ICS Engineering Manual

FOR MRF MTCA-EVR-300

	Name (Role/Title)
Author	Javier Cereijo Garcia Jeong Han Lee
Reviewer	TBD
Owner	ICS
Approver	ICS

Contents

Description	Engineering Manual
Document number	ESS-XXXXXXXX
Date	September 29, 2016
Revision	0.1
State	Early Draft
Classification	ESS Use Only

1 Overview

At European Spallation Source (ESS), Integrated Control System (ICS) does use the Micro Research Finland (MRF) Timing System[1] as its timing system of the ESS site. The consistent and up-to-date engineering manual is essential for the ESS Timing system.

1.1 Scope

- This document identifies one of the MRF Timing Event Receivers (EVR) that needs to be configured for an ESS subsystem that needs synchronous frequencies, trigger signals and sequences of events [1].
- This document provides the generic description of the MRF MTCA-EVR-300 and its interface board (IFB-300). In addition, it affords the minimal, essential, and generic information for the system configuration.
- The purpose of this document is to describe the engineering procedure and troubleshooting about how the MRF MTCA-EVR-300 board will be integrated in cooperation with the ESS EPICS Environment (EEE).
- This document attempts to maintain consistency with existing ESS Timing system hardware as far as possible.

Note that this is a very early draft document and should be updated as development progresses.

1.2 Target Audience

This document is targeted to ICS engineers and technical stakeholders of the ESS timing system. It is assumed that the target audience has a technical background in the MRF Timing System, the EPICS development, and a Linux environment.

2 System Description

MRF Technical Reference [see 1, p45] explained Event Receivers and wrote :

> *Event Receivers decode timing events and signals from an optical event stream transmitted by an Event Generator. Events and signals are received at predefined rate the event clock that is usually divided down from an accelerators main RF reference. The event receivers lock to the phase event clock of the Event Generator and are thus phase locked to the RF reference. Event Receivers convert event codes transmitted by an Event Generator to hardware*

[1] http://www.mrf.fi/

outputs. They can also generate software interrupts and store the event codes with globally distributed timestamps into FIFO memory to be read by a CPU.

ICS uses and will use the following different types of EVR :

- VME-EVR-230 / 230RF
- PMC-EVR-230
- MTCA-EVR-300 / 300DC
- PCIe-EVR-300 / 300DC
- VME-EVR-300 / 300DC

The scope of this document is to cover MTCA-EVR-300 board.

2.1 MTCA-EVR-300

Figure 1 shows the rough physical dimensions 181×148 mm^2 of the MTCA-EVR-300 card.

The MTCA-EVR-300 has a SFP transceiver as an input from EVG and several outputs: 4 front panel outputs, 16 front universal outputs (through the IFB-300 extension board) and 40 rear outputs. The initial 32 rear outputs map to the RTM connector, the last 8 rear outputs map to the MTCA backplane. The 16 front universal outputs are implemented through a micro-SCSI type connector for an interface board IFB-300. The IFB-300 has eight Universal I/O slots, shown in Figure 2. With different type of MRF Universal I/O modules, each slot can be used as an unique trigger or event signal source.

Description	Engineering Manual
Document number	ESS-XXXXXXXX
Date	September 29, 2016
Revision	0.1
State	Early Draft
Classification	ESS Use Only

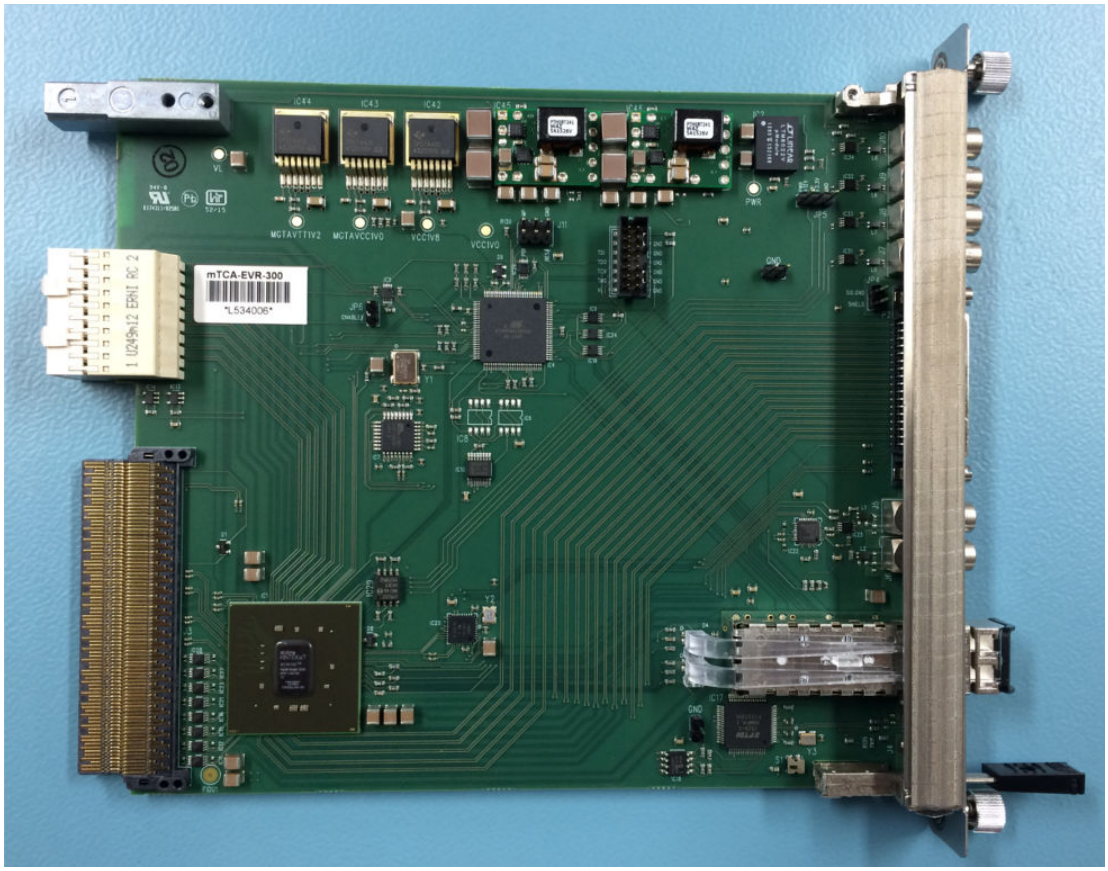

Figure 1 MRF MTCA-EVR-300 board.

