Universal RPvdsEx file

By Ruurd Lof

Version 2.3

Date: 08/05/2019

# Introduction

In several of our setups the RZ6 multipurpose Digital Signal Processor is used for carrying out trials that consists of several tasks, such as: registering triggers, set multiplexer settings, producing sounds, producing triggers (e.g. for LEDs) and producing event signals for the event recorder. In the current situation the number of tasks and the types of tasks that can be performed within one trial is limited, which leads to having different (ad hoc) solutions (e.g. different RPvdsEx files) for different experiments. The purpose of a new design for the RPvdsEx file is threefold:

1. to be more flexible in combining different tasks.

2. to be less limited in the amount of tasks that can be performed within a single trial.

3. to provide matlab with simple unified interface.

A trial consists of several tasks specified in a \*.cfg file. Before starting a trial, parameters will be uploaded from the computer to the RZ6 by the matlab program. The RZ6 will be able to perform the complete trial without further intervention of the computer.

The new RPvdsEx program will use a matrix for the specification of several tasks that are contained in a trial. Each column of the matrix specifies all the parameters necessary to time and perform a single task. This document describes the different components of the task matrix and specify the parameters and the constants that will be used to describe the tasks.

# Different versions

There are three versions of the RPvdsEx program. A version that has all the basics covered, that can run on a single core RZ6. This version can only play two different WAV files in a single trial. A multicore version that runs on three and four core versions of the RZ6. It can play eight different WAV files and has the possibility to play moving sounds over a linear array of speakers. The third version is the same as the single core version, but has added the possibility to play Ripple sounds.

# Organisation of the Tasks

## Tasks

A task is specified by a column in the matrix.

Every task has the following elements:

1. The type of **task**
2. A delay time
3. The **external outputs** used to signal events
4. Several task dependent parameters

## Task types

The following tasks can be specified:

* Wait for trigger
* Start/Stop playing sound on channel A\*.
* Start/Stop playing sound on channel B\*.
* Start/Stop playing moving sound (over multiple speakers).
* Start/Stop Acquisition by ADC channels 0...7.
* Send a single trigger.
* Send a double trigger.
* Set/Reset digital output bits.
* Set event output byte.

\*Sounds can be on either or both speakers at the same time or moving between two speakers.

Sounds can be generated (Sinewave, Sweep, Noise, Ripple) or uploaded (WAV).

## Trigger task

Triggers are used to start a series of tasks. If the current task specifies that it needs to be triggered, the program waits for that trigger. After receiving the proper trigger, the program reads the next columns and executes the corresponding tasks one by one in the order they are listed in the matrix. The tasks are timed with respect to the trigger at the beginning of the series of tasks. If two consecutive tasks have the same delay time specified, the second task will be executed 2 DSP-clock pulses after the first.

The trigger types can be the following:

* **ZBus A or ZBusB**: a ZBus trigger that is often triggered by the software, but can be triggered by other ZBus equipment.
* **External**: a digital input
* **Software**: software trigger

The trigger types are specified by a byte; therefore, a combination of valid trigger types is possible.

N.B. The **ZBusA** trigger is used exclusively as a reset of the trial (without losing the uploaded matrix data) and can be used any moment to stop a running trial.

## The external I/O

* Triggers can be detected at inputs A0...A7. The used input channels are specified by a byte.
* Triggers can be generated at B0...B7. The used output channels are specified by a byte.
* Events can be generated at B0...B7. The used output channels are specified by a byte.

# Coding of task data

## Introduction

The data that specifies the tasks is sent from Matlab in the form of a (7xn) matrix, where 7 is the number of parameters that specify a task and n is the number of tasks. Each element of the matrix is a 32 bits integers (I32). The first three integers of the task have a fixed meaning. The integers four to seven make up parameters of which the meaning is context dependent.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Task 1 | Task 2 | Task …. | Task n-1 | Task n |
| Row 1 |  |  |  |  |  |
| Row 2 |  |  |  |  |  |
| Row 3 |  |  |  |  |  |
| Row 4 |  |  |  |  |  |
| Row 5 |  |  |  |  |  |
| Row 6 |  |  |  |  |  |
| Row 7 |  |  |  |  |  |

## Row 1: Task types

* 0 = Wait for trigger
* 1 = Start/Stop sound A
* 2 = Start/Stop sound B
* 3 = Set/Reset MUX
* 4 = Set signaling byte (signals the task for eventrecorder)
* 5 = Start/Stop moving sound
* 6 = Start/Stop Data Acquisition
* 7 = Set digital output
* 8 = Sent Single/Double trigger

## Row 2: Sound types

* 0 = Stop sound/None
* 1 = Start Tone
* 2 = Start Sweep
* 3 = Start Noise
* 4 = Start Ripple {only in four core version of RZ6}
* 5 = Start WAV
* 6 = Start B=A\*

\* Only relevant when B is set; the same sound (SND, WAV or Ripple) is played at channel A and channel B

## Row 3: Delay time

* Delay in milliseconds

## Row 4…7: context dependent parameters

The integers of row four to seven make up the context dependent parameters Par1 to Par4. The meaning of the parameters differ dependent on the different task and sound types specified by row one and two.

