EUROPEAN SPALLATION SOURCE

Timing system training for ICS integrators

Javier Cereijo Garcia

Integrated Control System Division
ESS, Sweden

https://www.europeanspallationsource.se
August 21, 2018

Outline

- Timing introduction
- System configuration
- EVR: basic start-up
- EVR: basic configuration
- EVR: events and triggers
- EVR: timestamping
- EVR: standalone mode
- EVR: inputs
- EVR: backplane clocks
- EVG: basic start-up
- EVG: basic configuration
- EVG: creating events and data
- Data buffer
- Supercycle
- Delay compensation

Timing introduction (I...)

Principles and requirements

- Synchronize the operation of the facility
  - Accelerator, target, neutron instruments, CF (small extent)
- Provide the base frequency of 14 Hz
  - Also lower (beam) repetition rates
- Provide timestamping to the facility
  - Data correlation, EPICS archiving
- Provide trigger and clock signals
  - With a defined position in an operation sequence
- Support post-mortem analysis
  - What was the cause of a beam trip?

Timing introduction (...II...)

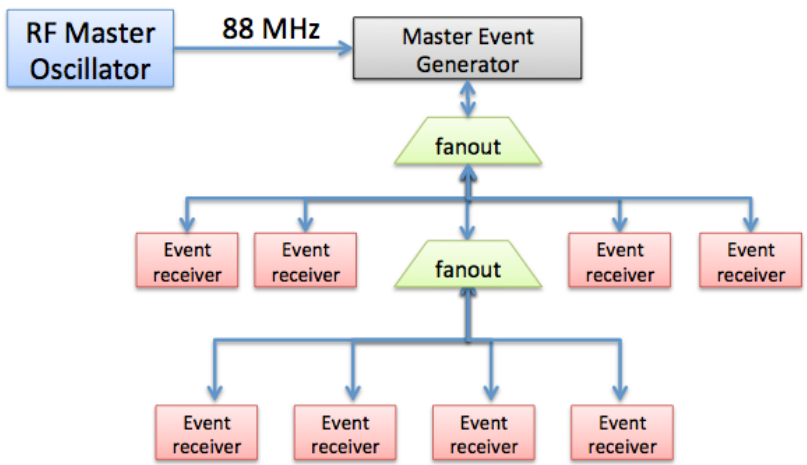
Implementation principles

- A single master unit controls the timing distribution
  - Control from one central point
- Client units (receivers) act on masters orders
  - No dependency on e.g. computer network traffic
  - Actions pre-programmed at the receivers
- Synchronization to RF
  - All timing clocks phase-synchronous to RF (subharmonic)
- Distributes "wall-clock" time from the master
- Timestamp support in hardware
  - Used for data reconstruction and correlation

Timing introduction (...III...)

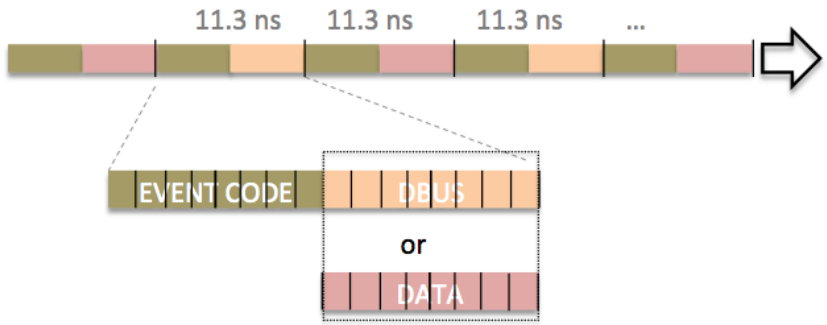
Structure of the system

- Master Event Generator (EVG) generates a continuous data stream
- The (accelerator) synchronization frequency (of 88 MHz) will be fed into the master event generator and used as the system clock. Event receivers synchronize to that clock.
- The 14 Hz operating frequency will be generated by down conversion from RF in the event generator (divide 88.0525 MHz by 6289464 to get 14.00000064 Hz)
- All timing sequences, events, etc., generated in hardware
- Links work upstream, too



Timing introduction (...IV...)

