XMC4000

XMC4500 Family

CCU4 and Cross Interconnections

Top-Level Interconnect Control

User Guide

Device Guide

<Revision 1.0>, <2012-30-03>

Microcontroller

Edition <2012-30-03>

Published by
Infineon Technologies AG
81726 Munich, Germany

© 2012 Infineon Technologies AG
All Rights Reserved.

Legal Disclaimer

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics. With respect to any examples or hints given herein, any typical values stated herein and/or any information regarding the application of the device, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation, warranties of non-infringement of intellectual property rights of any third party.

Information

For further information on technology, delivery terms and conditions and prices, please contact the nearest Infineon Technologies Office (**www.infineon.com**).

Warnings

Due to technical requirements, components may contain dangerous substances. For information on the types in question, please contact the nearest Infineon Technologies Office.

Infineon Technologies components may be used in life-support devices or systems only with the express written approval of Infineon Technologies, if a failure of such components can reasonably be expected to cause the failure of that life-support device or system or to affect the safety or effectiveness of that device or system. Life support devices or systems are intended to be implanted in the human body or to support and/or maintain and sustain and/or protect human life. If they fail, it is reasonable to assume that the health of the user or other persons may be endangered.

Confidential

Revision History	
Page or Item	**Subjects (major changes since previous revision)**
<Revision 1.0>, <2012-30-03>	

Trademarks of Infineon Technologies AG

AURIX™, C166™, CanPAK™, CIPOS™, CIPURSE™, EconoPACK™, CoolMOS™, CoolSET™, CORECONTROL™, CROSSAVE™, DAVE™, EasyPIM™, EconoBRIDGE™, EconoDUAL™, EconoPIM™, EiceDRIVER™, eupec™, FCOS™, HITFET™, HybridPACK™, I²RF™, ISOFACE™, IsoPACK™, MIPAQ™, ModSTACK™, my-d™, NovalithIC™, OptiMOS™, ORIGA™, PRIMARION™, PrimePACK™, PrimeSTACK™, PRO-SIL™, PROFET™, RASIC™, ReverSave™, SatRIC™, SIEGET™, SINDRION™, SIPMOS™, SmartLEWIS™, SOLID FLASH™, TEMPFET™, thinQ!™, TRENCHSTOP™, TriCore™.

Other Trademarks

Advance Design System™ (ADS) of Agilent Technologies, AMBA™, ARM™, MULTI-ICE™, KEIL™, PRIMECELL™, REALVIEW™, THUMB™, µVision™ of ARM Limited, UK. AUTOSAR™ is licensed by AUTOSAR development partnership. Bluetooth™ of Bluetooth SIG Inc. CAT-iq™ of DECT Forum. COLOSSUS™, FirstGPS™ of Trimble Navigation Ltd. EMV™ of EMVCo, LLC (Visa Holdings Inc.). EPCOS™ of Epcos AG. FLEXGO™ of Microsoft Corporation. FlexRay™ is licensed by FlexRay Consortium. HYPERTERMINAL™ of Hilgraeve Incorporated. IEC™ of Commission Electrotechnique Internationale. IrDA™ of Infrared Data Association Corporation. ISO™ of INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. MATLAB™ of MathWorks, Inc. MAXIM™ of Maxim Integrated Products, Inc. MICROTEC™, NUCLEUS™ of Mentor Graphics Corporation. Mifare™ of NXP. MIPI™ of MIPI Alliance, Inc. MIPS™ of MIPS Technologies, Inc., USA. muRata™ of MURATA MANUFACTURING CO., MICROWAVE OFFICE™ (MWO) of Applied Wave Research Inc., OmniVision™ of OmniVision Technologies, Inc. Openwave™ Openwave Systems Inc. RED HAT™ Red Hat, Inc. RFMD™ RF Micro Devices, Inc. SIRIUS™ of Sirius Satellite Radio Inc. SOLARIS™ of Sun Microsystems, Inc. SPANSION™ of Spansion LLC Ltd. Symbian™ of Symbian Software Limited. TAIYO YUDEN™ of Taiyo Yuden Co. TEAKLITE™ of CEVA, Inc. TEKTRONIX™ of Tektronix Inc. TOKO™ of TOKO KABUSHIKI KAISHA TA. UNIX™ of X/Open Company Limited. VERILOG™, PALLADIUM™ of Cadence Design Systems, Inc. VLYNQ™ of Texas Instruments Incorporated. VXWORKS™, WIND RIVER™ of WIND RIVER SYSTEMS, INC. ZETEX™ of Diodes Zetex Limited.

