XMC4000

XMC4500 Family

Event Request Unit - ERU

ERU Basics

User Guide

Device Guide

<Revision 1.0>, <2012-03-31>

Microcontroller

Edition <2012-03-31>

Published by
Infineon Technologies AG
81726 Munich, Germany

Revision History	
Page or Item	**Subjects (major changes since previous revision)**
<Revision 1.0>, <2012-03-31>	

Table of Contents

List of Figures

1 ERU Basics

The Event Request Unit (ERU) gives the user the opportunity to Top-Level control interactions between moduls by tailor made event schemes. This benefits e.g. motor control by reducing SW in timer-ADC-IO interactions. The ERU enables realization of time critical interconnections that require total real-time correctness and safety.

1.1 ERU Use Cases

Now, for example, if an application requires ADC conversions to start on timer events under specific conditions, i.e. if not directly via some default signal path implementation, then an ERU is able to offer an alternative signal path. It may involve dependence on a port pin, time window due to a second timer – or a certain event pattern.

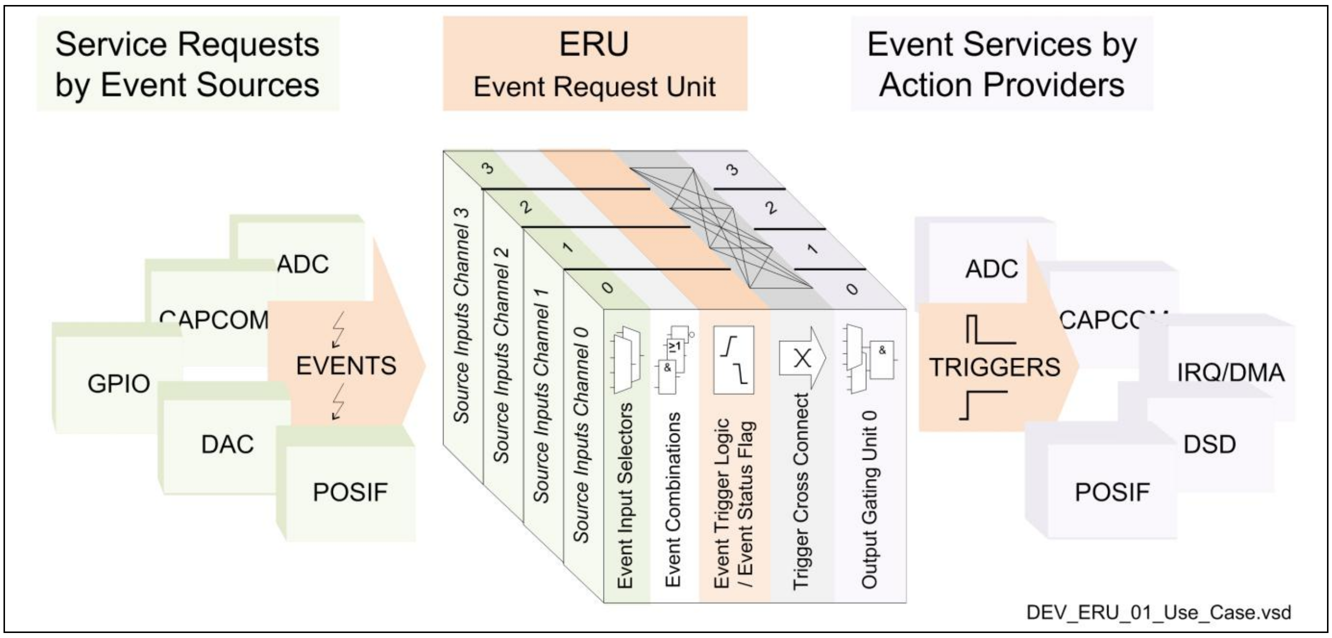

Figure 1 Use Cases for an Event Request Unit (ERU)

1.2 The ERU Concept

The Event Request Unit (ERU) is a modular logic that can select, combine, detect and memorize peripheral events, generate trigger pulses for distribution back to system on gating conditions, as requests for HW actions. The ERU can select input signals from totally 32 event sources and request for actions via four output channels.

1.3 Top Level Control of Event Requests

The ERU top-level control of events enables creation of comprehensive embedded tasks in HW, which may concatenate actions from different units like a statemashine. This gives interactions on low level, i.e. without SW involved - just controlled by an event flow and an action scheme that is configurated on a top-level by the user.