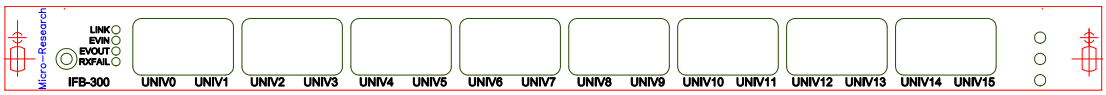

Figure 2 MRF Interface Board IFB 300 Front Panel [1].

3 System Environment

Before describing the engineering procedure for an EEE integration of the MRF MTCA-EVR-300 board, it is mandatory to have proper system environment that consists of specific hardware and software lists. Here we will show the hardware and software lists, their block diagrams, and their setup in the ICS lab at ESS. The information shown in this chapter is used in the ICS Lab at ESS.

3.1 Hardware

Table 1 shows the hardware list and its environment. The form factor and version of EVG can be changeable. It is assumed that the proper working EVG system is ready. Here, `TAG` is used as the prefix of the ICS internal inventory system in order to track it down. One can use only EVG without FOUT.

Hardware	Info	Serial Number
MRF MTCA-EVR-300	`ICS TAG-255`	L534006
NAT-MCH-PHYS	`ICS TAG-188`	1135150327
Concurrent Technologies AMC CPU	`ICS TAG-190`, hostname: icsb-mtcacpu-ct028	M2349/028
Struck SIS8300		
ELMA MTCA crate 12 slots, 9U	`ICS TAG-181`	10615060
Wiener power supply unit 1000W	`ICS TAG-187`	0985052
MRF IFB-300	`ICS TAG-352`	K472044
Universal I/O		
MRF cPCI-EVG-230	`ICS TAG-26`	E283016
MRF cPCI-FOUT-12	`ICS TAG-27`	H385013
Optical cables	LC, Optical 850 nm	
Ethernet cables		
LEMO cables		
LEMO to BNC Adapters		
Oscilloscope		

Table 1 Hardware List and Its Environment.

Figure 3 shows the MTCA-EVR-300 setup in the lab. From left to right, the power supply, MCH, CPU, MTCA-EVR-300 and Struck SIS8300.

Figure 3 Hardware Setup in the ICS lab.

3.2 Software

Table 2 shows the Software list and its environment. It is mandatory to check the kernel version, and the mrf kernel module version. Since the mrfioc2 is dependent upon devLibs2 EEE internally, an end-user is unnecessary to check its version explicitly.

Item	Version Info.
CentOS Linux	`7.1.1503`
Kernel	`3.10.0-229.7.2.el7.x86_64`
mrf kernel module	version : `1` / srcversion `124F2C1F3D5E2080AE1B755`
EEE	`1.8.0`
EPICS Base	`3.14.12.5`
mrfioc2	EEE module ver. `2.7.13`
devLib2	EEE module ver. `2.6+`

Table 2 Software and its version information.

3.3 EVR Firmware

Table 3 shows EVR FPGA Firmware Version Register.

EVR FPGA Firmware Version Register	`0x18000205`	
Board Type	EVR	`0x1`8000205
Form Factor	mTCA.4	`0x18`000205
EVR Firmware ID	Delay Compensation Firmware	`0x1800`**`02`**`05`
EVR Revision ID	5	`0x180002`**`05`**

Table 3 EVR FPGA Firmware Version Register in Reference [see 1, p66].

4 Engineering Procedure

This chapter provides the minimal information to configure the EVR board properly.

4.1 System Installation

Figure 3 shows the glimpse of what system might be like in a Lab. **Note that the cable between the mTCA-EVR-300DC and IFB-300 (not shown in the figure) should be connected, disconnected, or both only when powered down**. Please see the detail information in Reference [1, p54].

4.2 mTCA-EVR-300 Board Identification

4.2.1 Kernel Module

It is essential to load the mrf kernel module and to check its information as follows:

```
[root@icsb-mtcacpu-ct028 mTCA-EVR-300_tests]# modprobe mrf

[root@icsb-mtcacpu-ct028 mTCA-EVR-300_tests]# modinfo mrf
filename:       /lib/modules/3.10.0-229.7.2.el7.x86_64/extra/mrf.ko
author:         Michael Davidsaver <mdavidsaver@bnl.gov>
version:        1
license:        GPL v2
rhelversion:    7.1
srcversion:     124F2C1F3D5E2080AE1B755
depends:        parport,uio
vermagic:       3.10.0-229.7.2.el7.x86_64 SMP mod_unload modversions
parm:           cable:Name of JTAG parallel port cable to emulate (charp)
parm:           interfaceversion:User space interface version (int)
```

It is mandatory to check the access permission of an IOC user for the device file, e.g., `/dev/uio0` to allow the IOC process to open it. In case ones target system is NOT running `UDEV`, please consult the EVG user guide [2]. And for building and loading the EVR kernel module and for changing the device file permission, please see Reference [see 2, p12,13]. It is inadvisable to change the file permission by using `chmod`.

4.2.2 PCI Addressing

Each PCI device is identified by a domain, a bus, a device, and a function number in Linux. Therefore, in order to initialize the MRF MTCA-EVR-300 board in EEE, one needs the following information: a domain number, a bus number, a device number, and a function number. These numbers are the parameters of a `mrmEvrSetupPCI` function.