Last Trademarks Update 2011-02-24

Table of Contents

1 Top-Level Interconnect Control 5

2 Top-Level Control of Event Requests to/from a Timer Slice 6
2.1 CCU4 Using Top-Level Interconnect Control with Event Request Unit (ERU) 6
2.2 CCU4 and ERU Use Cases 6

3 Getting Started with CCU4 and Cross Interconnections 8
3.1 Getting Started with the CAPCOM4 Unit CCU4x 8
3.1.1 Using the CAPCOM4 External Events Control Setup Registers 8
3.1.2 CAPCOM4 External Event Control Register Bitfields 8
3.2 Getting Started with Event Request Unit – ERU1 8
3.3 Cross Interconnection Example – Using CCU4, ERU1, ADC and GPIO 8
3.3.1 ADC Request on Port Pin P2.1 State AND the Timer CCU40CC40 Status Bit 8
3.3.2 Event Request Unit Block Diagrams and Control Registers 10

4 Runtime Handling of CCU4 and Cross Interconnections 11
4.1 Handling Event Concatenation using ERU1 and Pattern Match Detection 11

List of Figures

Figure 1 The Top-Level Interconnect matrix 5
Figure 2 Example with an ADC cross connected to a Timer via the Top-Level Interconnect matrix 6
Figure 3 Top-Level Interconnect with conditional event request control by the Event Request Unit 6
Figure 4 Details of a CCU4 Timer Slice system interactions by Top-Level Interconnect Control 7
Figure 5 Event Request Unit Block Diagrams and Control Registers 10
Figure 6 Concatenation of events with Time Window for a press-button Debounce Rejection Filter 11

1 Top-Level Interconnect Control

The comprehensive matrix of interconnect lines enables signals to get directly from one system unit to another – e.g. an ADC conversion may be started directly by a signal from a timer. Such immediate interconnections are described by the so called **Top-Level Interconnect** matrix – a table that is embodied by a block shown here.

See Figure 1.

See Figure 2.