2 ERU Implementations

There are two implementations, ERU0 and ERU1.

See Figure 2.

- ERU0 selects External Event Requests from mainly General Purpose IOs (GPIO).
- ERU1 selects Internal Event Requests from embedded peripherals (PERIPH) and General Purpose IOs too.

The ERU0 links trigger pulses (named TRIGGER) to the interrupt system only, whilst the ERU1 also may link the trigger pulses to the peripherals - and distribute *Event Pattern Match Detect* status signals (named LEVEL).

See Figure 2.

The picture shows how units are interfaced to the ERUs and may interact via a **Top-Level Interconnect Matrix**. To complete the picture of capable interaction scenarios there is also shown how operations may be extended to involve DMA transfers (by the **GPDMA**), triggered by a handler (**DLR**) on the **Service Request Lines (SRn)**.

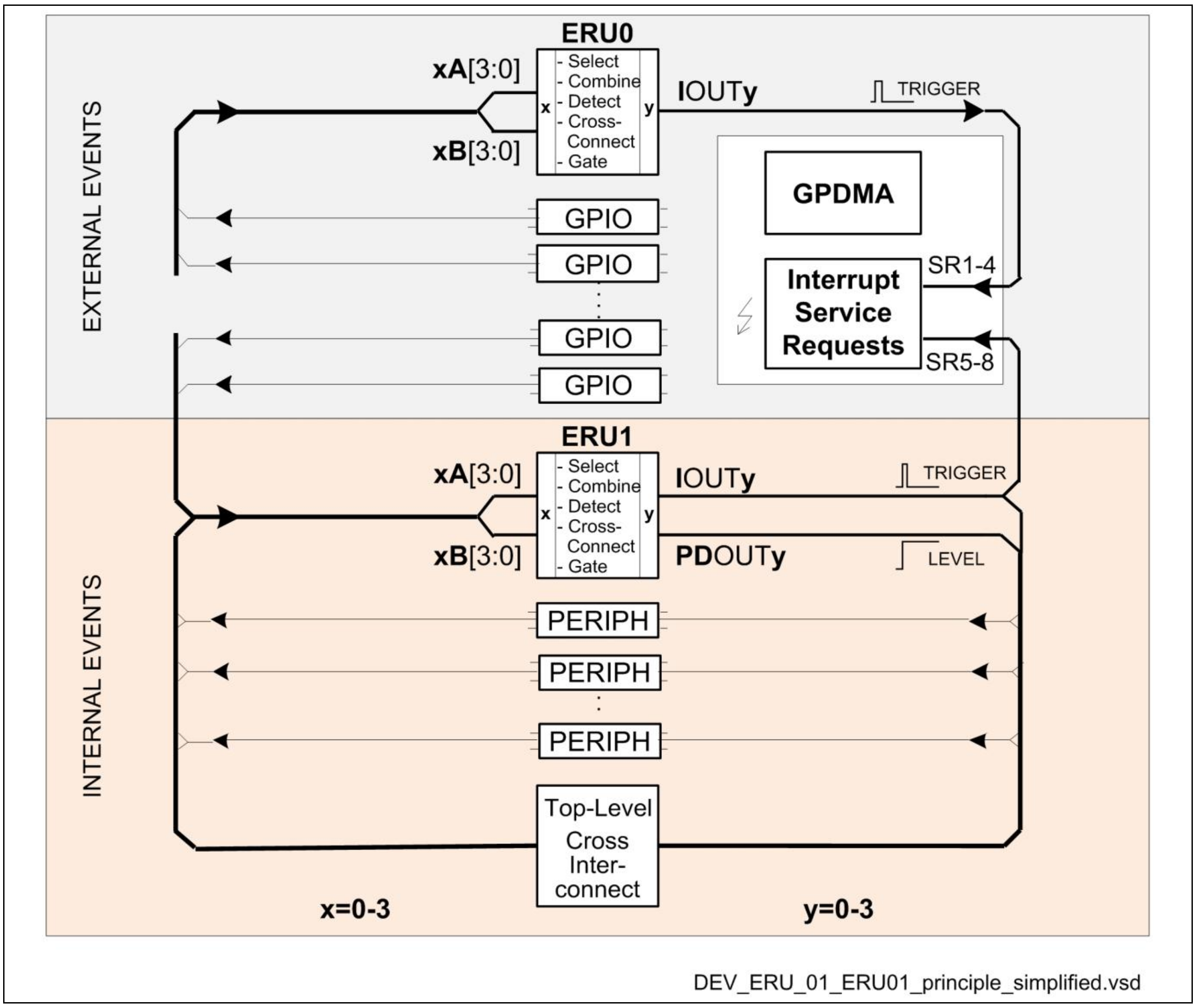

Figure 2 Signal Flow Principle with Event Request Units

2.1 Event Request Selection

Each input line, to xA(3:0) or xB(3:0) of an ERU input channel x (x=0-3), is hard wired to a specific event source. So, a specific Event Request, from an I/O pin or a peripheral unit, is selected by a MUX switch at either the xA(3:0) or at the xB(3:0) input line group of the ERU0 or ERU1, according to the **ERU Pin Connections** tables.

2.2 Signal Combination Logic

An Event Request signal pair (A,B), selected by the input line groups xA(3:0) and xB(3:0) respectively, may represent a considered compound event. Such a conditional signal can be created by the Combination Logic of the channel x input stage - before it is transfered to the Event Trigger Logic stage for an event edge detection.

2.3 Input and Trigger functions

A signal edge is regarded as a true *event edge* if it complies with the *considered edge*, positive or negative – and if so, then an event flag FLx is set, a trigger pulse TRx will be created and, if enabled, passed to the Cross Connect Select stage - and so will the event flag level status. The flag may be reset by a false event or by SW.