One can use `lspci` to find them as follows:

```
[root@icsb-mtcacpu-ct028 mTCA-EVR-300_tests]# lspci
...
06:00.0 Signal processing controller: Xilinx Corporation Device 7011
...
[root@icsb-mtcacpu-ct028 mTCA-EVR-300_tests]# lspci -s 06:00 -vv
06:00.0 Signal processing controller: Xilinx Corporation Device 7011
```

```
Subsystem: Device 1a3e:132c
Physical Slot: 3
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 18
Region 0: Memory at c0700000 (32-bit, non-prefetchable) [size=256K]
Capabilities: [40] Power Management version 3
   Flags: PMEClk- DSI+ D1- D2- AuxCurrent=0mA PME(D0-,D1-,D2-,D3hot-,D3cold-)
   Status: D0 NoSoftRst+ PME-Enable- DSel=0 DScale=0 PME-
Capabilities: [48] MSI: Enable- Count=1/1 Maskable- 64bit+
   Address: 0000000000000000  Data: 0000
Capabilities: [60] Express (v2) Endpoint, MSI 00
   DevCap:  MaxPayload 256 bytes, PhantFunc 1, Latency L0s <64ns, L1 <1us
      ExtTag+ AttnBtn- AttnInd- PwrInd- RBE+ FLReset-
   DevCtl:  Report errors: Correctable- Non-Fatal- Fatal- Unsupported-
      RlxdOrd+ ExtTag- PhantFunc- AuxPwr- NoSnoop+
      MaxPayload 256 bytes, MaxReadReq 512 bytes
   DevSta:  CorrErr- UncorrErr- FatalErr- UnsuppReq- AuxPwr- TransPend-
   LnkCap:  Port #0, Speed 2.5GT/s, Width x1, ASPM L0s, Exit Latency L0s unlimited, L1
  unlimited
      ClockPM- Surprise- LLActRep- BwNot-
   LnkCtl:  ASPM Disabled; RCB 64 bytes Disabled- CommClk-
      ExtSynch- ClockPM- AutWidDis- BWInt- AutBWInt-
   LnkSta:  Speed 2.5GT/s, Width x1, TrErr- Train- SlotClk+ DLActive- BWMgmt- ABWMgmt-
   DevCap2: Completion Timeout: Not Supported, TimeoutDis-, LTR-, OBFF Not Supported
   DevCtl2: Completion Timeout: 50us to 50ms, TimeoutDis-, LTR-, OBFF Disabled
   LnkCtl2: Target Link Speed: 2.5GT/s, EnterCompliance- SpeedDis-
       Transmit Margin: Normal Operating Range, EnterModifiedCompliance- ComplianceSOS-
       Compliance De-emphasis: -6dB
   LnkSta2: Current De-emphasis Level: -3.5dB, EqualizationComplete-, EqualizationPhase1-
       EqualizationPhase2-, EqualizationPhase3-, LinkEqualizationRequest-
Capabilities: [100 v1] Device Serial Number 00-00-00-00-00-00-00-00
Kernel driver in use: mrf-pci
```

And one should identify four number as follows:

```
[root@icsb-mtcacpu-ct028 mTCA-EVR-300_tests]# lspci -s 06:00 -t
-+-[0000:06]---00.0
 \-[0000:00]-
```

, where `-+-[0000:06]---00.0` can be translated to -+-[domain:bus]---device.function. Thus in the above case, four numbers are shown in Table 4.

domain	0x0
bus	0x6
device	0x0
function	0x0

Table 4 MRF MTCA-EVR-300 Identification Numbers

4.3 EPICS IOC Setup under EEE

In this section, in order to start the EPICS IOC for the MRF MTCA-EVR-300 under EEE, one should consider the following things: 1) the EPICS database file, and 2) the EPICS start-up script. All files are located in a directory, where an user can create, e.g., in the ICS Lab,

```
/home/javiercereijogarcia/mTCA-EVR-300_tests
```

Listing 4.1 Working Directory in the ICS lab.

4.3.1 EPICS Database File

One could use the existent database file in the EEE as a template for the EPICS IOC, the file is located in the following location :

```
/opt/epics/modules/mrfioc2/2.7.13/db/evr-mtca-300.db
```

4.3.2 Start-Up Script

Listing 4.2 shows the IOC start-up script which has the MRF MTCA-EVR-300 Identification Numbers, shown in Table 4. Note that the start-up script is located in the directory in Listing 4.1.

```
1  epicsEnvSet("SYS"             "tests-mTCA-EVR-300")
2  epicsEnvSet("DEVICE"          "EVR0")
3  epicsEnvSet("EVR_PCIDOMAIN"   "0x0")
4  epicsEnvSet("EVR_PCIBUS"      "0x6")
5  epicsEnvSet("EVR_PCIDEVICE"   "0x0")
6  epicsEnvSet("EVR_PCIFUNCTION" "0x0")
7
8  require mrfioc2,2.7.13
9
10 mrmEvrSetupPCI($(DEVICE), $(EVR_PCIDOMAIN), $(EVR_PCIBUS), $(EVR_PCIDEVICE), $(EVR_PCIFUNCTION))
11
12 dbLoadRecords("evr-mtca-300.db", "DEVICE=$(DEVICE), SYS=$(SYS), Link-Clk-SP=88.0525")
```

Listing 4.2 Start-up script `mTCA-EVR-300_1.cmd`. Line 3-6 should be matched to Table 4.