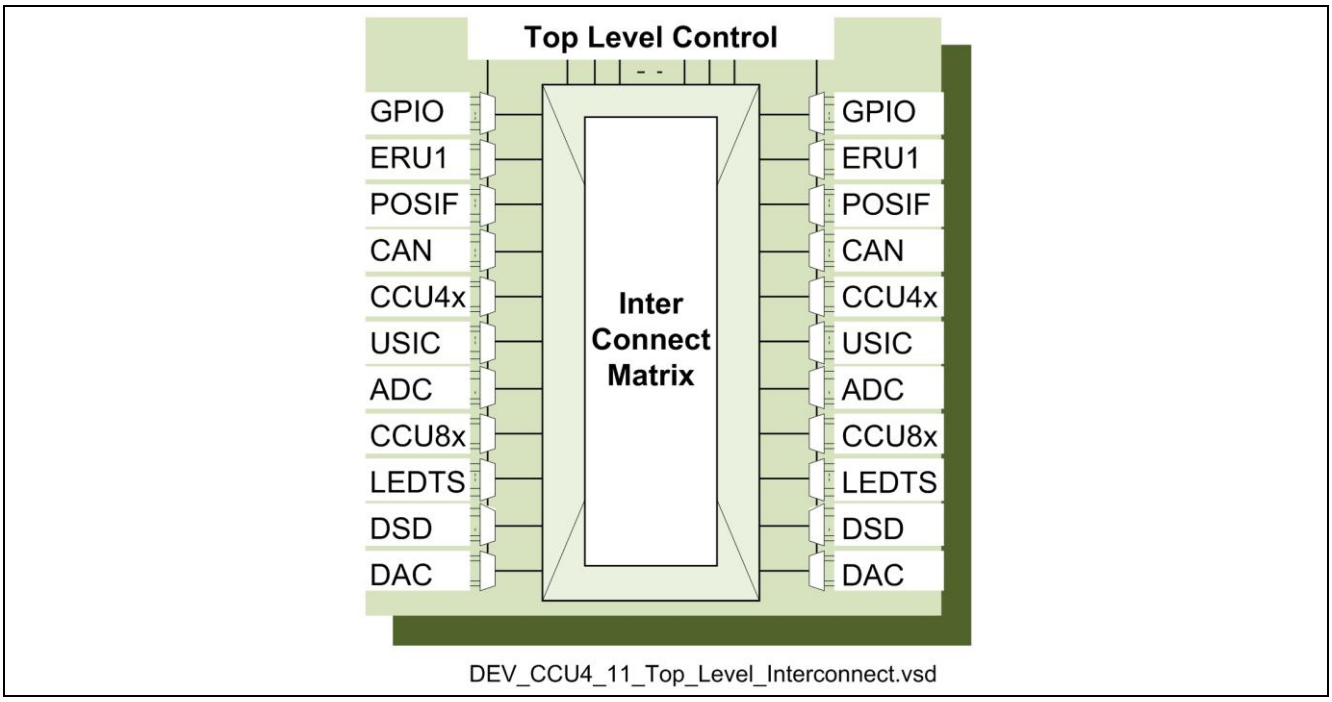

Figure 1 The Top-Level Interconnect matrix

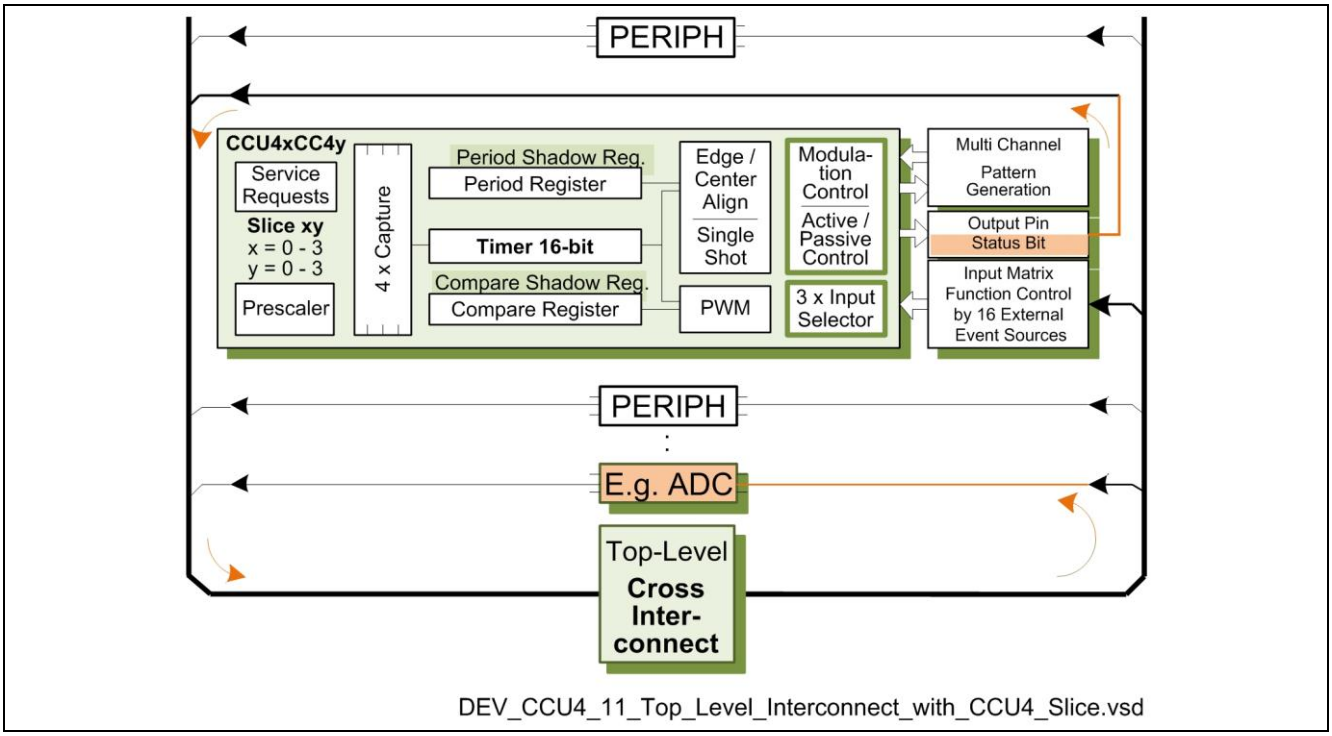

Figure 2 Example with an ADC cross connected to a Timer via the Top-Level Interconnect matrix

2 Top-Level Control of Event Requests to/from a Timer Slice

Top-Level Control of Event Requests is a *flexible control and crossunit distribution* of event signals to/from a slice. The Event Request Unit (**ERU1**) may combine events via the **Top-Level Interconnect** matrix, whereby a crossunit event flow can be top-level controlled without SW according to a request-to-action event pattern setup.