2.4 Cross Connect Selection

The Cross Connect Select stage, for each channel x of an ERU, is an Event Trigger pulse routing matrix that is able to switch the trigger pulse distribution path back to the system via any output channel y (y=0-3), targeting a certain event service action provider - according to the **Top-Level control** by the **ERU Pin Connections** tables.

2.5 Output Gating Selection

The Output Gating Selection stage of an ERU may gate the Trigger Pulse signals (TRIGGER) in the output channels IOUTy (y=0-3) by the *Event Pattern Match/Mismatch Detect* status (LEVEL) output signal, PDOUTy (y=0-3). An Event Pattern is a combination of memorized x-channel events stored in the event flags FLx (x=0-3).

2.5.1 Output Gating by Designated Peripheral Trigger Inputs

The ERU1 has designated trigger inputs from the ADC and from some of the CAPCOM4/-8 units that may be be linked *directly* to the Gated Trigger stage of an Output Gating Unit channel OGUy. By this “kitchen entrance” like concept these **Peripheral Triggers** act immediately, unconditionally in the output gating by the event flags FLx.

3 Top-Level Interconnect Matrix

Most interconnect lines go directly from one system unit to another unit, without passing an ERU – i.e. an ADC conversion may be started directly by a signal from a timer. These immediate interconnections are described virtuelly by the so called **Top-Level Interconnect** matrix – a table which is embodied by a block in the drawings.

See Figure 3.

3.1.1 The ERU Pin Connection Tables

It should be mentioned and understood though that the **ERU Pin Connections** tables (which describe the ERU0 and ERU1 interfaces to peripheral units or IO:s) are actually subsets of the total system **Top-Level Interconnect** matrix – split up like this in order to ease understanding of those paths where an ERU is involved!

See Figure 4.