On-the-wire protocol



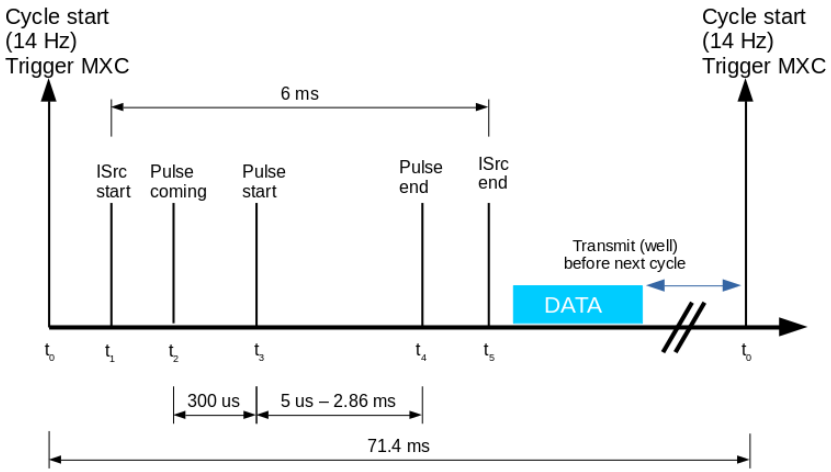
- 16-bit "frame" repeating at 88.0525 MHz (11.3 ns), i.e. 352.2 MHz/4
- Each frame contains an 8-bit event code and alternating distributed bus/data byte (8 bits).
- Distributed bus (DBus) broadcasts eight binary signals which can be output from each Event Receiver. Clocks, status bits, etc.
- Event Generator can broadcast arbitrary data (up to 2kB blocks)
- Data buffer sent one byte at a time and collected at the receiver(s)
- Trigger events and DBUS/Data are independent of each other

Timing introduction (...V...)

Events, clocks and data

- Meaning of "event" in this context: signal at a precise time
  - Hardware outputs with programmable delay, width, polarity
  - Event can also trigger software (EPICS) to do actions
    - "Collect post-mortem data now", "start counting", etc.
  - Events can be (synchronously) time-stamped
  - "Upstream" events: from EVR to EVG (and back)
- Clock: a repetitive sequence of pulses
  - E.g., a 1 kHz synchronization clock
- Data transmission
  - Limited amount of data delivered synchronously
  - For configuration and validation purposes
  - Still under development (some space for requests)

Accelerator cycle



- Pre-programmed sequences repeat at 14 Hz
  - Hardware-based (FPGA) sequencer with a RF-synchronous 88 MHz clock
  - "Programming" can happen in less than a millisecond
- "Data", i.e., beam mode, etc., transmitted in parallel
  - Contains "plan" for the next cycle: envelope mode, beam/no beam, pulse counter, rep rate, ...
  - Mode changes require a handshake with BIS
    - When mode change is transmitted, timing does not send beam before MPS/BIS gives OK

EUROPEAN SPALLATION SOURCE

Timing adjustment policy

- Local delay settings are used for local fine-tuning; adjusting for propagation, processing, etc., delays during commissioning
- Pulse length, etc. information in beam data buffer is not for setting the pulse timing locally but for:
  - Pulse verification, feed-back/forward, etc.
- Beam off will be a dedicated event. A gate signal will be set on with a beam on event and set off with a beam off event
  - Central handling of the beam duration, no software activity needed at the (hundreds of) local receivers (to set the delays)
  - Actions can be tied to the beam off event, like beam-sync data collection and broadcast
  - Pulse may be cut short by the timing system. Separate event will allow clean handling of the early pulse abort

Timing introduction (... and VIII)

EUROPEAN SPALLATION SOURCE

Documentation

- ESS Timing System System Architecture Description - ESS-0088633
- Data Model Specification for the ESS Timing System - ESS-0225673
- Event System with Delay Compensation Technical Reference - `http://mrf.fi/`
- EVR Usage Guide - `http://epics.sourceforge.net/mrfioc2/evr-usage.pdf` or `https://github.com/epics-modules/mrfioc2/blob/master/documentation/evr-usage.lyx` (lyx version)
- Manuals in CHESS