* Task type = 0: Wait for trigger
  + - Par1 Trigger type (Bit 0 = ZBusB; Bit 1 = External; Bit 2 = Soft1)
    - Par2 External trigger (RZ6 byte A)
* Task type = 1 or 2: Start/Stop sound A or B
  + Sound type = 0: Stop Sound
    - no parameters
  + Sound type = 1: Start Sound Tone
    - Par1 frequency
    - Par4 attenuation
  + Sound type = 2: Start Sound Sweep
    - Par1 start frequency
    - Par2 number of octaves
    - Par3 sweeps per second
    - Par4 attenuation
  + Sound type = 3: Start Sound Noise
    - Par1 LPFreq
    - Par2 HPFreq
    - Par4 attenuation
  + Sound type = 4: Start Sound Ripple
    - par1 Start frequency
    - par2 Modulation in time (Hz)
    - par3 Modulation in frequency (Phase/Octave)
    - Par4 attenuation
  + Sound type = 5: Start Sound WAV
    - Par1 WAV index number (up to 8 WAVs can be stored)
    - Par4 attenuation
  + Sound type = 6: Start Sound B=A
    - Par1 MOV type (0 = no move; 1 = sinus move; 2 = linear move);
    - Par2 MOV frequency (in mHz)
    - Par3 Starting Phase (-180 … 180) Left = -90; Middle = 0; Right = 90
    - Par4 attenuation
* Task type = 3: Set/Reset MUX
  + - Par1 MUX byte (RZ6 byte C)
* Task type = 4: Set signaling byte (signal to event recorder)
  + - Par1 Signaling byte
* Task type = 5: Start/Stop Moving sounds
  + - Par1: Start/Stop
    - Par2: The number of speakers currently used in the moving sound array
    - Par3: period of the movement measured in milliseconds
    - Par4: starting phase (-180 … 180) Left = -90; Middle = 0; Right = 90
* Task type = 6: Start/Stop Data acquisition
  + - Par1: Start/Stop
    - Par2: ACQ time
    - Par3: Analog input selection byte
* Task type = 7: Set digital out
  + - Par1: output byte (RZ6 byte B)
* Task type = 8: Output trigger
  + - Par1: output trigger byte (RZ6 byte B)
    - Par2: double trigger interval in microseconds (0 = single trigger; >100 = double trigger)

## RZ6 Digital-I/O

The RZ6 Digital-IO consists of three bytes, A, B and C. Byte A is used for digital input. Bits A0…A3 are connected to the buttons of the RBOXRZ6. Bits A4…A7 are connected to BNC sockets on the RZ6\_Digital\_I/O patch panel. Byte B is used for digital output. Bits B0…B3 are connected to the LEDs of the RBOXRZ6. Bits B4…B7 are again connected to BNC sockets on the RZ6\_Digital\_I/O patch panel. Byte C is dedicated to setting the configuration of up to four PM2R multiplexers.

#### Inputs (RZ6 byte A)

* Bit 0 = A0 (RBOXRZ6 Button)
* Bit 1 = A1 (RBOXRZ6 Button)
* Bit 2 = A2 (RBOXRZ6 Button)
* Bit 3 = A3 (RBOXRZ6 Button)
* Bit 4 = A4 (PP RZ6 Digital-I/O BNC)
* Bit 5 = A5 (PP RZ6 Digital-I/O BNC)
* Bit 6 = A6 (PP RZ6 Digital-I/O BNC)
* Bit 7 = A7 (PP RZ6 Digital-I/O BNC)

#### Outputs (RZ6 byte B)

* Bit 0 = B0 (RBOXRZ6 LED)
* Bit 1 = B1 (RBOXRZ6 LED)
* Bit 2 = B2 (RBOXRZ6 LED)
* Bit 3 = B3 (RBOXRZ6 LED)
* Bit 4 = B4 (PP RZ6 Digital-I/O BNC)
* Bit 5 = B5 (PP RZ6 Digital-I/O BNC)
* Bit 6 = B6 (PP RZ6 Digital-I/O BNC)
* Bit 7 = B7 (PP RZ6 Digital-I/O BNC)

#### MUX (RZ6 byte C)

* Bit 0-3 = Channel Nr.
* Bit 4-5 = Device Nr.
* Bit 6 = Set
* Bit 7 = Reset

## RZ6 Analog inputs

The analog input channels that used in the data acquisition tasks is controlled by eight bits, so that the channels can be activated and stopped individually. All versions of the RZ6 have two analog inputs A and B that correspond to bit0 and bit1. These are available by two BNC connectors on the front panel of the RZ6 (“In-A” and “In-B”). The multicore version of the RZ6 can be connected optically to an RA8GA Adjustable Gain Preamp with an additional set of analog inputs. Six of these analog inputs are available for data-acquisition which are controlled by bit2 to bit7. In case that magnetic fields are used for the detection of head movements, bit2, bit3 and bit4 correspond to the H, V and F fields.

#### RZ6 Analog input selection byte

* Bit 0 = Channel A
* Bit 1 = Channel B
* Bit 2 = Channel 1 (or channel H)
* Bit 3 = Channel 2 (or channel V)
* Bit 4 = Channel 3 (or channel F)
* Bit 5 = Channel 4
* Bit 6 = Channel 5
* Bit 7 = Channel 6