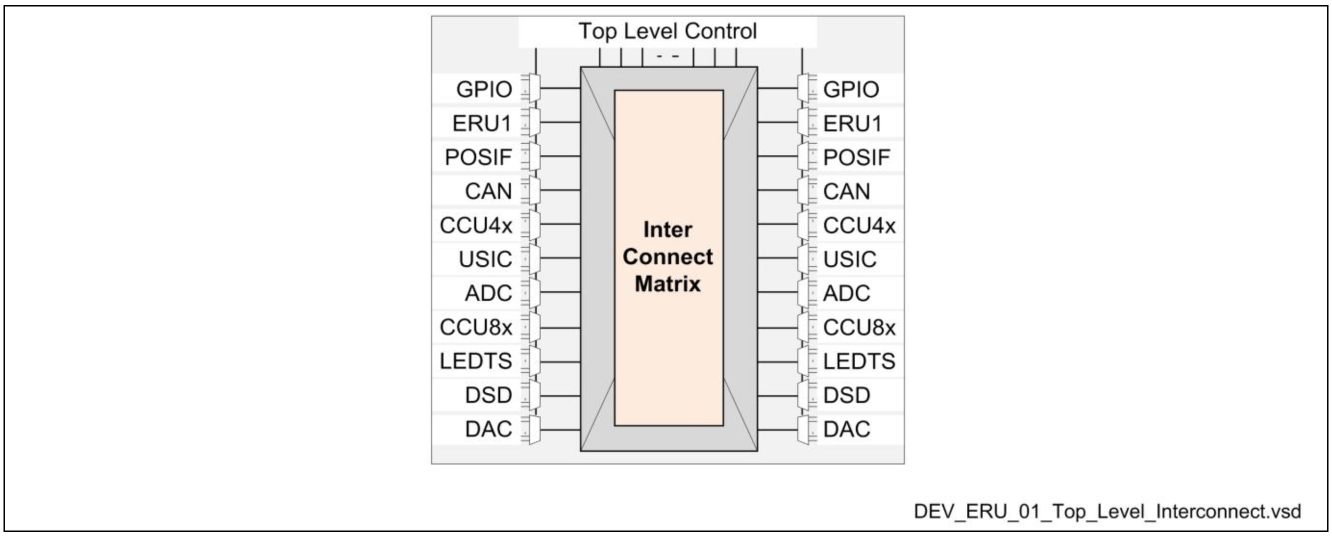

Figure 3 The Top-Level Interconnect Matrix

3.2 The Principle Blocks of the ERU0 vs. the ERU1 in Detail

See Figure 4.

The principle difference between the ERU0 and the ERU1 implementations are which event request sources they are connected to as service providers and which action providers they are linked to as service requestors: The ERU0 for *external* events - and the ERU1 for *internal* events *plus* 3 Peripheral Trigger “*kitchen entrances*”.

