

# EVR Tutorials

## Contents

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Introduction</b>  | <b>2</b>  |
| <b>2</b> | <b>Quick start</b>   | <b>2</b>  |
| <b>3</b> | <b>Generate a pulse upon receiving an event</b>                              | <b>4</b>  |
| 3.1      | Instructions: . . . . .  | 5         |
| 3.2      | Substitution snippet explanation: . . . . .                                  | 6         |
| <b>4</b> | <b>Trigger an EPICS event upon receiving an event from the timing system</b> | <b>6</b>  |
| 4.1      | Instructions: . . . . .  | 7         |
| 4.2      | Substitution snippet explanation: . . . . .                                  | 7         |
| <b>5</b> | <b>Generate a clock signal</b>   | <b>7</b>  |
| 5.1      | Instructions: . . . . .  | 8         |
| 5.2      | Substitution snippet explanation: . . . . .                                  | 8         |
| <b>6</b> | <b>Output a Distributed Bus bit</b>  | <b>9</b>  |
| 6.1      | Instructions: . . . . .  | 9         |
| 6.2      | Substitution snippet explanation: . . . . .                                  | 10        |
| <b>7</b> | <b>GUI</b>   | <b>10</b> |

# 1 Introduction

A timing system consists of an event generator (EVG), a series of event receivers (EVR), software controlling them and a timing network. EVG generates a series of events, which are delivered to EVRs through a timing network. An EVR is then configured to respond to specific events in various ways, including processing EPICS records and generating pulses, synchronized clock or custom signals on its outputs. This document contains step-by-step instructions to configuring some of the basic functionalities of the event receiver. A detailed EVR manual is available in [2].

## 2 Quick start

To set up a timing system we need a VME crate, a Single Board Computer (SBC) and an EVR. A VME crate has a number of slots where SBC, EVR and other components can be inserted. Slot numbering should be checked with the VME crate documentation. The tutorials in this document are written for the following setup:

- a VME64x IFC 1210 Single Board Computer inserted into VME crate slot 1 (how to set up IFC 1210 [3]),
- VME-EVR-230RF event receiver inserted in slot 2,
- the EVR connected to the timing network through an optical cable.

To set up an IOC application for EVR we need to set up a startup script and a substitution file. The following steps demonstrate how to prepare such a SWIT compatible IOC application:

1. Create a project folder, eg. `MTEST-VME-EVRTEST` and a sub-folder named `cfg` in your project folder: `MTEST-VME-EVRTEST/cfg/`

```
mkdir MTEST-VME-EVRTEST
cd MTEST-VME-EVRTEST
mkdir cfg
```

2. Copy the startup script `/fin/dev1/iocBoot/templates/EVR/example.startup` to your project and rename it to `MTEST-VME-EVRTEST_startup.script` using the following command in your project folder:

```
cp /fin/devl/iocBoot/templates/EVR/example.startup
MTEST-VME-EVRTEST_startup.script
```

The startup script should look similar to:

```
## This will be set automatically from
INSTBASE + MRFIOC2 version in the future
epicsEnvSet TEMPLATE_DIR "/fin/devl/iocBoot/templates/EVR"

##System configuration
epicsEnvSet SYS MTEST-VME-EVRTEST

### EVR configuration
#epicsEnvSet EVR EVR0 ##EVR name (default EVR0)
#epicsEnvSet EVR_SLOT 3 ##EVR slot (default slot 2)
#epicsEnvSet EVR_MEMOFFSET 0xffff ## A24 base address
                                (default value 0x3000000)
#epicsEnvSet EVR_IRQLINE 0xff ## IRQ level (default value 5)
#epicsEnvSet EVR_IRQVECT 0xff ## IRQ vector (default value 0x26)

< $(TEMPLATE_DIR)/EVR_VME.startup

## END OF EVR configuration
```

The configurable variables in the startup script are:

- `TEMPLATE_DIR` is the location of the template files, and must not be changed by the user.
- `SYS` is the system name. **This variable is mandatory.**
- `EVR` is the event receiver name. If the variable is not defined in the startup script, it defaults to `EVR0`.
- `EVR_SLOT` is the VME crate slot where EVR is inserted. If the variable is not defined in the startup script, it defaults to 2.
- The the base A24 address (`EVR_MEMOFFSET`), interrupt level (`EVR_IRQLINE`) and interrupt vector (`EVR_IRQVECT`) variables configure the interaction between the SBC and the EVR. The details are out of scope of this document. If a variable is not defined in the startup script, it gets set to its default value.

