utshutgt

Timing system training for ICS integrators

Javier Cereijo Garcia

Integrated Control System Division
ESS, Sweden

https://www.europeanspallationsource.se
August 17, 2018

Outline

- Timing introduction
- (E3 introduction)
- System configuration
- e3-mrfioc2
- EVR: basic start-up
- EVR: basic configuration
- EVR: events and triggers
- EVR: backplane clocks
- EVR: timestamping
- EVR: standalone mode
- EVR: inputs
- EVG: basic start-up
- EVG: basic configuration
- EVG: creating events and data
- Features in development

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...)

EUROPEAN SPALLATION SOURCE

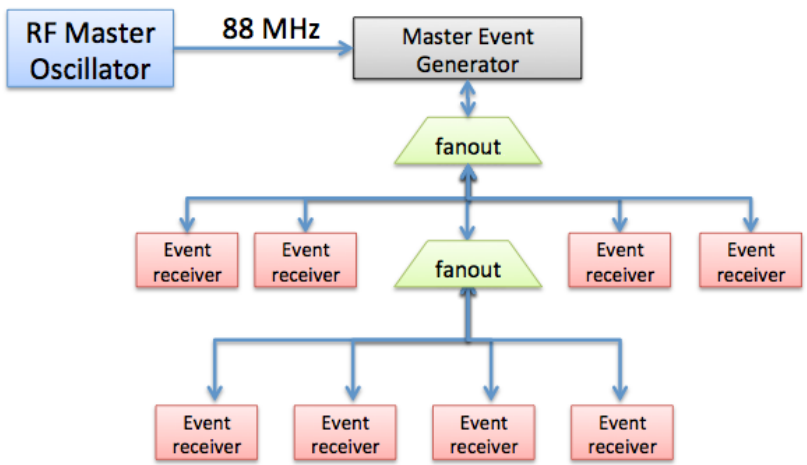
Implementation principles

- 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...)

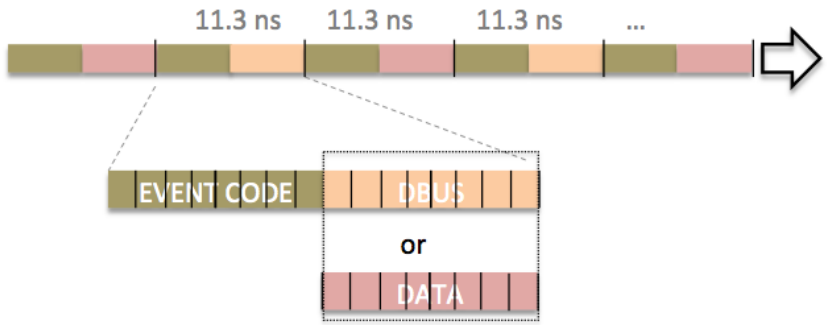
Structure of the system

- 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



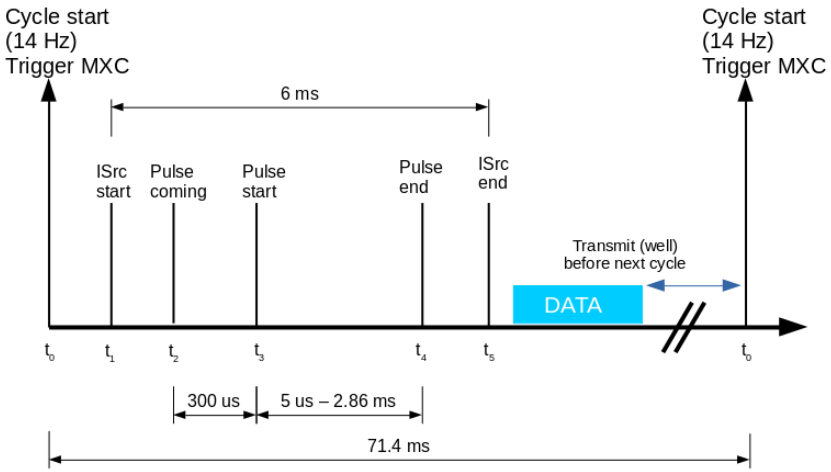
- 16-bit "frame" repeating at 88.0525 MHz (11.3 ns), i.e. 352.2 MHz/4
- Each frame contains an 8-bit event code and alternating distributed bus/data byte (8 bits).
- Distributed bus (DBus) broadcasts eight binary signals which can be output from each Event Receiver. Clocks, status bits, etc.
- Event 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)