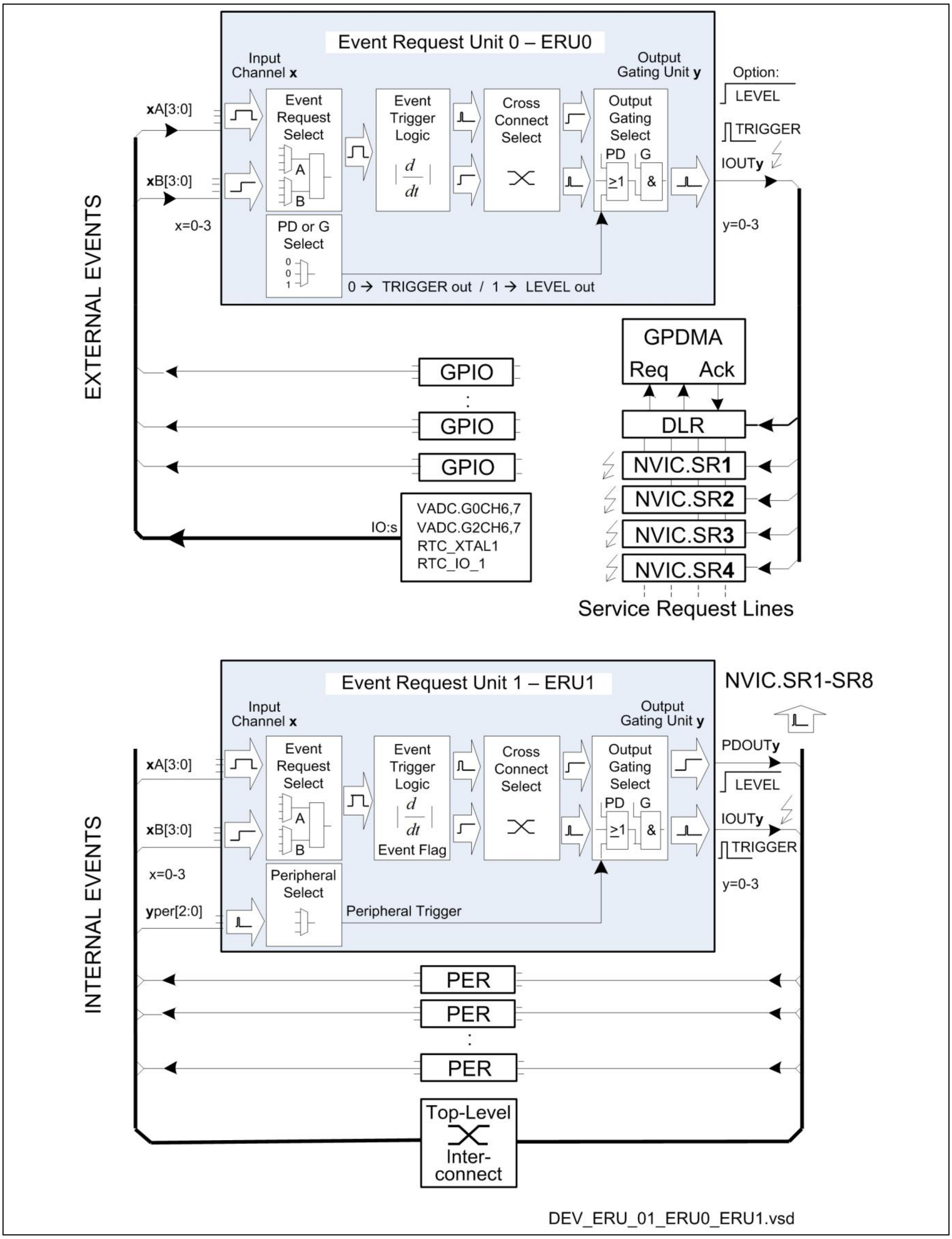

Figure 4 Event Request Unit Overview (ERU0 vs ERU1)

4 Getting Started with Event Request Unit - ERU

There are simply just one Input Selection Register plus two Control Registers per input channel **x** (x=0-3) to setup the entire functionality of an ERU on a Top Level and channel wise target the Output Channels y (y=0-3):

• **EXISEL**	Input Selection Register
• **EXICONx**	Input and Trigger Control Register
• **EXOCONx**	Output Control Register

Initializations and functions are determined by writing to the respective register **bitfields**.

See Figure 5.

4.1 Initialization Example

```
// Set up ERU1 Input Channel x=0, targeting the Output Channel y=2 (IOUT2):

// Select Event Request for input A and B in the ERU1_EXISEL register bitfields:
EXS0A = 0; // select input ERU1_0A0 – i.e. A connected to GPIO port pin P1.5
EXS0B = 1; // select input ERU1_0B1 – i.e. B connected to CCU80.ST0 status bit 0

// Select logical combinations that should be taken into account as event request
NA = 0; // input A is used directly, i.e. not inverted
NB = 0; // input A is used directly, i.e. not inverted
SS = 3; // setup source combination, i.e. logical condition: input A AND input B

// Select Event Trigger Logic conditions for trigger, level detect and event edge
PE = 1; // set trigger pulse enable, i.e. an output trigger pulse will be created
LD = 0; // define event flag sticky, i.e. it will not be rebuild by HW, but by SW
RE = 1; // detection on Rising Edge, i.e. event edge on positive signal transition
FE = 0; // no detection on Falling Edge

// Perform Cross Connect Selection, i.e. select target output channel y (here y=2)
// for the Event Trigger Logic Output Pulse TR0 from input channel x (here x=0):
OCS = 2; // the channel output (IOUT2) would trigger the following destinations:
         // CCU4x.IN2(K)      - i.e. CAPCOM4 Unit CCU4x, Slice CC42 Timer Input
         // CCU8x.IN2(G)      - i.e. CAPCOM8 Unit CCU8x, Slice CC42 Timer Input
         // VADC.G2REQTRN     - i.e. ADC Trigger Request N Input, Group 2
         // VADC.G3REQTRN     - i.e. ADC Trigger Request N Input, Group 3
         // ERU1.1B3          - i.e. A trigger feed-back to ERU1 Channel Input 1B3
         // NVIC.SR7          - i.e. A trigger puls to Service Request Line nr 7
         // POSIF0.MSET(F)    - i.e. POSIF 0 Multi Channel Next Pattern Update Set
         // POSIF1.MSET(F)    - i.e. POSIF 1 Multi Channel Next Pattern Update Set

// Select Event Output Trigger Control 2 Register ERU1_EXOCON2 for output gating
// Select Output Gating functions – Here: just a straight forward alternative:
ISS  = 0; // set the Internal Peripheral Trigger Source Selection =0, "no source"
GEEN = 0; // disable "Gating Event Enable on Pattern Detection Changes" trigger
GP   = 1; // set "Output Gating Select on Pattern Detection" =1, to activate IOUT2
IPEN0, IPEN1, IPEN2, IPEN3 = 0; // disable the Pattern Detection Enable flags
```