System configuration (I...)

```
[iocuser@icslab-ts03 ~]$ lspci
0a:00.0 Signal processing controller: Xilinx Corporation Device 7011
[iocuser@icslab-ts03 ~]$ lspci -s a:0 -vv
0a:00.0 Signal processing controller: Xilinx Corporation Device 7011
Subsystem: Device 1a3e:132c
Physical Slot: 1-1
Control: I/O+ Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr- Stepping-
     SERR- FastB2B- DisINTx-
Status: Cap+ 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast >TAbort- <TAbort- <MAbort-
      >SERR- <PERR- INTx-
Latency: 0, Cache Line Size: 64 bytes
Interrupt: pin A routed to IRQ 7
Region 0: Memory at c0600000 (32-bit, non-prefetchable) [size=256K]
Capabilities: <access denied>
```

System configuration (...II...)

```
[iocuser@icslab-ts03 ics_gitsrc]$ git clone https://github.com/jeonghanlee/
    pciids

[iocuser@icslab-ts03 pciids]$ sh replace-pciids.bash

[iocuser@icslab-ts03 pciids]$ lspci -s a:0 -vv
0a:00.0 Signal processing controller: Xilinx Corporation XILINX PCI DEVICE
Subsystem: Micro-Research Finland Oy MTCA Event Receiver 300
Physical Slot: 1-1
Control: I/O+ Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr- Stepping-
    SERR- FastB2B- DisINTx-
Status: Cap+ 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast >TAbort- <TAbort- <MAbort-
    >SERR- <PERR- INTx-
Latency: 0, Cache Line Size: 64 bytes
Interrupt: pin A routed to IRQ 7
Region 0: Memory at c0600000 (32-bit, non-prefetchable) [size=256K]
Capabilities: <access denied>
```

System configuration (...III...)

```
[iocuser@icslab-ts03 e3-mrfioc2]$ make init
[iocuser@icslab-ts03 e3-mrfioc2]$ make dkms_add
[iocuser@icslab-ts03 e3-mrfioc2]$ make dkms_build
[iocuser@icslab-ts03 e3-mrfioc2]$ make dkms_install
[iocuser@icslab-ts03 e3-mrfioc2]$ make setup
```

System configuration (... and IV)

```
[iocuser@icslab-ts03 e3-mrfioc2]$ systemctl status dkms
 dkms.service - Builds and install new kernel modules through DKMS
   Loaded: loaded (/usr/lib/systemd/system/dkms.service; enabled; vendor preset:
           enabled)
   Active: active (exited) since Fri 2018-08-17 10:28:19 CEST; 57s ago
     Docs: man:dkms(8)
  Process: 793 ExecStart=/bin/sh -c dkms autoinstall --verbose --kernelver $(
          uname -r) (code=exited, status=0/SUCCESS)
 Main PID: 793 (code=exited, status=0/SUCCESS)
    Tasks: 0
   CGroup: /system.slice/dkms.service

Aug 17 10:28:18 icslab-ts03 systemd[1]: Starting Builds and install new kernel
       modules through DKMS...
Aug 17 10:28:19 icslab-ts03 systemd[1]: Started Builds and install new kernel
       modules through DKMS.
```

EVR: basic start-up

```
### Load the mrfioc2 module ###
require mrfioc2,2.2.0-rc2

### Define several needed macros ###
epicsEnvSet("IOC", "TRAINING")
epicsEnvSet("DEV1", "EVR1")
epicsEnvSet("MainEvtCODE" "14")
epicsEnvSet("HeartBeatEvtCODE"    "122")
epicsEnvSet("ESSEvtClockRate"   "88.0525")

### Register the EVR with the IOC and load the database ###
mrmEvrSetupPCI("$(DEV1)",   "0a:00.0")
dbLoadRecords("evr-mtca-300u-ess.db","EVR=$(DEV1), SYS=$(IOC), D=$(DEV1), FEVT=$
      (ESSEvtClockRate)")

### Needed with software timestamp source w/o RT thread scheduling ###
var evrMrmTimeNSOverflowThreshold 100000


iocInit()

### Set delay compensation to 70 ns, needed to avoid timesptamp issue ###
dbpf $(IOC)-$(DEV1):DC-Tgt-SP 70
```