4.3.3 EPICS IOC

Under EEE, the EPICS IOC can be started via the command `iocsh mTCA-EVR-300_1.cmd`. The output should look like as follows:

```
[root@icsb-mtcacpu-ct028 mTCA-EVR-300_tests]# iocsh mTCA-EVR-300_1.cmd
/opt/epics/bases/base-3.14.12.5/bin/centos7-x86_64/softIoc -D /opt/epics/bases/base-3.14.12.5/dbd/
    softIoc.dbd /tmp/iocsh.startup.23209
#date="Fri  9 Sep 13:52:57 UTC 2016"
#user="javiercereijogarcia"
#PWD="/home/javiercereijogarcia/mTCA-EVR-300_tests"
#EPICSVERSION="3.14.12.5"
#EPICS_HOST_ARCH="centos7-x86_64"
```

```
#SHELLBOX=""
#EPICS_CA_ADDR_LIST=""
#EPICS_MODULE_INCLUDE_PATH=".:/usr/lib64:/usr/lib:/lib64:/lib"
dlload         /opt/epics/modules/environment/1.8.0/3.14.12.5/lib/centos7-x86_64/libenvironment.so
dbLoadDatabase /opt/epics/modules/environment/1.8.0/3.14.12.5/dbd/environment.dbd
environment_registerRecordDeviceDriver
< "mTCA-EVR-300_1.cmd"
epicsEnvSet("SYS"             "tests-mTCA-EVR-300")
epicsEnvSet("DEVICE"          "EVR0")
epicsEnvSet("EVR_PCIDOMAIN"   "0x0")
epicsEnvSet("EVR_PCIBUS"      "0x6")
epicsEnvSet("EVR_PCIDEVICE"   "0x0")
epicsEnvSet("EVR_PCIFUNCTION" "0x0")
require mrfioc2,2.7.13
require: mrfioc2 depends on devlib2 (2.6+).
require: Loading library /opt/epics/modules/devlib2/2.6.0/3.14.12.5/lib/centos7-x86_64/libdevlib2.
     so.
require: Loading /opt/epics/modules/devlib2/2.6.0/3.14.12.5/dbd/devlib2.dbd.
require: Calling devlib2_registerRecordDeviceDriver function.
require: Loading library /opt/epics/modules/mrfioc2/2.7.13/3.14.12.5/lib/centos7-x86_64/libmrfioc2
     .so.
require: Adding /opt/epics/modules/mrfioc2/2.7.13/db.
require: Adding /opt/epics/modules/mrfioc2/2.7.13/startup.
require: Loading /opt/epics/modules/mrfioc2/2.7.13/3.14.12.5/dbd/mrfioc2.dbd.
require: Calling mrfioc2_registerRecordDeviceDriver function.

mrmEvrSetupPCI(EVR0, 0x0, 0x6, 0x0, 0x0)
Device EVR0 6:0.0
Using IRQ 18
Setting magic LE number!
FPGA version 0x18000205
Firmware version: 00000205
Found EVR0:SFP0 SFP transceiver
Flash access: this form factor is not supported.
MTCA: Out FP:4 FPUNIV:16 RB:40 IFP:2 GPIO:0
dbLoadRecords("evr-mtca-300.db", "DEVICE=EVR0, SYS=tests-mTCA-EVR-300, Link-Clk-SP=88.0525")
iocInit
Starting iocInit
############################################################################
## EPICS R3.14.12.5-2015-08 $Date: Tue 2015-03-24 09:57:35 -0500$
## EPICS Base built Oct  9 2015
############################################################################
Set EVR clock 88052500.000000
iocRun: All initialization complete
epicsEnvSet IOCSH_PS1,"icsb-mtcacpu-ct028> "
icsb-mtcacpu-ct028>
```

In addition, the PCI information is available within the running IOC via `devPCIShow` as follows:

```
icsb-mtcacpu-ct028> devPCIShow
Look for the line with your configuration information, in our case is:
PCI 0000:06:00.0 IRQ 18
  vendor:device 10ee:7011 rev 00
Where the vendor id 10ee is Xilinx Corporation.
And show the PCI information with (the second parameter is the verbosity level):
icsb-mtcacpu-ct028> devPCIShow 9 0x10ee
PCI 0000:06:00.0 IRQ 18
  vendor:device 10ee:7011 rev 00
  subved:subdev 1a3e:132c
  class 118000 generic signal processing controller
  driver mrf-pci
```

```
  BAR 0 32-bit MMIO    256 kB
Matched 1 devices
icsb-mtcacpu-ct028>
```

5 System In-Situ Verification Procedure

This chapter provides the minimal system verification procedure. If one wants to do more step-by-step procedure which may be useful when testing the function of hardware and software, please see Reference [see 2, p14].

Step	Goal	Info.
1	Check the EVR & EVR connection	Link status, link clock, and heartbeat timeout counter
2	Monitor Receiving and acknowledging Events	Event counter and receiving event frequency
3	Generate Trigger Signals from EVR	Various trigger signals with an oscilloscope

Table 5 System In-Situ Verification Procedure

5.1 Step 1 : Check the EVR and EVR connection

Short comments on each command or a series of commands are shown before the corresponding command.

```
#
# We can check the EVG and EVR link status and the link clock setting, and can also see the link
    down counter as well.
#
[user@host ~]$ caget tests-mTCA-EVR-300-EVR0:Link-Sts
tests-mTCA-EVR-300-EVR0:Link-Sts OK
[user@host ~]$ caget tests-mTCA-EVR-300-EVR0:Link-Clk-I
tests-mTCA-EVR-300-EVR0:Link-Clk-I 88.0519
#
# Change the wrong clock setting on EVR, then we expect that the link status will be Fail.
#
[user@host ~]$ caput tests-mTCA-EVR-300-EVR0:Link-Clk-SP 100
Old : tests-mTCA-EVR-300-EVR0:Link-Clk-SP 88.0525
New : tests-mTCA-EVR-300-EVR0:Link-Clk-SP 100
[user@host ~]$ caget tests-mTCA-EVR-300-EVR0:Link-Sts
tests-mTCA-EVR-300-EVR0:Link-Sts Fail
[user@host ~]$ caget tests-mTCA-EVR-300-EVR0:Link-Clk-I
tests-mTCA-EVR-300-EVR0:Link-Clk-I 100
#
# Revert it back to the proper clock setting, and the link status will be OK.
#
[user@host ~]$ caput tests-mTCA-EVR-300-EVR0:Link-Clk-SP 88.0525
Old : tests-mTCA-EVR-300-EVR0:Link-Clk-SP 100
New : tests-mTCA-EVR-300-EVR0:Link-Clk-SP 88.0525
[user@host ~]$ caget tests-mTCA-EVR-300-EVR0:Link-Sts
tests-mTCA-EVR-300-EVR0:Link-Sts OK
#
# There is a link heartbeat counter that gets updated aproximately every 1.3 s when the link is
    off. Camonitor it to check its functionality.
```

```
#
[user@host ~]$ camonitor tests-mTCA-EVR-300-EVR0:Cnt-LinkTimo-I
tests-mTCA-EVR-300-EVR0:Cnt-LinkTimo-I 2016-09-12 11:49:11.463914 35
#
# Open another terminal, to change the wrong link clock.
#
[user@host ~]$ caput tests-mTCA-EVR-300-EVR0:Link-Clk-SP 100
Old : tests-mTCA-EVR-300-EVR0:Link-Clk-SP 88.0525
New : tests-mTCA-EVR-300-EVR0:Link-Clk-SP 100
#
# The heartbeat counter should be increasing as follows:
#
tests-mTCA-EVR-300-EVR0:Cnt-LinkTimo-I 2016-09-12 11:50:06.760264 36
tests-mTCA-EVR-300-EVR0:Cnt-LinkTimo-I 2016-09-12 11:50:08.056687 37
tests-mTCA-EVR-300-EVR0:Cnt-LinkTimo-I 2016-09-12 11:50:09.353116 38
tests-mTCA-EVR-300-EVR0:Cnt-LinkTimo-I 2016-09-12 11:50:10.649505 39
tests-mTCA-EVR-300-EVR0:Cnt-LinkTimo-I 2016-09-12 11:50:11.945937 40
#
# The counter will be stopped after the proper value is given.
#
```