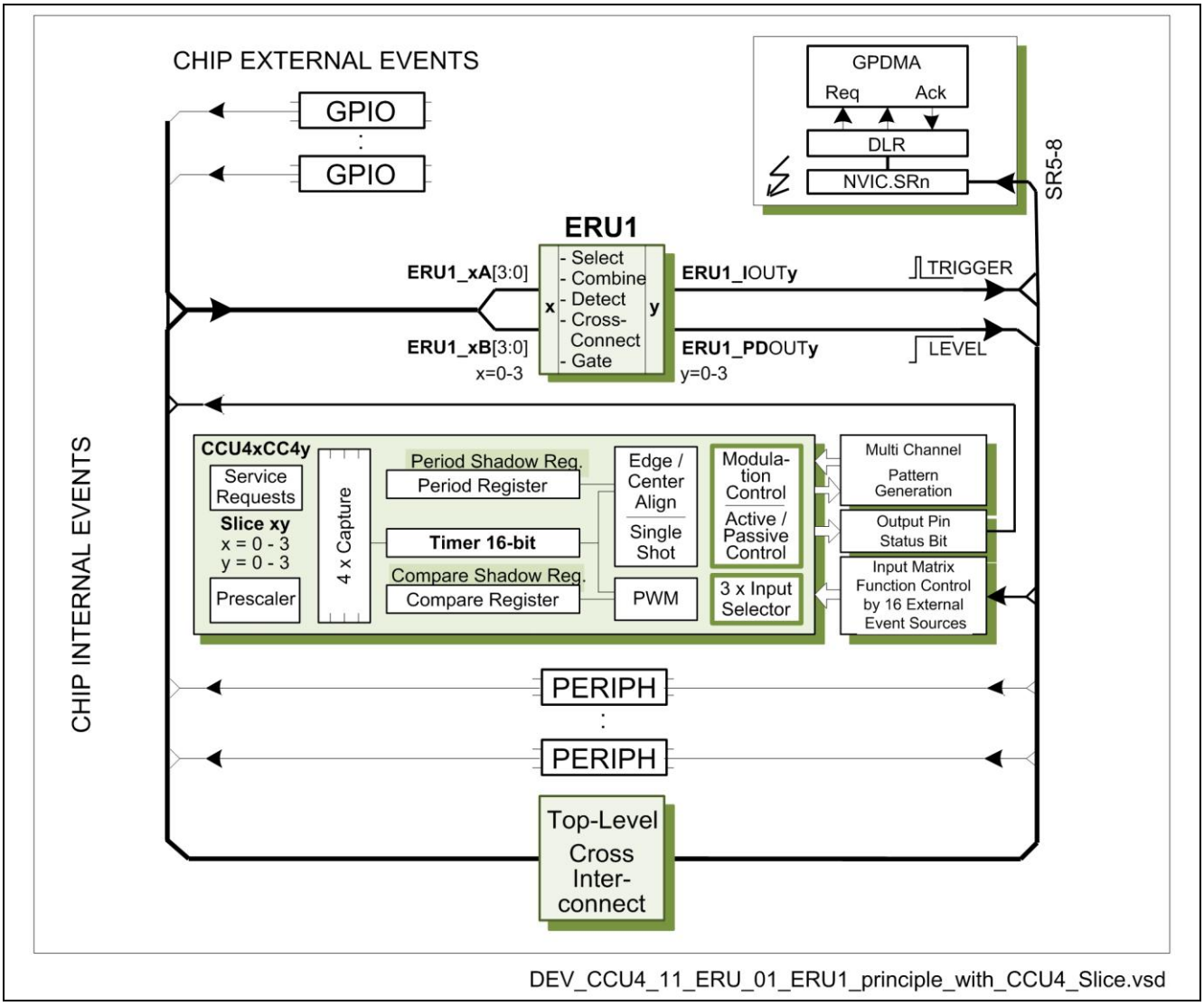

Figure 3 Top-Level Interconnect with conditional event request control by the Event Request Unit

2.1 CCU4 Using Top-Level Interconnect Control with Event Request Unit (ERU)

A slice interface to the **ERU1** and to the **Top-Level Interconnect** matrix can be shown by a simplified scheme. To complete the picture of capable interaction, this scheme also shows how operations can be extended to involve DMA transfers (by the **GPDMA**), triggered by a handler (**DLR**) on the **Service Request Lines (SRn)**.