EVR: basic configuration

```
[iocuser@icslab-ts03 cmds]$ iocsh.bash TRAINING-EVR1.cmd

caget TRAINING-EVR1:Link-Sts
caget TRAINING-EVR1:Link-Clk-I
caget TRAINING-EVR1:Cnt-LinkTimo-I
caput TRAINING-EVR1:Link-Clk-SP 100
caget TRAINING-EVR1:Link-Clk-I
caget TRAINING-EVR1:Link-Sts
caget TRAINING-EVR1:Cnt-LinkTimo-I
caget TRAINING-EVR1:Cnt-LinkTimo-I
caput TRAINING-EVR1:Link-Clk-SP 88.0525
caget TRAINING-EVR1:Link-Clk-I
caget TRAINING-EVR1:Link-Sts

caget TRAINING-EVR1:Time-Valid-Sts

caget TRAINING-EVR1:FwVer-I
```

EVR: events and triggers

```
camonitor TRAINING-EVR1:EvtECnt-I

caput TRAINING-EVR1:DlyGen0-Evt-Trig0-SP 14

caput TRAINING-EVR1:DlyGen0-Width-SP 1000

caput TRAINING-EVR1:OutFP0-Src-SP 0

/epics/base-3.15.5/require/3.0.0/siteMods/mrfioc2/2.2.0-rc2/db/evr-mtca-300u-ess
      .db

TRAINING-EVR1:OutFP0-Src-Pulse-SP
TRAINING-EVR1:OutFP0-Src-DBus-SP
TRAINING-EVR1:OutFP0-Src-Scale-SP
```

EVR: timestamping

https://github.com/icshwi/esstex/tree/master/document/ics_eng_docs_timestamping

```
6690f8a.icslab-ipc01.14335 > generalTimeReport
Backwards time errors prevented 1464 times.

Current Time Providers:
    "EVR", priority = 50
        Current Time is 2018-08-13 11:18:40.234652.
    "OS Clock", priority = 999
        Current Time is 2018-08-13 11:20:05.683007.

Event Time Providers:
    "EVR", priority = 50


dbLoadRecords("./counter.db")

camonitor counter

caput TRAINING-EVR1:Time-I.EVNT 14
caput TRAINING-EVR1:Time-I.INP "@OBJ=EVR1, Code=14"
caput counter.TSEL TRAINING-EVR1:Time-I.TIME
```

EVR: standalone mode (I...)

```
### Load the mrfioc2 module ###
require mrfioc2,2.2.0-rc2

### Define several needed macros ###
epicsEnvSet("IOC", "TRAINING")
epicsEnvSet("DEV1", "EVR1")
epicsEnvSet("MainEvtCODE" "14")
epicsEnvSet("HeartBeatEvtCODE"    "122")
epicsEnvSet("ESSEvtClockRate"   "88.0525")

### Register the EVR with the IOC and load the database ###
mrmEvrSetupPCI("$(DEV1)",   "0a:00.0")
dbLoadRecords("evr-mtca-300u-ess.db","EVR=$(DEV1), SYS=$(IOC), D=$(DEV1), FEVT=$
      (ESSEvtClockRate)")

### Needed with software timestamp source w/o RT thread scheduling ###
var evrMrmTimeNSOverflowThreshold 100000
```

EVR: standalone mode (...II...)

```
iocInit()

### Set delay compensation to 70 ns, needed to avoid timesptamp issue ###
dbpf $(IOC)-$(DEV1):DC-Tgt-SP 70

### Get current time from system clock ###
dbpf $(IOC)-$(DEV1):TimeSrc-Sel "Sys. Clock"

### Set up the prescaler that will trigger the sequencer at 14 Hz ###
dbpf $(IOC)-$(DEV1):PS0-Div-SP 6289424 # in standalone mode because real freq
     from synthsiezer is 88.05194802 MHz, otherwise 6289464

### Set up the sequencer ###
dbpf $(IOC)-$(DEV1):SoftSeq0-RunMode-Sel "Normal"
dbpf $(IOC)-$(DEV1):SoftSeq0-TrigSrc-2-Sel "Prescaler 0"
dbpf $(IOC)-$(DEV1):SoftSeq0-TsResolution-Sel "uSec"
dbpf $(IOC)-$(DEV1):SoftSeq0-Load-Cmd 1
dbpf $(IOC)-$(DEV1):SoftSeq0-Enable-Cmd 1

### Run the script that configures the events and timestamp of the sequence ###
system("/bin/sh ./configure_sequencer_14Hz.sh $(IOC) $(DEV1)")
```