5.2 Step 2 : Monitor Receiving and acknowledging Events

For this step, an additional EVR database file is needed to add to the end line of the `mTCA-EVR-300_1.cmd` file. The new start-up script file, `mTCA-EVR-300_2.cmd` is shown in Listing 5.1.

```
1  epicsEnvSet("SYS"             "tests-mTCA-EVR-300")
2  epicsEnvSet("DEVICE"          "EVR0")
3  epicsEnvSet("EVR_PCIDOMAIN"   "0x0")
4  epicsEnvSet("EVR_PCIBUS"      "0x6")
5  epicsEnvSet("EVR_PCIDEVICE"   "0x0")
6  epicsEnvSet("EVR_PCIFUNCTION" "0x0")
7
8  require mrfioc2,2.7.13
9
10 mrmEvrSetupPCI($(DEVICE), $(EVR_PCIDOMAIN), $(EVR_PCIBUS), $(EVR_PCIDEVICE), $(EVR_PCIFUNCTION))
11
12 dbLoadRecords("evr-mtca-300.db", "DEVICE=$(DEVICE), SYS=$(SYS), Link-Clk-SP=88.0525")
13
14 dbLoadRecords("evr-softEvent.template", "DEVICE=$(DEVICE), SYS=$(SYS), EVT=14, CODE=14")
```

Listing 5.1 Start-up script `mTCA-EVR-300_2.cmd`.

Short comments on each command or a series of commands are shown before the corresponding command.

```
1  #
2  # start the IOC
3  #
4  user@host:~$ iocsh mTCA-EVR-300_2.cmd
5
6  #
7  # Monitor the Event counter with 14 and check the time difference between counters, e.g., 278 and
       279, is 0.071429s, i.e., 14 Hz.
8  # In another terminal:
9  [user@host ~]$ camonitor tests-mTCA-EVR-300-EVR0:Event-14-Cnt-I
```

```
tests-mTCA-EVR-300-EVR0:Event-14-Cnt-I 2016-09-12 11:52:57.724374 278
tests-mTCA-EVR-300-EVR0:Event-14-Cnt-I 2016-09-12 11:52:57.795803 279
tests-mTCA-EVR-300-EVR0:Event-14-Cnt-I 2016-09-12 11:52:57.867231 280
tests-mTCA-EVR-300-EVR0:Event-14-Cnt-I 2016-09-12 11:52:57.938660 281
tests-mTCA-EVR-300-EVR0:Event-14-Cnt-I 2016-09-12 11:52:58.010099 282
```

5.3 Step 3 : Generate Trigger Signals from EVR

In this step output triggers will be generated after receiving an event from the event generator. The mTCA-EVR-300 has 3 types of outputs: 4 front panel outputs, 16 front universal outputs (through the IFB-300 extension board) and 40 rear outputs. The front and front universal outputs can be checked with an oscilloscope, and the rear outputs with an AMC capable of acknowledging the triggers, such as a data acquisition board Struck SIS8300.

The start-up script file for the front and front universal outputs, `mTCA-EVR-300_3.cmd` is shown in Listing 5.2.

```
epicsEnvSet("SYS"             "tests-mTCA-EVR-300")
epicsEnvSet("DEVICE"          "EVR0")
epicsEnvSet("EVR_PCIDOMAIN"   "0x0")
epicsEnvSet("EVR_PCIBUS"      "0x6")
epicsEnvSet("EVR_PCIDEVICE"   "0x0")
epicsEnvSet("EVR_PCIFUNCTION" "0x0")

require mrfioc2,2.7.13

mrmEvrSetupPCI($(DEVICE), $(EVR_PCIDOMAIN), $(EVR_PCIBUS), $(EVR_PCIDEVICE), $(EVR_PCIFUNCTION))

dbLoadRecords("evr-mtca-300.db", "DEVICE=$(DEVICE), SYS=$(SYS), Link-Clk-SP=88.0525")

dbLoadRecords("evr-softEvent.template", "DEVICE=$(DEVICE), SYS=$(SYS), EVT=14, CODE=14")

dbLoadRecords("evr-pulserMap.template", "DEVICE=$(DEVICE), SYS=$(SYS), PID=0, F=Trig, ID=0, EVT
    =14")
dbLoadRecords("evr-pulserMap.template", "DEVICE=$(DEVICE), SYS=$(SYS), PID=1, F=Trig, ID=0, EVT
    =14")
```

Listing 5.2 Start-up script `mTCA-EVR-300_3.cmd`.

