Getting started with MRF Timing System

MRF Linux Driver and API

Timing Workshop ICALEPCS 2019

Jukka Pietarinen

Micro-Research Finland Oy

Example System Set Up

MTCA-7S-PH5a (PowerBridge)

- 11850-21 Chassis
- NAT PM-AC600D
- NAT MCH-PHYS
- AM G64/471-41 CPU
  - Ubuntu pre-installed
- MRF mTCA-EVM-300
  Event Master/Generator
- MRF mTCA-EVR-300U
  Event Receiver

Linux Driver

- Available on https://github.com/jpietari/mrf-linux-driver
- Provides memory mapped (mmap) interface do event hardware
- Support for upgrading FPGA firmware
- Compatible with all MRF PCI/PCI Express hardware

Starting from a fresh (pre-installed) Ubuntu installation

- Install ssh (to be able to use system remotely)

```
sudo apt-get install openssh-server
```

- Install git

```
sudo apt-get install git
```

- Update & Upgrade

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get dist-upgrade
```

- Install packages required for building kernel modules

```
sudo apt install dkms build-essential linux-headers-`uname -r`
sudo apt install libelf-dev
sudo apt-get install linux-headers-generic
```

Linux Driver Installation

- Get driver sources

  ```
  git clone https://github.com/jpietari/mrf-linux-driver
  ```

- Build driver

  ```
  cd mrf-linux-driver
  make
  ```

- Add group privileges for user space driver access

  ```
  sudo groupadd -g 502 mrf
  sudo usermod -a -G mrf user
  ```

- Install driver

  ```
  sudo make modules_install
  sudo depmod -a
  ```

- Add udev-rules and load driver

  ```
  sudo ./module_load
  ```

- Driver will be loaded automatically at next reboot
- EVG devices show up as /dev/ega0..3, /dev/egb0..3, /dev/egc0..3, …
- EVR devices show up as /dev/era0..3, /dev/erb0..3, /dev/erc0..3, …

```
user@MTCA-7S-PH5a:~/mrf-linux-api$ ls -al
/dev/eg* /dev/er*
crw-r----- 1 root mrf 238, 0 Sep 30 09:02 /dev/ega0
crw-rw---- 1 root mrf 238, 1 Sep 30 09:02 /dev/ega1
crw-r----- 1 root mrf 238, 2 Sep 30 09:02 /dev/ega2
crw-rw---- 1 root mrf 238, 3 Sep 30 09:02 /dev/ega3
crw-r----- 1 root mrf 239, 0 Sep 30 09:02 /dev/era0
crw-rw---- 1 root mrf 239, 1 Sep 30 09:02 /dev/era1
crw-r----- 1 root mrf 239, 2 Sep 30 09:02 /dev/era2
crw-rw---- 1 root mrf 239, 3 Sep 30 09:02 /dev/era3
```

Driver Issues/TODO

- Does not build automatically – drivers disappear when kernel is upgraded (happens very often with Ubuntu)
- Driver is unsigned (annoying warning/error when installing)

- Conversion of API to use Userspace I/O would get rid of whole driver

Application Programming Interface

- Available on https://github.com/jpietari/mrf-linux-api
- Functions for programming EVG/EVR in C
- Shell wrapper functions to control devices from command line

Requirements for Building (Ubuntu installation)

- See slide 3 for packages needed for building driver
- For building documentaton doxygen and latex are needed
- Install doxygen

```
sudo apt-get install doxygen
```

- Install latex

```
sudo apt-get install texlive-full
```

API Installation

- Get sources

  ```
  git clone https://github.com/jpietari/mrf-linux-api
  ```

- Build api, shell wrappers and documentation

  ```
  cd mrf-linux-api
  ```

  ```
  make
  ```

mmap_test – Tool to access registers

- Commands
  - d – dump memory block of 128 bytes
  - m – modify memory
  - q – quit

mmap_test – Display memory

- Usage: d [<address offset>]

```
user@MTCA-7S-PH5a:~/mrf-linux-api/api$ ./mmap_test /dev/era3
Buffer 0x7fcf9449c000
test-> d 0
00: 00010000 00000000 00000007 00000000
10: 00000000 00000000 00000000 40000000
20: 00001000 00110000 00110000 180e0207
30: 00000000 00000000 00000000 00000000
40: 00000000 00000000 00000000 00000058
50: 40000200 00000000 00000000 00000000
60: 00000000 00000000 00000000 00000000
70: 00000000 00000000 00000000 00000000
test-> d
80: 079e41ed 00000000 00000003 00000000
90: 00000000 000000ff 00000000 00000000
a0: 00000000 00000030 00000000 00000000
b0: 00080000 00000000 00000000 00000000
c0: 00000000 00000000 00000000 00000000
d0: 00000000 00000000 00000000 00000000
e0: 0000003f 00000000 00000000 00000000
f0: 00000000 00000000 00000000 00000000
test->
```

mmap_test – Modify memory

- Usage: m [<address offset>][,<size in bytes, 1, 2 or 4>]