See Figure 3

2.2 CCU4 and ERU Use Cases

If an application requires ADC conversions to start on timer events under specific conditions, i.e. if not directly via a **Top-Level Interconnect** matrix path, then the ERU1 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 sequence of event patterns.

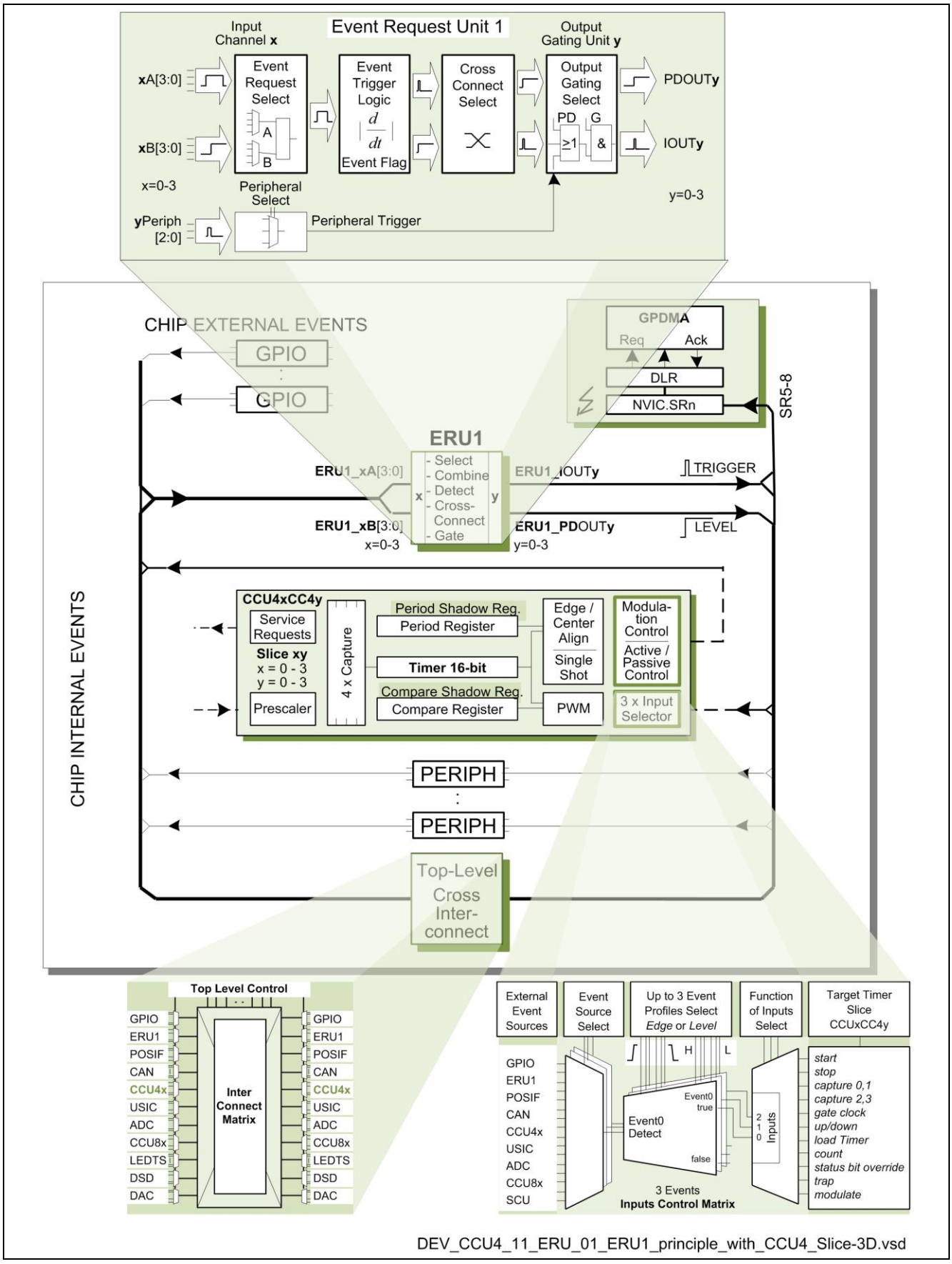

Figure 4 Details of a CCU4 Timer Slice system interactions by Top-Level Interconnect Control