EVR: standalone mode (... and III)

EUROPEAN SPALLATION SOURCE

configure_sequencer_14Hz.sh:

```
# Bash script to configure the EVG/EVR sequencer
# All values in us

# Event code 14 (14 Hz), 127 is the end of sequence
caput -a $1-$2:SoftSeq0-EvtCode-SP 2 14 127

# Defining time at which the event codes are sent in us
caput -a $1-$2:SoftSeq0-Timestamp-SP 2 0 1

# Commit the sequence to HW
caput $1-$2:SoftSeq0-Commit-Cmd 1
```

EVR: inputs

```
# Set up FPIN0. Generate Code 10 locally on rising edge.
#dbpf $(IOC)-$(DEV1):OutRB00-Src-SP 61 # PCIe-EVR-300DC
dbpf $(IOC)-$(DEV1):In0-Trig-Ext-Sel "Edge"
dbpf $(IOC)-$(DEV1):In0-Code-Ext-SP 10

TRAINING-EVR1:In0-Lvl-Sel
TRAINING-EVR1:In0-Edge-Sel
TRAINING-EVR1:In0-Trig-Ext-Sel
TRAINING-EVR1:In0-Code-Ext-SP
```

EVR: backplane clocks

https://github.com/icshwi/esstex/tree/master/document/ics_eng_docs_mtca_backplane_clocks

```
# Set TCLKB to low, enable it and power it up
dbpf $(IOC)-$(DEV1):OutTCLKB-Src-SP 63
dbpf $(IOC)-$(DEV1):OutTCLKB-Ena-Sel 1
dbpf $(IOC)-$(DEV1):OutTCLKB-Pwr-Sel 1
# TCLKB is 40-bit pattern, set the starting 20 bits to 1 (and the rest to 0 -
    default)
dbpf $(IOC)-$(DEV1):OutTCLKB-Pat-Low00_15-SP.BF 1
dbpf $(IOC)-$(DEV1):OutTCLKB-Pat-Low00_15-SP.BE 1
dbpf $(IOC)-$(DEV1):OutTCLKB-Pat-Low00_15-SP.BD 1
dbpf $(IOC)-$(DEV1):OutTCLKB-Pat-Low00_15-SP.BC 1
dbpf $(IOC)-$(DEV1):OutTCLKB-Pat-Low00_15-SP.BB 1
dbpf $(IOC)-$(DEV1):OutTCLKB-Pat-Low00_15-SP.BA 1
dbpf $(IOC)-$(DEV1):OutTCLKB-Pat-Low00_15-SP.B9 1
dbpf $(IOC)-$(DEV1):OutTCLKB-Pat-Low00_15-SP.B8 1
dbpf $(IOC)-$(DEV1):OutTCLKB-Pat-Low00_15-SP.B7 1
dbpf $(IOC)-$(DEV1):OutTCLKB-Pat-Low00_15-SP.B6 1
dbpf $(IOC)-$(DEV1):OutTCLKB-Pat-Low00_15-SP.B5 1
dbpf $(IOC)-$(DEV1):OutTCLKB-Pat-Low00_15-SP.B4 1
dbpf $(IOC)-$(DEV1):OutTCLKB-Pat-Low00_15-SP.B3 1
dbpf $(IOC)-$(DEV1):OutTCLKB-Pat-Low00_15-SP.B2 1
dbpf $(IOC)-$(DEV1):OutTCLKB-Pat-Low00_15-SP.B1 1
dbpf $(IOC)-$(DEV1):OutTCLKB-Pat-Low00_15-SP.B0 1
dbpf $(IOC)-$(DEV1):OutTCLKB-Pat-Low16_31-SP.BF 1
dbpf $(IOC)-$(DEV1):OutTCLKB-Pat-Low16_31-SP.BE 1
dbpf $(IOC)-$(DEV1):OutTCLKB-Pat-Low16_31-SP.BD 1
dbpf $(IOC)-$(DEV1):OutTCLKB-Pat-Low16_31-SP.BC 1
```