```
test-> m 8
00000008: 00000007 ? 7
0000000c: 00000000 ? .
test-> m 0xb,1
0000000b: 05 ? 05
0000000c: 00 ? .
test->
```

System Hardware Setup

- EVG/EVM and EVR in the same crate
- Duplex fiber connected from EVM port 1 to EVR
- EVR TTL Outputs connected to oscilloscope

EVG/EVM:

1. Set up event frequency – we will use internal reference programmed to 125 MHz
2. Set up delay compensation

EVR:

1. Set up event frequency 125 MHz (reference clock needed by GTX to be able to lock to EVG/EVM clock)
2. Set up delay compensation with target delay
3. Clear RX violation flag & check status

Setting up EVM

Set up EVM to use internal event clock generator (fractional synthesizer) and disable external RF divider:

```
jpietari@MTCA-7S-PH5a:~$ cd wrapper
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvgSetRFInput /dev/ega3 0 12
```

Set up event clock frequency:

```
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvgSetEventFrequency /dev/ega3 125
124.997067
```

Enable the EVM to operate as Delay compensation master:

```
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvgSystemMasterEnable /dev/ega3 1
1
```

Enable Beacon event required by delay compensation:

```
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvgBeaconEnable /dev/ega3 1
1
```

Setting up EVR

Set up EVR reference clock:

```
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvrSetEventFrequency /dev/era3 125
124.997067
```

Enable EVR Delay compensation mode:

```
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvrDCEnable /dev/era3 1
1
```

Set up EVR Target delay:

```
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvrSetTargetDelay /dev/era3 0x200000
2097152
```

Clear violation (and verify link is up if we can clear the violation flag):

```
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvrGetViolation /dev/era3 1
0
```

Check Delay Compensation Status:

```
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvrDumpDC /dev/era3
DC Enable 1
DC Status 0x0100
DC Delay Target 00200000, (1000.000000 ns)
DC Delay Value 001fb6e0, (991.073608 ns)
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvrDumpDC /dev/era3
DC Enable 1
DC Status 0x0701
DC Delay Target 00200000, (1000.000000 ns)
DC Delay Value 0020000c, (1000.005722 ns)
```

Working with Events

- EVR has an event FIFO to store received events
- We use evr_fifo_monitor tool in API to show received events

Tasks:

- Enable EVR
- Set up EVR mapping RAM to store event codes 0x01 to 0x04 in FIFO
- Enable mapping RAM
- Start evr_fifo_monitor

Setting up EVR (2)

Enable EVR:

```
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvrEnable /dev/era3 1
1
```

Enable events to be stores into event FIFO:

```
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvrSetFIFOEvent /dev/era3 0 1 1
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvrSetFIFOEvent /dev/era3 0 2 1
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvrSetFIFOEvent /dev/era3 0 3 1
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvrSetFIFOEvent /dev/era3 0 4 1
```

Enable map RAM 0:

```
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvrMapRamEnable /dev/era3 0 1
```

Start evr_fifo_monitor tool (in another window):

```
jpietari@MTCA-7S-PH5a:~/api$ ./evr_fifo_monitor /dev/era3
```

Sending Events with EVG

- EVG can send software events

  Tasks:
- Enable EVG
- Enable EVG software events
- Send software events
- Monitor events received by EVR

Sending Software events with EVG/EVM

Enable EVG:

```
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvgEnable /dev/ega3 1
1
```

Enable software events:

```
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvgSWEventEnable /dev/ega3 1
1
```

Send software events:

```
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvgSendSWEvent /dev/ega3 1
1
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvgSendSWEvent /dev/ega3 2
1
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvgSendSWEvent /dev/ega3 3
1
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvgSendSWEvent /dev/ega3 4
1
```

evr_fifo_monitor shows received events (timestamping has not been set up yet):