4.2 Event Request Unit Block Diagrams and Control by Registers

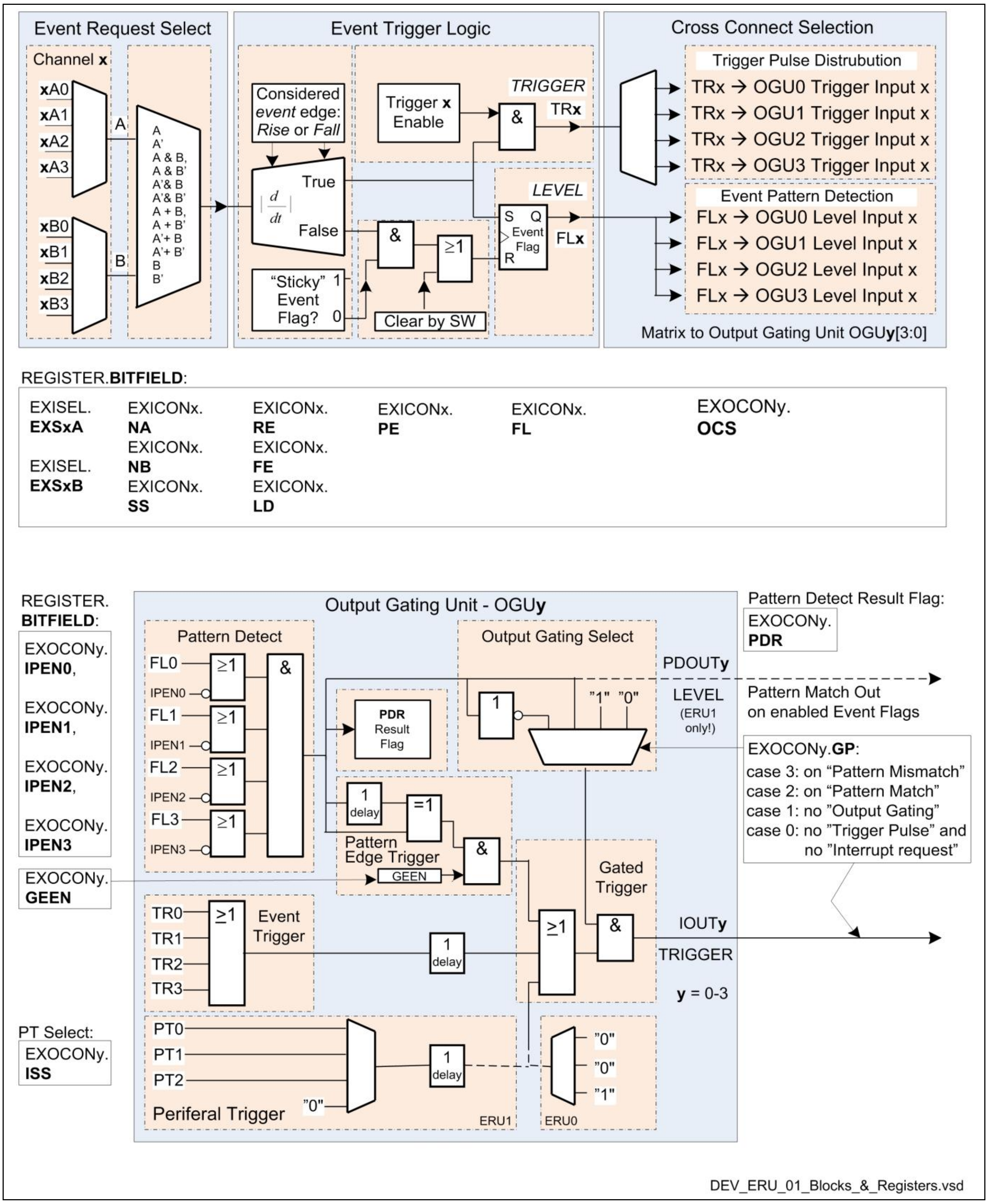

Figure 5 Event Request Unit Block Diagrams and Control by Registers