Documentation

- ESS Timing System System Architecture Description - ESS-0088633 - `https://chess.esss.lu.se/enovia/link/ESS-0088633/21308.51166.24576.44510/valid`
- Data Model Specification for the ESS Timing System - ESS-0225673 - not available yet
- Event System with Delay Compensation Technical Reference - `http://mrf.fi/fw/DCManual-170209.pdf` (not latest version, ask Javier if interested)
- 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 `https://github.com/icshwi/esstex/tree/master/document` (not all of them up to date)

(E3 introduction (I...))

```
[iocuser@icslab-ts03 ics_gitsrc]$ git clone https://github.com/icshwi/e3

[iocuser@icslab-ts03 e3-mrfioc2]$ ./e3.bash -t all

[iocuser@icslab-ts03 ~]$ source ~/ics_gitsrc/e3/tools/setenv

Set the ESS EPICS Environment as follows:
THIS Source NAME     : setE3Env.bash
THIS Source PATH     : /epics/base-3.15.5/require/3.0.0/bin
EPICS_BASE           : /epics/base-3.15.5
EPICS_HOST_ARCH      : linux-x86_64
E3_REQUIRE_LOCATION  : /epics/base-3.15.5/require/3.0.0
PATH                 : /epics/base-3.15.5/require/3.0.0/bin:/epics/base-3.15.5/
      bin/linux-x86_64:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/
      iocuser/.local/bin:/home/iocuser/bin
LD_LIBRARY_PATH      : /epics/base-3.15.5/lib/linux-x86_64:/epics/base-3.15.5/
      require/3.0.0/lib/linux-x86_64:/epics/base-3.15.5/require/3.0.0/siteLibs/
      linux-x86_64

Enjoy E3!
```

(E3 introduction (... and II))

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

[iocuser@icslab-ts03 ics_gitsrc]$ bash essics_scripts/iocuser_env/
    iocuser_env_setup.bash clean

[iocuser@icslab-ts03 ics_gitsrc]$ bash  essics_scripts/iocuser_env/
    iocuser_env_setup.bash install

[iocuser@icslab-ts03 ics_gitsrc]$ bash  essics_scripts/iocuser_env/
    iocuser_env_setup.bash link
```

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.
```

e3-mrfioc2

- `https://github.com/icshwi/e3-mrfioc2/`
- Cmds: startup scripts
- Opi: community opis with ESS naming (check readme for opis up to date)
- Template: substitutions files
- Installation path: /epics/base-3.15.5/require/3.0.0/siteMods/mrfioc2/2.2.0-rc2/db/

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 startup_EVR.cmd
TRAINING-EVR1:Link-Sts
TRAINING-EVR1:Link-Clk-SP
TRAINING-EVR1:Link-Clk-I
TRAINING-EVR1:Cnt-LinkTimo-I
TRAINING-EVR1:FwVer-I
```

EVR: events and triggers

EUROPEAN SPALLATION SOURCE

```
TRAINING-EVR1:EvtECnt-I

TRAINING-EVR1:DlyGen0-Evt-Trig0-SP

TRAINING-EVR1:DlyGen0-Width-SP

TRAINING-EVR1:OutFPUV00-Src-SP

TRAINING-EVR1:OutFPUV00-Src-Pulse-SP
TRAINING-EVR1:OutFPUV00-Src-DBus-SP
TRAINING-EVR1:OutFPUV00-Src-Scale-SP
```

EVR: backplane clocks

EVR: timestamping

EVR: standalone mode

EVR: inputs

EVG: basic start-up

EVG: basic configuration

EVG: creating events and data

Features in development

Features in development (I...)

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

EUROPEAN SPALLATION SOURCE

Questions?