```
jpietari@MTCA-7S-PH5a:~/api$ ./evr_fifo_monitor /dev/era3
FIFO Code 00000001, 00000000:00000000
FIFO Code 00000002, 00000000:00000000
FIFO Code 00000003, 00000000:00000000
FIFO Code 00000004, 00000000:00000000
```

Setting up MXC and trigger events with EVG/EVM

- We set up a multiplexed counter to run and generate an event at 1 Hz

  Tasks:
- Set MXC0 prescaler to 125000000
- Enable MXC0 to trigger Trigger event 0
- Set up Trigger event 0 to produce event code 1

- Set up timestamp counter on EVR to count with 1us resolution

Setting up MXC and trigger events with EVG/EVM

Set up multiplexed counter 0 (MXC0) for 1 Hz:

```
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvgSetMXCPrescaler /dev/ega3
Usage: ./EvgSetMXCPrescaler /dev/ega3 <MXC> [<prescaler_value>]
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvgSetMXCPrescaler /dev/ega3 0 125000000
125000000
```

Reset/synchronize multiplexed counters:

```
./EvgSyncMxc /dev/ega3
```

Set up MXC0 to trigger Trigger event 0:

```
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvgSetMXCTrigMap
Usage: ./EvgSetMXCTrigMap /dev/ega3 <MXC> [<map_value>]
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvgSetMXCTrigMap /dev/ega3 0 0
```

Set up Trigger event 0 to produce event code 01:

```
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvgSetTriggerEvent /dev/ega3
Usage: ./EvgSetTriggerEvent /dev/ega3 <trigger event> [<event code> <enable>]
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvgSetTriggerEvent /dev/ega3 0 1 1
1 1
```

Setting up MXC and trigger events with EVG/EVM

Set up EVR internal counter for timestamping:

```
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvrSetTimestampDivider /dev/era3 125

evr_fifo_monitor starts to display event at 1 Hz:

FIFO Code 00000001, 00000000:00084283
FIFO Code 00000001, 00000000:001784c3
FIFO Code 00000001, 00000000:0026c703
FIFO Code 00000001, 00000000:00360943
```

Setting up a sequence on EVG/EVM

- We set up a sequence of events that is triggered at 1 Hz. We want to produce the following events:
  - 2, 5 event clock cycles after triggering
  - 3, 10 event clock cycles after triggering
  - 4, 50 event clock cycles after triggering

Tasks:

- Write events into sequence RAM
- Append end-of-sequence event 127
- Set up sequence triggering and enable

Setting up a sequencer on EVG/EVM

Set sequence RAM 0 events:

```
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvgSetSeqRamEvent /dev/ega3
Usage: ./EvgSetSeqRamEvent /dev/ega3 <ram> <pos> <timestamp> <event code>
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvgSetSeqRamEvent /dev/ega3 0 0 5 2
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvgSetSeqRamEvent /dev/ega3 0 1 10 3
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvgSetSeqRamEvent /dev/ega3 0 2 50 4
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvgSetSeqRamEvent /dev/ega3 0 3 100 127
```

Dump sequence RAM contents:

```
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvgSeqRamDump /dev/ega3 0
Ram0: Timestamp 00000000 Code 02
Ram0: Timestamp 0000000a Code 03
Ram0: Timestamp 00000032 Code 04
Ram0: Timestamp 00000064 Code 7f
```

Set up sequence RAM triggering:

```
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvgSeqRamControl /dev/ega3
Usage: ./EvgSeqRamControl /dev/ega3 <ram> <enable> <single> <recycle> <reset> <trigsel>
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvgSeqRamControl /dev/ega3 0 1 0 0 0 0
```

Setting up a sequencer on EVG/EVM

evr_fifo_monitor starts to display sequence RAM events and MXC0 event at 1 Hz:

```
FIFO Code 00000001, 00000000:691e0103
FIFO Code 00000002, 00000000:691e0103
FIFO Code 00000003, 00000000:691e0103
FIFO Code 00000004, 00000000:691e0104
FIFO Code 00000001, 00000000:692d4343
FIFO Code 00000002, 00000000:692d4343
FIFO Code 00000003, 00000000:692d4343
FIFO Code 00000004, 00000000:692d4344
FIFO Code 00000001, 00000000:693c8583
FIFO Code 00000002, 00000000:693c8583
FIFO Code 00000003, 00000000:693c8583
FIFO Code 00000004, 00000000:693c8584
```

Setting up pulse generators and HW output on EVR

- We want to see actual hardware signals on EVR
- We setup pulse generators for each event code 1 through 4 and map the pulse generator outputs to front panel TTL outputs

Tasks:

- Set up mapping RAM to trigger pulse generator 0 from event 1, pulse generator 1 from event 2, etc.
- Set up pulse generators with pulse delay of 10 event clock cycles and pulse widths from 100 to 400 event clock cycles
- Map pulse generators to front panel outputs
- Configure and enable pulse generators

Setting up EVR HW Outputs

Setup event code to pulse generator trigger mappings:

```
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvrSetPulseMap
Usage: ./EvrSetPulseMap /dev/era3 <ram> <code> <trig> <set> <clear>
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvrSetPulseMap /dev/era3 0 1 0 -1 -1
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvrSetPulseMap /dev/era3 0 2 1 -1 -1
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvrSetPulseMap /dev/era3 0 3 2 -1 -1
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvrSetPulseMap /dev/era3 0 4 3 -1 -1
```

Setup pulse generator parameters, prescaler, delay, width:

```
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvrSetPulseParams
Usage: ./EvrSetPulseParams /dev/era3 <pulse> <presc> <delay> <width>
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvrSetPulseParams /dev/era3 0 0 10 100
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvrSetPulseParams /dev/era3 1 0 10 200
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvrSetPulseParams /dev/era3 2 0 10 300
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvrSetPulseParams /dev/era3 4 0 10 400
```

Setup Front Panel TTL output mappings:

```
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvrSetFPOutMap
Usage: ./EvrSetFPOutMap /dev/era3 <output> <map>
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvrSetFPOutMap /dev/era3 0 0x3f00
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvrSetFPOutMap /dev/era3 1 0x3f01
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvrSetFPOutMap /dev/era3 2 0x3f02
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvrSetFPOutMap /dev/era3 3 0x3f03
```

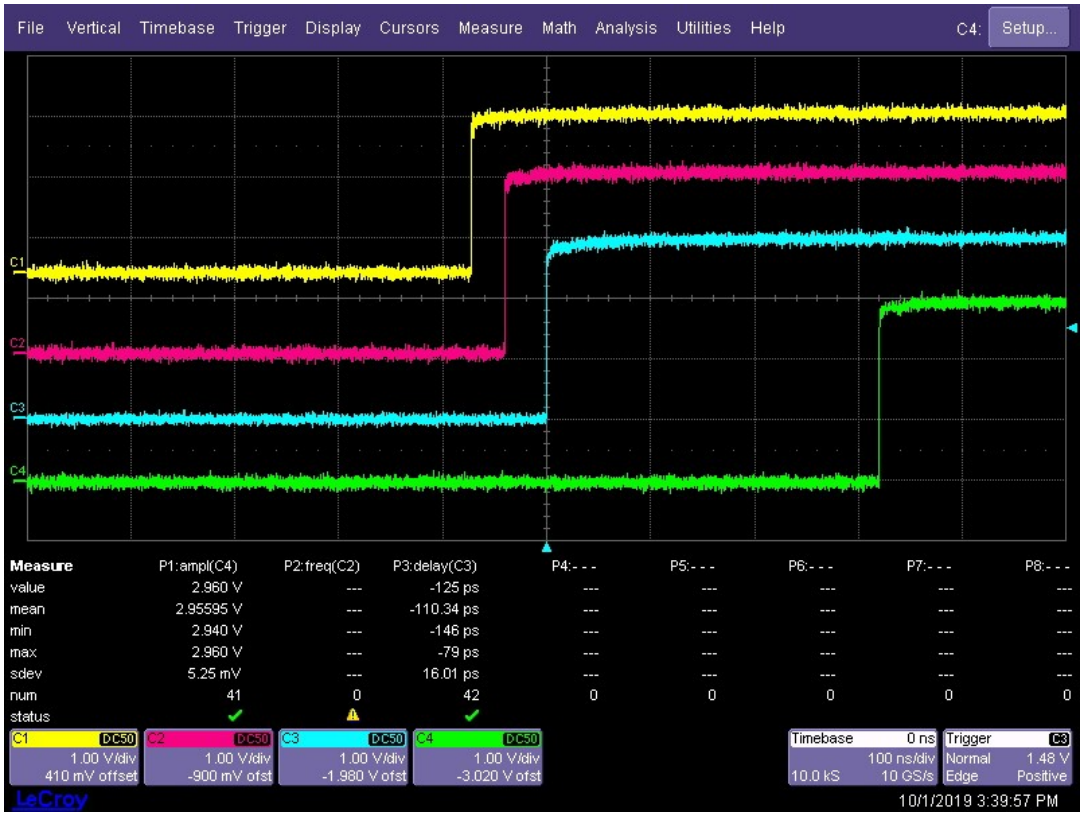
Setting up EVR HW Outputs (2)

Setup pulse generator properties, enable triggering and enable pulse generators:

```
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvrSetPulseProperties /dev/era3
Usage: ./EvrSetPulseProperties /dev/era3 <pulse> <polarity> <map_reset_ena>
<map_set_ena> <map_trigger_ena> <enable>
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvrSetPulseProperties /dev/era3 0 0 0 0 1 1
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvrSetPulseProperties /dev/era3 1 0 0 0 1 1
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvrSetPulseProperties /dev/era3 2 0 0 0 1 1
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvrSetPulseProperties /dev/era3 3 0 0 0 1 1
```

CH1 (yellow): TTL OUT0 mapped to pulse gen. 0
CH2 (pink): TTL OUT1 mapped to pulse gen. 1
CH3 (blue): TTL OUT2 mapped to pulse gen. 2
CH4 (green): TTL OUT3 mapped to pulse gen. 3

Pulse gen. 0 triggered by event 0x01 (seq. Trigger)
Pulse gen. 1 triggered by event 0x02 (seq. event #1)
Pulse gen. 2 triggered by event 0x03 (seq. event #2)
Pulse gen. 3 triggered by event 0x04 (seq. event #3)

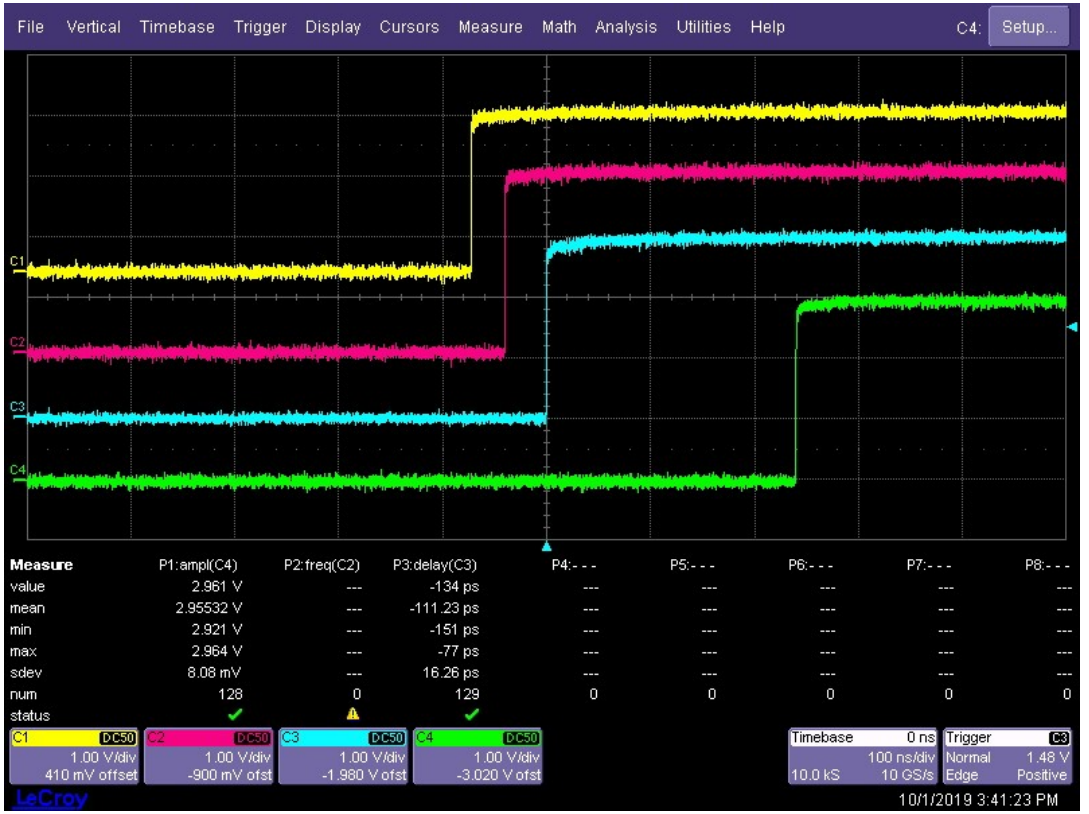


Setting up EVR HW Outputs (3)

Change delay of pulse generator 3 from 10 to 0:

```
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvrSetPulseParams /dev/era3 3 0 0 300
```

CH4 (green): pulse has moved by -80 ns



Setting up EVR HW Outputs (4)

We move event 3 in sequence by -10 cycles (timestamp changed from 50 to 40):

```
jpietari@MTCA-7S-PH5a:~/wrapper$ ./EvgSetSeqRamEvent /dev/ega3 0 2 40 4
```

CH4 (green): pulse has moved another -80 ns