3 Getting Started with CCU4 and Cross Interconnections

3.1 Getting Started with the CAPCOM4 Unit CCU4x

In the following examples the **ERU1** and **CAPCOM4 Ultimate Initialization Sequence** is assumed before start!

3.1.1 Using the CAPCOM4 External Events Control Setup Registers

The *Input Selection* register **CC4yINS** maps *External-Event-Sources* and *Edge/Level Mode* conditions to the *Event[2:0] Input Lines*. In turn the *Connection Matrix Control* register **CC4yCMC** maps the three *Event[2:0] Input Lines* to any of the 11 *Input Functions*. The *Timer Control* register **CC4yTC** enables different *Mode Alternatives*.

3.1.2 CAPCOM4 External Event Control Register Bitfields

1. **CC4yCMC.**<function> Input *function* selection and Event Number (x=0-2) Input Line Selection bitfield
2. **CC4yINS.**<Event-*x* bitfields> Event-x *source* EVxIS (**CC4yINS**[P:A]), *mode* EVxEM or EVxLM bitfields
3. **CC4yTC.**<Mode bitfields> Timer Mode selections e.g. Modes/*Extended modes, Counting Schemes etc*

Note: When using the index 'x' in event nomenclature, it should be understood as 'Actual Event[2:0] Number '.

Note: When using the index 'x' in CCU4x nomenclature, it should be understood as 'Actual CCU4 Unit Number'.

Note: The Ultimate Event Number 'x' for the Input Function is x == <function>-1 (when <function> != 0). <function> == 0 means "disabled".

3.2 Getting Started with Event Request Unit – ERU1

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.

3.3 Cross Interconnection Example – Using CCU4, ERU1, ADC and GPIO

3.3.1 ADC Request on Port Pin P2.1 State AND the Timer CCU40CC40 Status Bit

See Figure 5.

Setup ERU1 Input Channel x=0, targeting the Output Channel y=2 (IOUT2), from which a link to the ADC Trigger Request N Input, Group 2 or 3 can be routed – as follows:

```
// Select Event Request for input A and B in the ERU1_EXISEL register bitfields:
EXS0A = 2; // select input ERU1_0A2 – i.e. B connected to CCU40.ST0 status bit 0
EXS0B = 0; // select input ERU1_0B0 – i.e. B connected to a GPIO port pin P2.1

// 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):
// Target register for the following mapping is EXICONx, i.e. here ERU1_EXICON0:
        // 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:
// Target register for the following mapping is EX0CONy, i.e. here ERU1_EXOCON2:
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
```