Using the above startup script, the system name is set to *MTEST-VME-EVRTEST*, and the event receiver named *EVR0* is placed in the physical slot 2 of the VME crate. It uses default A24 address and interrupt configuration.

Line `< $(TEMPLATE_DIR)/EVR_VME.startup` includes a generic EVR startup script, and must not be changed by the user.

3. Copy the substitution file `/fin/dev1/iocBoot/templates/EVR/EVR.subs` to `MTEST-VME-EVRTEST/cfg/EVR.subs`. This substitution file can always be used as a starting point for new applications. Use the following command in your project folder:

```
cp /fin/dev1/iocBoot/templates/EVR/EVR.subs cfg/
```

The macro definitions in the substitution file are used to configure the EVR. All the available macros are already present in the substitution file and set to their default values, so the user can simply change the desired values. Detailed description of the substitution file is available in the EVR manual [2].

### 3 Generate a pulse upon receiving an event

EVR has a number of pulsers available and each of them can generate a pulse upon receiving an event. The pulse can then be outputted through desired EVR outputs.

This tutorial demonstrates how to configure an EVR to generate a 80 ns wide pulse, 40 ns after each reception of event 4, as seen in Figure 1.

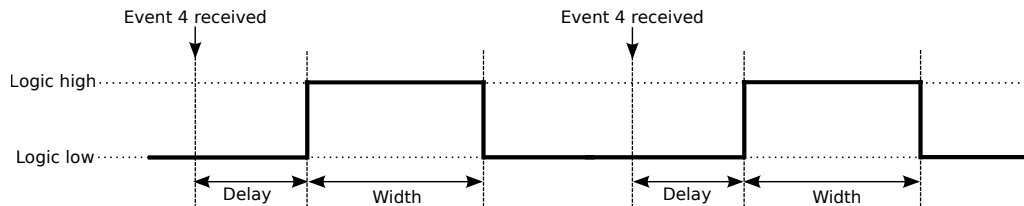


Figure 1: An example of a pulse generated after the reception of the event 4.

The pulse in this tutorial is generated using pulser 3 and outputted through the front panel TTL output 0 (FrontOut0), as seen in Figure 2.

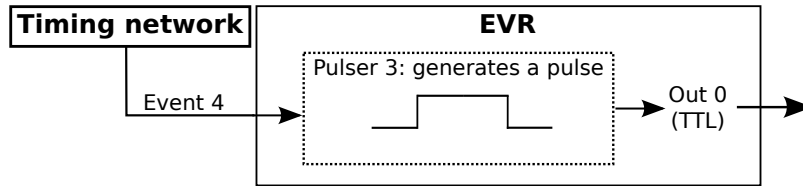


Figure 2: Use pulser 3 to generate a pulse upon reception of the event 4. The pulse is outputted through front panel TTL output 0.

### 3.1 Instructions:

1. If starting a new IOC application, consult the quick start in Section 2.
2. Set the macro values in the substitution file (`MTEST-VME-EVRTEST/cfg/EVR.subs`) according to this snippet (explained in 3.2):

```

file "$(TEMPLATE_DIR)/evr-vmerf230.template"
{
    {
        ...
        Pul3-Delay-SP=40,
        Pul3-Width-SP=80,
        ...
        FrontOut0-Src-SP=3,
        ...
    }
}

file "$(TEMPLATE_DIR)/evr-pulserMap.template"{
pattern { PID    F,          EVT, ID}
        ...
        { 3,      Trig,      4,    0 }
        ...
}

```

3. Optionally, you can remove all the macros whose values you did not change.
4. Install the prepared IOC by running command `swit -V` from your project folder `MTEST-VME-EVRTEST`.

## 3.2 Substitution snippet explanation:

First we set up the pulse generator 3 (Pul3):

- `Pul3-Delay-SP=40`: Set the delay between the reception of the event and the start of the pulse (pulse rising edge) for pulser 3 to 40 ns.
- `Pul3-Width-SP=80`: Set the pulse width (time between the pulse rising and falling edge) for pulser 3 to 80 ns.