EVG: basic start-up

```
require mrfioc2, 2.2.0-rc2

epicsEnvSet("IOC", "TRAINING")
epicsEnvSet("DEV1", "EVG1")

epicsEnvSet("ESSEvtClockRate"   "88.0525")

mrmEvgSetupPCI($(DEV1), "16:0e.0")
dbLoadRecords("cpci-evg230-ess.db",   "SYS=$(IOC), D=$(DEV1), EVG=$(DEV1), FEVT=$
      (ESSEvtClockRate), FRF=$(ESSEvtClockRate), FDIV=1")

iocInit()
```

EVG: basic configuration

```
# Event frequency:
caput TRAINING-EVG1:EvtClk-FracSynFreq-SP 100

# Software event:
caput TRAINING-EVG1:SoftEvt-EvtCode-SP 10

# Timestamping:
dbpf $(IOC)-$(DEV1):1ppsInp-Sel "Sys Clk"
...
epicsThreadSleep 5
dbpf $(IOC)-$(DEV1):SyncTimestamp-Cmd 1

# RF input:
dbpf $(IOC)-$(DEV1):EvtClk-Source-Sel "RF"
dbpf $(IOC)-$(DEV1):EvtClk-RFFreq-SP 88.0525
dbpf $(IOC)-$(DEV1):EvtClk-RFDiv-SP 1

# Configure front panel for event 125 1 Hz
dbpf $(IOC)-$(DEV1):TrigEvt1-EvtCode-SP 125
dbpf $(IOC)-$(DEV1):TrigEvt1-TrigSrc-Sel "Univ0"
dbpf $(IOC)-$(DEV1):1ppsInp-Sel "Univ0"
dbpf $(IOC)-$(DEV1):1ppsInp-MbbiDir_.TPRO 1
```

EVG: creating events and data

```
# Heart Beat 1 Hz:
dbpf $(IOC)-$(DEV1):Mxc7-Prescaler-SP 88052500
dbpf $(IOC)-$(DEV1):TrigEvt7-EvtCode-SP $(HeartBeatEvtCODE)
dbpf $(IOC)-$(DEV1):TrigEvt7-TrigSrc-Sel "Mxc7"

# Setup of sequencer:
dbpf $(IOC)-$(DEV1):Mxc0-Prescaler-SP 6289464
dbpf $(IOC)-$(DEV1):SoftSeq0-RunMode-Sel 0 # normal mode
dbpf $(IOC)-$(DEV1):SoftSeq0-TrigSrc-Sel "Mxc0"
dbpf $(IOC)-$(DEV1):SoftSeq0-TsResolution-Sel 2 # us
dbpf $(IOC)-$(DEV1):SoftSeq0-Load-Cmd 1
dbpf $(IOC)-$(DEV1):SoftSeq0-Enable-Cmd 1
system("/bin/sh ./configure_sequence.sh $(IOC) $(DEV1)")
```

EUROPEAN SPALLATION SOURCE

Data buffer

TRAINING-EVR1:dbus-recv-u32
TRAINING-EVM1:dbus-send-u32

TRAINING-EVR1:Link-RxMode-Sel
TRAINING-EVM1:Link-TxMode-Sel

EUROPEAN SPALLATION SOURCE

Supercycle

- Supercycle: a predefined sequence of machine cycles:
  - Timing pattern
  - Beam parameter (data)
- Cycles are pre-configured (with a control-room application)
- E.g., a supercycle for:
  - 1 Hz operation must have 1 cycle with beam, 13 cycles without beam
  - 0.1 Hz: 1 cycle with beam, 139 cycles without beam
- Editor application (to be developed) allows to define
  - pulse length (timing events in HW sequencer)
  - beam current (iris change, probably not between cycles)
  - pulse delay (within 2.86 time window, in sequencer)
  - beam destination (in data buffer)
  - beam mode (in data buffer)
  - etc.
- Concept still needs refinement

Delay compensation

DC

Delay compensation is achieved in measuring the propagation delay of events from the delay compensation master EVM through the distribution network up to the Event Receivers. At the last stage the EVR is aware of the delay through the network and adjusts an internal FIFO depth to match a programmed target delay value.

EUROPEAN SPALLATION SOURCE

Questions?