3.3.2 Event Request Unit Block Diagrams and Control Registers

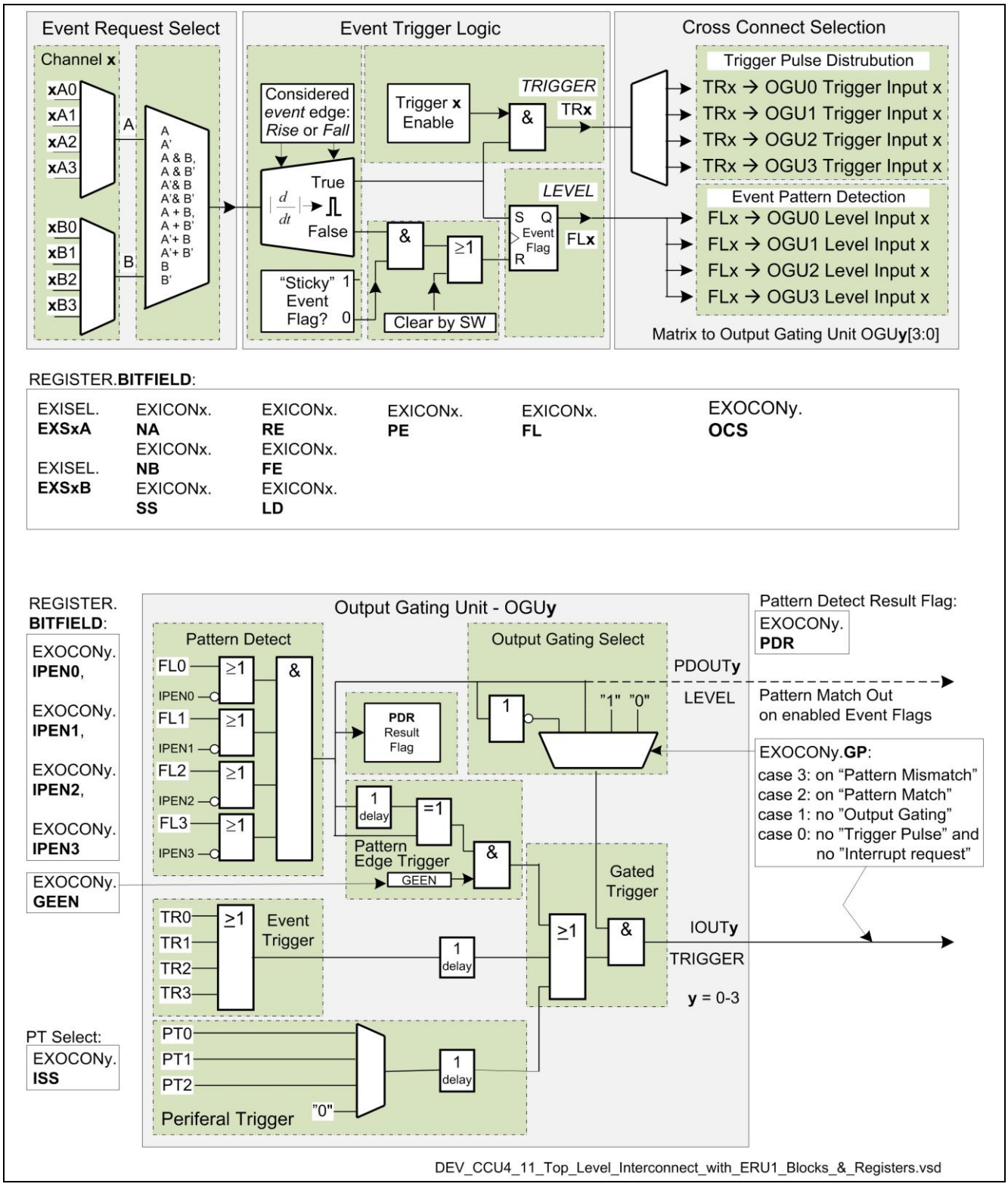

Figure 5 Event Request Unit Block Diagrams and Control Registers

4 Runtime Handling of CCU4 and Cross Interconnections

4.1 Handling Event Concatenation using ERU1 and Pattern Match Detection

In the following example the intention is to show a statemashine-like configuration setup of the ERU1. It will be used to explain how a certain sequence of events can be concatenated on a Top_Level control and detected by an **Event Pattern Match** on the ERU output gating stage. The scenario is e.g. a Debounce Rejection Filter.