Then the value of the output source macro `FrontOut0-Src-SP` is set to 3, which configures the front panel output 0 (`FrontOut0`) to use pulser 3 as its source. Macro values 0-15 correspond to pulsers 0-15. A complete list of available values is available in the EVR manual [2].

Finally, the Pulser 3 is set to trigger on reception of the event 4:

- `PID`: Select Pulser 3
- `F`: Select the *Trigger* function of the pulser
- `EVT`: Map Pulser 3 Trig function to event 4
- `ID`: Unique ID for each PID-F combination.

In order to use different pulser simply change the pulser number, eg. using `Pul5-Delay-SP` instead of `Pul3-Delay-SP` sets the delay of pulser 5 instead of pulser 3. Similar is for outputs, eg. using `FrontOut1-Src-SP` instead of `FrontOut0-Src-SP` sets the output source signal of front panel output 1 instead of front panel output 0.

## 4 Trigger an EPICS event upon receiving an event from the timing system

Using the macros in the substitution file it is possible to configure triggering of the EPICS events. Each event from the timing system can be configured to trigger an EPICS event.

This tutorial demonstrates how to trigger an EPICS event number 1 upon reception of event 1 from the timing system.

## 4.1 Instructions:

1. If starting a new IOC application, consult the quick start in Section 2.
2. Set the macro values in the substitution file (`MTEST-VME-EVRTEST/cfg/EVR.subs`) according to this snippet (explained in 4.2):

```
file "${TEMPLATE_DIR}/evr-softEvent.template"{
pattern { EVT,      CODE }
        { "1",      "1"}
        ...
}
```

3. Optionally, you can remove all the macros whose values you did not change.
4. Install the prepared IOC by running command `swit -V` from your project folder `MTEST-VME-EVRTEST`.

## 4.2 Substitution snippet explanation:

An EPICS event 1 (`CODE=1`) is triggered upon reception of the event a (`EVT=1`) from the timing system. It is suggested that macros `EVT` and `CODE` are set to the same value for simplicity, all-though this is not mandatory.

## 5 Generate a clock signal

Event receivers have synchronized event clock across the timing system (the same phase and frequency). The event clock can be prescaled and mapped to the EVR output.

This tutorial demonstrates how to configure the prescaler 0 (PS0) to divide the event clock frequency by 2, and output it through the front panel output 1 (FrontOut1), as seen in Figure 3.

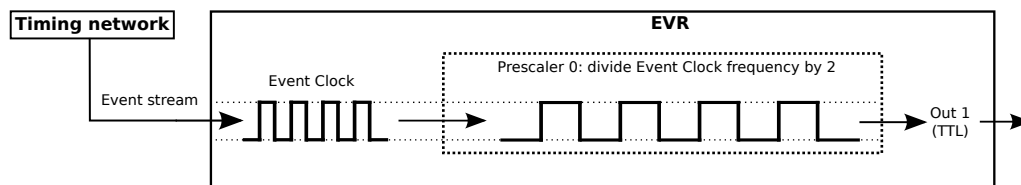


Figure 3: An example clock signal generation

## 5.1 Instructions:

1. If starting a new IOC application, consult the quick start in Section 2.
2. Set the macro values in the substitution file (`MTEST-VME-EVRTEST/cfg/EVR.subs`) according to this snippet (explained in 5.2):

```
file "${TEMPLATE_DIR}/evr-vmef230.template"
{
    {
        ...
        PS0-Div-SP=2,
        ...
        FrontOut1-Src-SP=40,
        ...
    }
}
```

3. Optionally, you can remove all the macros whose values you did not change.
4. Install the prepared IOC by running command `swit -V` from your project folder `MTEST-VME-EVRTEST`.

## 5.2 Substitution snippet explanation:

- `PS0-Div-SP=2`: Set the Prescaler 0 to divide event clock frequency by 2.
- `FrontOut1-Src-SP=40`: Set the source of the Front Panel Output 1 to Prescaler 0. Values 40-42 correspond to prescalers 0-2. A complete list of values is available in the EVR manual [2].

