300DC

- I/O through microSCSI connector (becoming obsolete)
I/O Breakout Issues

- IFB-300 mechanics becoming obsolete
- Molex cable discontinued
- VHDCI/microSCSI connector is fragile and you have to fix the cable mechanically, cannot rely on screwing in connector
- Alternative ways to get I/O from small front panels?
- Asked HW-talk mailing list for advice – no replies :( 

- Samtec ERC-031-01-0
- Samtec ERI8-031-S-D-RA
- Samtec EPLSP-031-1000
Universal I/O Modules

- **UNIV-TTL**
  - LVTTL output

- **UNIV-TTL5V**
  - 5V TTL output

- **UNIV-TTL-DLY**
  - LVTTL output
  - Delay tuning 1024 steps of ~9 ns

- **UNIV-LVPECL**
  - differential LVPECL output

- **UNIV-LVPECL-DLY**
  - differential LVPECL output
  - Delay tuning 1024 steps of ~9 ns
Universal I/O Modules

- **UNIV-TTLIN**
  - TTL input
  - configurable term.
    - 240 ohm to +5V
    - 50 ohm to GND

- **UNIV-NIM**
  - NIM output

- **UNIV-HFBR-1528**
  - Optical Output

- **UNIV-HFBR-1414**
  - Optical Output
MRF Timing System Delay Compensation Details

Timing Workshop EPICS Meeting Fall 2018
Jukka Pietarinen
Micro-Research Finland Oy
Delay Compensation Implementation

- The Timing System Master sends out a periodic beacon event.
- Each node immediately sends the beacon event back upon reception.
- The loop delay between the sent and “reflected” beacon is measured and this delay value is added up and forwarded to the downstream node.
- Compensation is done at the last stage by adjusting the depth of the delay compensation FIFO to match a programmable target delay.
VME-EVM-300 Configured as Delay Compensation Master
VME-EVM-300 Configured as Delay Compensation Fan-Out
Delay Compensation Capable Event Receiver in non-DC mode

Event Receiver in non-DC mode

- transceiver
- FIFO
- event stream decoder
- RX segm. data buffer
- delay measure
- clock cleaner
- phase control
- loopback beacon event 0x7E
- recovered event clock
- event 0x7E
- event 0x7E
- DC target value

Σ
MRF Timing System Delay Compensation Protocol Details

Timing Workshop EPICS Meeting Fall 2018
Jukka Pietarinen
Micro-Research Finland Oy
Compatibility with the Earlier Protocol

- Delay Compensation extends the protocol by:
  - Using event code 0x7E as beacon event
  - Adds the segmented data buffer that uses a different transfer start “comma character” than the data buffer
- Both protocol changes are ignored by pre-DC event hardware
- Timing is similarly deterministic as with a pre-DC distribution network
- Delay Compensation EVRs can be used with the earlier protocol in non-DC mode
  - The Delay Compensation FIFO can be used to fine tune the EVR output delays as a group with very high resolution
- EVMs require the top node to be configured as the Delay Compensation Master
- EVMs can be used in Delay Compensation Mode only
MRF Event System Protocol

- Based on 8B10B encoding using data characters D00.0 through D31.7 and K-characters K28.0, K28.1, K28.2 and K28.5
- Each event clock cycle two characters are transmitted
  - One “event code” character
  - One “data” character
- Event code slot
  - D00.0 – null event
  - D01.0 through D31.7 – event codes 0x01 through 0xFF
  - K28.5 comma character for synchronization, both 8B10B encoding and event protocol, sent out every 4\textsuperscript{th} event slot instead of character D00.0
- Data slot
  - Multiplexed between 8 bit distributed data bus and data transfers
  - Distributed data bus transfers parallel to K28.5 event slots
MRF Event System Protocol

- Data transfers
  - Configurable size data buffer
    - K28.0 start of data packet
    - Dxx.x, Dxx.x, Dxx.x, ... packet data
    - K28.1 end of data packet
    - Dxx.x, Dxx.x checksum 0xFFFF – sum of data bytes
  - Segmented data buffer (16 byte segments)
    - K28.2 start of segmented data packet
    - Dxx.x start segment number 0x00 to 0x7F (last segment reserved for delay compensation)
    - Dxx.x, Dxx.x, Dxx.x, ... segment data
    - K28.1 end of transfer
    - Dxx.x, Dxx.x checksum 0xFFFF – segment address – sum of data bytes