Note: No runtime SW involved, so far …

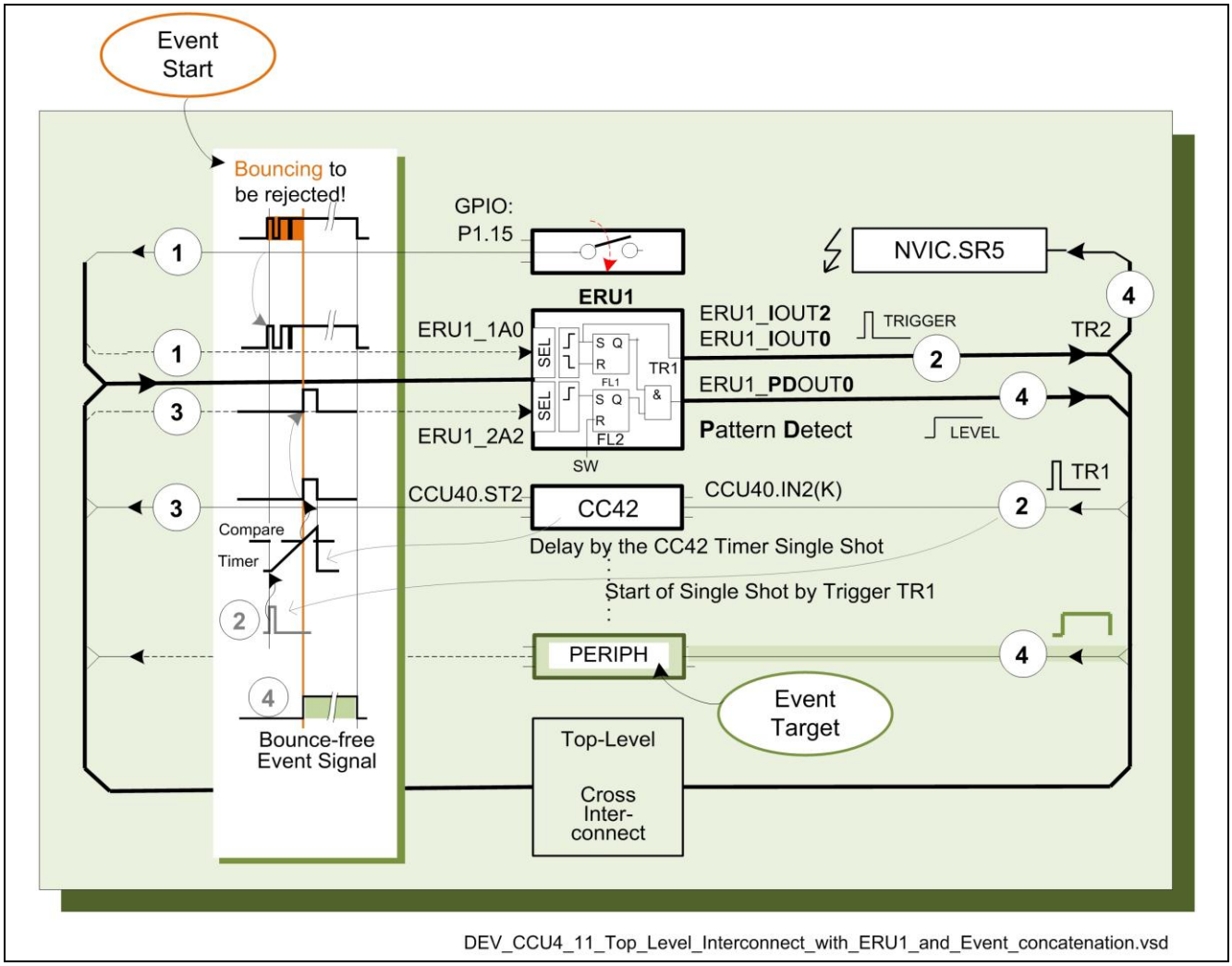

Figure 6 Concatenation of events with Time Window for a press-button Debounce Rejection Filter

1. Bouncing signal from SWITCH-ON. This signal is connected to the ERU1 Channel 1 Input Selection Pin A0 (ERU1_1A0). The event will be detected on Rising Edge, memorized in event flag FL1 and generate a trigger pulse TR1 on the Output Gating Unit 2 (IOUT2) after the ERU1 internal Cross Connect Stage.

2. Trigger TR1 on Output Gating Unit 2 (IOUT2) starts the Timer CCU40T42 in Single Shot mode by an External Event Control command (including the Extended Mode condition for *Flush/Start*).

3. On the single shot timer Compare Event that starts the Single Shot time frame, the CCU40ST2 status bit ST2 will be set. This signal is connected to the ERU1 Channel 2 Input Selection Pin A2 (ERU1_2A2) and will set the sticky flag FL2 and send a trigger pulse TR2 on IOUT0 after the Cross Connect Stage.

4. On the Output Gating Unit 0 (IOUT0) pin the trigger pulse TR2 can be distributed to the Service Request Line nr 5. The PDOUT0 output pin of the Output Gating Unit 0 there will be a Pattern Match

Detection by the AND-Gate and the flags FL1 and FL2. This is the debounced press-button switch signal - extracted by event concatenation – and is sent to the Peripheral Unit named the “Event Target”.

w w w . i n f i n e o n . c o m

Published by Infineon Technologies AG