In order to use different prescaler, simply change the prescaler number, eg. using `PS2-Div-SP` instead of `PS0-Div-SP` sets the divider of prescaler 2 instead of prescaler 0. Similar is for outputs, eg. using `FrontOut0-Src-SP` instead of `FrontOut1-Src-SP` sets the output source signal of front panel output 0 instead of front panel output 1.



## 6 Output a Distributed Bus bit

A custom distributed bus (DBus) bit can be outputted through desired EVR outputs. This tutorial demonstrates how to set up the DBus bit 0 as a source of an EVR front panel output 1, as seen in Figure 4.

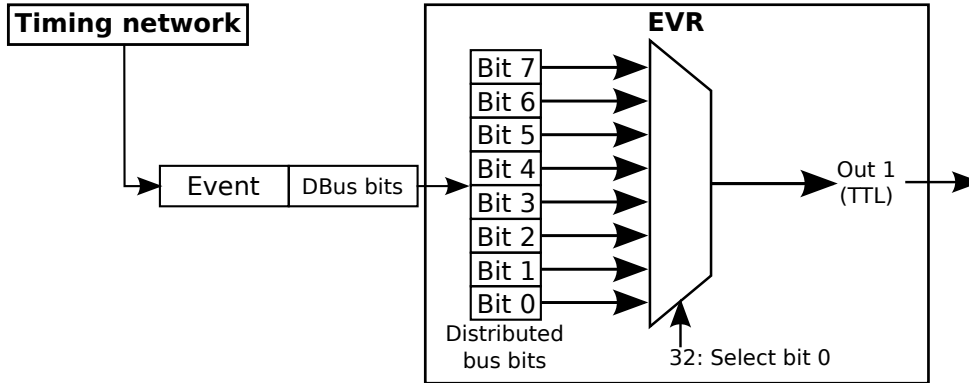


Figure 4: Send DBus bit 0 to the front panel output 1

### 6.1 Instructions:

1. If starting a new IOC application, consult the quick start in Section 2.
2. Set the macro values in the substitution file (`MTEST-VME-EVRTEST/cfg/EVR.subs`) according to this snippet (explained in 6.2):

```
file "$(TEMPLATE_DIR)/evr-vmerf230.template"
{
  {
    ...
    FrontOut1-Src-SP=32,
    ...
  }
}
```

3. Optionally, you can remove all the macros whose values you did not change.
4. Install the prepared IOC by running command `swit -V` from your project folder `MTEST-VME-EVRTEST`.

## 6.2 Substitution snippet explanation:

- **FrontOut1-Src-SP=32**: Set the source of the front panel output 1 to DBus bit 0. Values 32-39 correspond to DBus bits 0-7. A complete list of values is available in the EVR manual [2].

In order to use different front panel output, simply change the front panel output number, eg. using **FrontOut0-Src-SP** instead of **FrontOut1-Src-SP** sets the output source signal of front panel output 0 instead of front panel output 1.

## 7 GUI

There is a caQtDM [1] GUI for the Event Receiver available. It can be used to further configure the EVR or simply check the running configuration. The GUI is launched using the following command:

```
start_EVR.sh -s SYS [options]
```

where the **SYS** represents a **mandatory** system name, and **[options]** are as follows:

```
-r <event receiver name> ..... set the event receiver name
                                (default:EVR0).
-h ..... shows the options and usage
```

**Example 1:** Open the GUI for the event receiver EVR0, using system name MTEST-VME-EVRTEST.

```
start_EVR.sh -s MTEST-VME-EVRTEST
```

**Example 2:** Open the GUI for the event receiver EVR3, using system name MTEST-VME-EVRTEST.

```
start_EVR.sh -s MTEST-VME-EVRTEST -r EVR3
```

**Example 3:** Shows options and usage.

```
start_EVR.sh -h
```

## References

- [1] caQtDM - a medm replacement based on QT. <http://epics.web.psi.ch/software/caqtdm/>.
- [2] Evr manual. [https://github.psi.ch/projects/ED/repos/mrfioc2/browse/documentation/PSI/evr\\_manual.pdf](https://github.psi.ch/projects/ED/repos/mrfioc2/browse/documentation/PSI/evr_manual.pdf).
- [3] PSI. How to set up IFC 1210. <https://controls.web.psi.ch/cgi-bin/twiki/view/Main/HowToSetupIFC1210ioc>.