An EVR is composed of several logical sub-units, and one of the logical sub-units is the pulse generator. Each pulse generator has an associated Delay and Width [2]. In the following procedure, two pulse generators are mapped to the `OUT0` and `OUT2` of IFB-300, which are connected to the channel 1 and 2 of the oscilloscope respectively in order to see whether output signals are generated according to Delay's and Width's changes. Short comments on each command or a series of commands are shown before the corresponding command.

5.3.1 `OUT0` output

```
#
# Start the IOC
#
```

```
user@host:~$ iocsh mTCA-EVR-300_3.cmd

#
# Open a new terminal and set the Pulse generator 0 to the OUT0 output
#
user@host:~$ caput tests-mTCA-EVR-300-EVR0:FrontOut0-Src-SP 0
Old : tests-mTCA-EVR-300-EVR0:FrontOut0-Src-SP 63
New : tests-mTCA-EVR-300-EVR0:FrontOut0-Src-SP 0

#
# Set the trigger event 14 to the pulse generator 0
#
user@host:~$ caput tests-mTCA-EVR-300-EVR0:Pul0-Evt-Trig0-SP 14
Old : tests-mTCA-EVR-300-EVR0:Pul0-Evt-Trig0-SP    14
New : tests-mTCA-EVR-300-EVR0:Pul0-Evt-Trig0-SP    14

#
# Set the width time of the pulse generator 0. 10 000 is translated to 10 ms.
#
user@host:~$ caput tests-mTCA-EVR-300-EVR0:Pul0-Width-SP 10000
Old : tests-mTCA-EVR-300-EVR0:Pul0-Width-SP        0
New : tests-mTCA-EVR-300-EVR0:Pul0-Width-SP        10000
```

Figure 4 shows the output of `OUT0` in an oscilloscope.

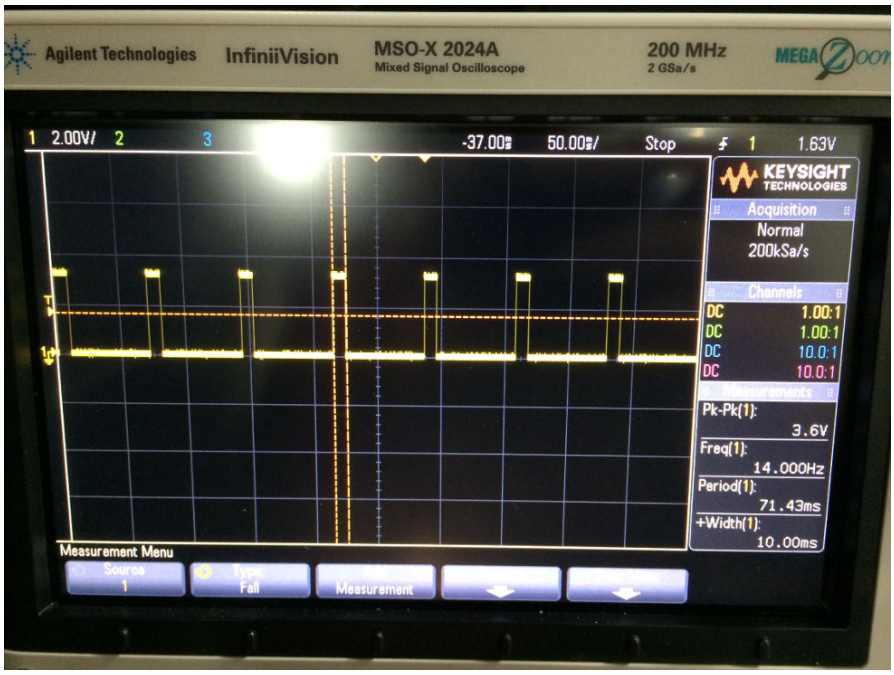

Figure 4 14 Hz signal with 10 ms width

5.3.2 Width Time of Pulse Generator

```
#
```

```
# Change the width time of the pulse generator 0 from 10 ms to 50 ms
#
user@host:~$ caput tests-mTCA-EVR-300-EVR0:Pul0-Width-SP 50000
Old : tests-mTCA-EVR-300-EVR0:Pul0-Width-SP        10000
New : tests-mTCA-EVR-300-EVR0:Pul0-Width-SP        50000
```

and the output is shown in Figure 5.

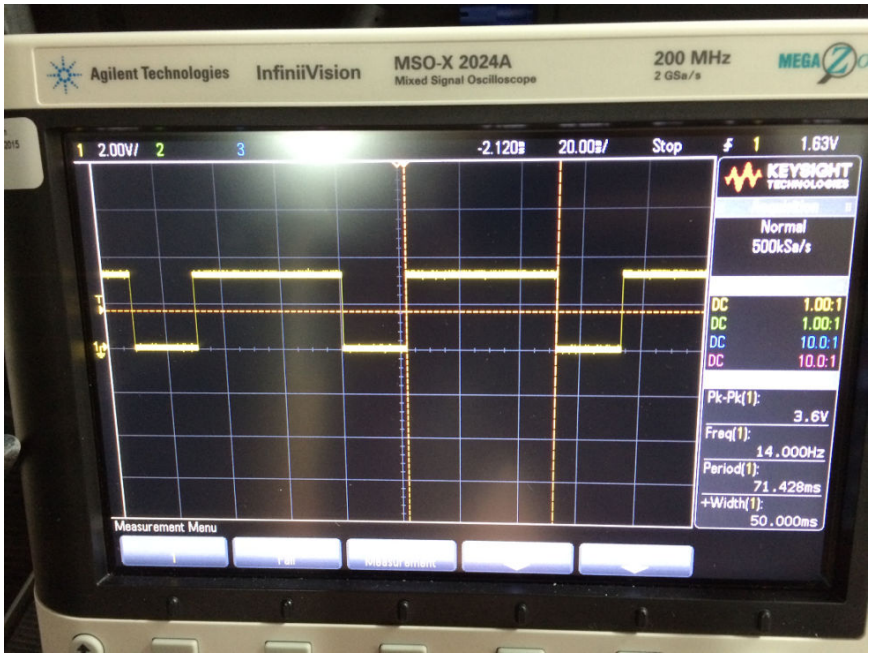

Figure 5 14 Hz signal with 50 ms width

5.3.3 Delay Time of Pulse Generator

```
#
# Set the width time - 20 ms - of the pulse generator 0
#
user@host:~$ caput tests-mTCA-EVR-300-EVR0:Pul0-Width-SP 20000
Old : tests-mTCA-EVR-300-EVR0:Pul0-Width-SP        50000
New : tests-mTCA-EVR-300-EVR0:Pul0-Width-SP        20000

#
# Set the Pulse generator 1 to the OUT2 output
#
user@host:~$ caput tests-mTCA-EVR-300-EVR0:FrontOut2-Src-SP 1
Old : tests-mTCA-EVR-300-EVR0:FrontOut2-Src-SP 63
New : tests-mTCA-EVR-300-EVR0:FrontOut2-Src-SP 0

#
# Set the trigger event 14 to the pulse generator 1
#
user@host:~ $ caput tests-mTCA-EVR-300-EVR0:Pul1-Evt-Trig0-SP 14
Old : tests-mTCA-EVR-300-EVR0:Pul1-Evt-Trig0-SP    14
```

```
New : tests-mTCA-EVR-300-EVR0:Pul1-Evt-Trig0-SP    14

#
# Set the width time - 20 ms - of the pulse generator 1
#
user@host:~$ caput tests-mTCA-EVR-300-EVR0:Pul1-Width-SP 20000
Old : tests-mTCA-EVR-300-EVR0:Pul1-Width-SP        0
New : tests-mTCA-EVR-300-EVR0:Pul1-Width-SP        20000

#
# Set the delay time - 30 ms - of the pulse generator 1
#
user@host:~$ caput tests-mTCA-EVR-300-EVR0:Pul1-Delay-SP 30000
Old : tests-mTCA-EVR-300-EVR0:Pul1-Delay-SP        0
New : tests-mTCA-EVR-300-EVR0:Pul1-Delay-SP        30000
```

Figure 6 shows the result.

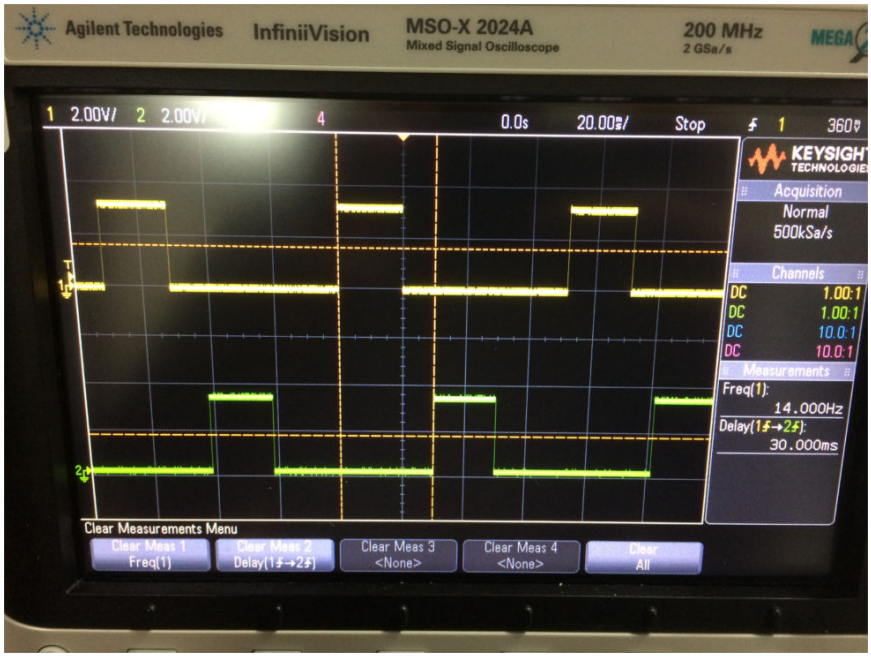

Figure 6 Two 14 Hz signals with 30 ms delay

5.3.4 UNIV2 output

The UNIV0 to UNIV15 outputs in the IFB-300 board work just in the same way as OUT0 to OUT3. To set the pulse generator 0 to the UNIV2 output do:

```
#
# Set the Pulse generator 0 to the UNIV2 output
#
user@host:~$ caput tests-mTCA-EVR-300-EVR0:FrontUnivOut2-Src-SP 0
Old : tests-mTCA-EVR-300-EVR0:FrontUnivOut2-Src-SP 63
New : tests-mTCA-EVR-300-EVR0:FrontUnivOut2-Src-SP 0
```

And set the pulser 0 parameters in the same way as in the previous section. The outputs shown in the oscilloscope should be the same.

5.3.5 Rear outputs

The mTCA-EVR-300 has 40 rear outputs. The initial 32 rear outputs map to the RTM connector, the last 8 rear outputs map to the mTCA backplane. Next it will be shown how to use the rear outputs to trigger the data acquisition in in a Struck SIS8300 board.

The start-up script file for this is `mTCA-EVR-300_4.cmd`, shown in Listing 5.3.

```
epicsEnvSet("SYS"             "EVRTEST")
epicsEnvSet("DEVICE"          "EVRMTCA")
epicsEnvSet("EVR_PCIDOMAIN"   "0x0")
epicsEnvSet("EVR_PCIBUS"      "0x6")
epicsEnvSet("EVR_PCIDEVICE"   "0x0")
epicsEnvSet("EVR_PCIFUNCTION" "0x0")

require recsync
dbLoadRecords(reccaster.db, "P=EVRTEST:recsync:")

require mrfioc2

mrmEvrSetupPCI($(DEVICE), $(EVR_PCIDOMAIN), $(EVR_PCIBUS), $(EVR_PCIDEVICE), $(EVR_PCIFUNCTION))

dbLoadRecords("evr-mtca-300.db", "DEVICE=$(DEVICE), SYS=$(SYS), Link-Clk-SP=88.0525")

dbLoadRecords("evr-softEvent.template", "DEVICE=$(DEVICE), SYS=$(SYS), EVT=14, CODE=14")

dbLoadRecords("evr-pulserMap.template", "DEVICE=$(DEVICE), SYS=$(SYS), PID=0, F=Trig, ID=0, EVT
    =14")

require sis8300

epicsEnvSet("BUFSIZE", "1024")

ndsCreateDevice("sis8300", "SIS8300", "FILE=/dev/sis8300-5")

dbLoadRecords("sis8300.db", "PREFIX=EVRTEST:DAQ, ASYN_PORT=SIS8300, AI_NELM=1024")
```

Listing 5.3 Start-up script `mTCA-EVR-300_4.cmd`.

The line

```
ndsCreateDevice("sis8300", "SIS8300", "FILE=/dev/sis8300-5")
```

should be changed according the slot occupied by the SIS8300 board, in this case slot 5 of the mTCA crate. It can be checked by:

```
[root@icsb-mtcacpu-ct028 mTCA-EVR-300_tests]# ls /dev/sis8300-*
```

To check that the SIS8300 board gets the trigger from the backplane we will run a CSS BOY screen located in

```
/opt/epics/modules/sis8300/1.12.1/opi/sis8300.opi
```

which is showed in Figure 7.

For CSS to work the machine running it should be in the same network as the mTCA crate and have the EEE filesystem. The ESS Development Machine meets these

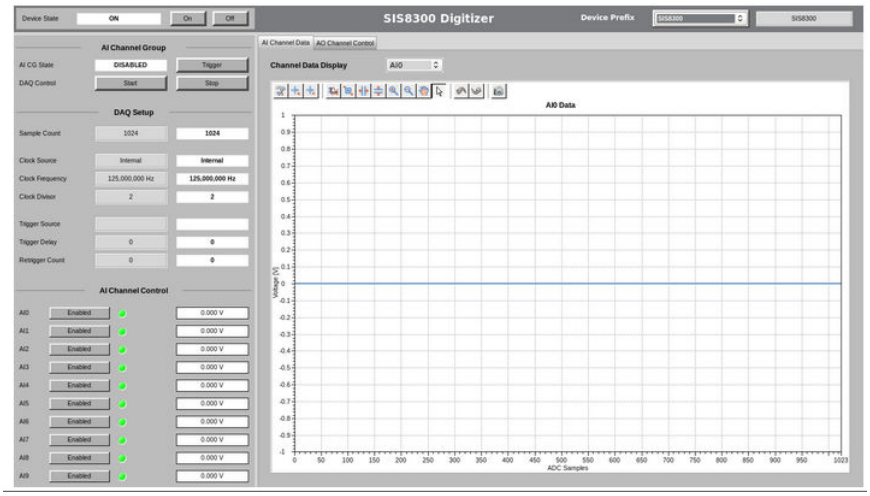

Figure 7 SIS8300 data acquisition OPI screen.

requirements if the PC running it is in the EEE network and connected via VPN to the ICS lab; it's also necessary to set the `EPICS_CA_ADDR_LIST` field in the CSS preferences to the IP address of the mTCA CPU.

CSS can be launched with the necessary parameters by:

```
css --launcher.openFile "/opt/epics/modules/sis8300/1.12.1/opi/sis8300.opi Device2Macro=EVRTEST
    [\58]DAQ"
```

In AI Channel Control, enable AI0, and set Trigger Source `backplane 1` in the DAQ Setup. Click the button DAQ Control `Start` under AI Channel Group to put the SIS8300 in a waiting for trigger state. Note that the graph is empty.

Start the IOC and:

```
#
# Disable Rear Universal Output 33
#
[javiercereijogarcia@javcerwin ~]$ caput EVRTEST-EVRMTCA:RearUniv33-Ena-SP "Disabled"
Old : EVRTEST-EVRMTCA:RearUniv33-Ena-SP Enabled
New : EVRTEST-EVRMTCA:RearUniv33-Ena-SP Disabled
#
# Map Rear Universal Output 33 to pulser o
#
[javiercereijogarcia@javcerwin ~]$ caput EVRTEST-MTCAEVR:RearUniv33-Src-SP 0
Old : EVRTEST-MTCAEVR:RearUniv33-Src-SP 63
New : EVRTEST-MTCAEVR:RearUniv33-Src-SP 0
#
# Map pulser 0 to event 14
#
[javiercereijogarcia@javcerwin ~]$ caput EVRTEST-EVRMTCA:Pul0-Evt-Trig0-SP 14
Old : EVRTEST-EVRMTCA:Pul0-Evt-Trig0-SP    14
New : EVRTEST-EVRMTCA:Pul0-Evt-Trig0-SP    14
#
# Set pulser 0 width to 10 ms
#
[javiercereijogarcia@javcerwin ~]$ caput EVRTEST-EVRMTCA:Pul0-Width-SP 10000
```

```
Old : EVRTEST-EVRMTCA:Pul0-Width-SP        0
New : EVRTEST-EVRMTCA:Pul0-Width-SP        10000
#
# Watch closely to the graph in the CSS OPI screen when enabling Rear Universal Output 33, since
     in the next event 14 received the SIS8300 will start the data acquisition; since it is not
     connected, the data acquired is just 0
#
[javiercereijogarcia@javcerwin ~]$ caput EVRTEST-EVRMTCA:RearUniv33-Ena-SP "Enabled"
Old : EVRTEST-EVRMTCA:RearUniv33-Ena-SP Disabled
New : EVRTEST-EVRMTCA:RearUniv33-Ena-SP Enabled
```

RearUniv33 is used to trigger `backplane 1` because there is an offset of 32 rear outputs in the mTCA-EVR-300 which are mapped to the RTM connector.

Bibliography

[1] MRF Technical Reference. *Event System with Delay Compensation Technical Reference Firmware 0205*, April 26, 2016.

[2] Michael Davidsaver. *EVR User Guide*, August, 2015. URL http://epics.sourceforge.net/mrfioc2/evr-usage.pdf.