XMC4000

XMC4500 Family

CCU4 and Compare Mode Features

Output Pattern Generation with CCU4

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 Output Pattern Generation with CCU4 5
1.1 The Principle Compare Blocks 5
1.2 PWM Range 0 – 100% in Up-Count Mode 5
1.3 PWM Range 0 – 100% in Down-Count Mode 6
1.4 PWM Range 0 – 100% in Center Aligned Mode 6

2 Compare Counting in Detail 6
2.1 Counting Rules 6

3 Compare Reload in Detail 7
3.1 Reload by Shadow Transfer Rules 7
3.2 CCU4 Shadow Transfers for Coherent Signal Pattern Update 8
3.2.1 The Global Shadow Transfer Set Enable Register 8
3.3 Shadow Transfer of Compare Register values 8
3.3.1 Compound Shadow Transfers 9

4 CCU4 Output Control 10
4.1 Output State and Output Pin PASSIVE / ACTIVE Level Control 10
4.2 External Control by Compare Events 10
4.3 Top-Level Control of Event Requests to/from a Timer in Compare Mode 11
4.4 CCU4 Using Top-Level Interconnect Control with Event Request Unit (ERU) 11
4.5 CCU4 and ERU1 Use Cases 11
4.5.1 Example – Using CCU4 and ERU1 for Delayed ADC-Start Controlled by an IO 11

5 Getting Started with CCU4 and Compare Mode Features 12
5.1 Ultimate Initialization Sequence 12
5.1.1 Ultimate Start-Up Enable 12
5.2 Setting Compare Mode Control 12
5.2.1 The Shadow Transfer Events for all registers concerned 12
5.2.2 Setting the Output Pin PASSIVE / ACTIVE Level Control 12
5.2.3 Initialize and Start the Timer 12
5.2.4 Using the Compare Mode / PWM Control Registers 12
5.2.5 Pseudo Code for Setting Compare Mode Control 12
5.3 External Events Control Setup Registers 13
5.3.1 Using the External Start Control Register Bitfields 13
5.3.2 Using the External Stop Control Register Bitfields 13
5.3.3 Pseudo Code for the External Start/Stop Control Setup 13

List of Figures

Figure 1 Timer Slice Compare Registers and PWM related Blocks 5
Figure 2 The Compare Blocks 5
Figure 3 PWM Range in Up-Count Mode 6
Figure 4 PWM Range in Down-Count Mode 6
Figure 5 PWM Range 0 – 100% in Center Aligned Mode 6
Figure 6 Compare Counting Schemes in Detail 7
Figure 7 Compare Reload Schemes in Detail 8
Figure 8 Shadow Transfer Mechanism with Compare Registers 9
Figure 9 Compound Shadow Transfer Mechanism with Coherent Update of PWM 10
Figure 10 Using CCU4 and ERU1 for Delayed ADC Start Controlled by an IO 11

1 Output Pattern Generation with CCU4

Each CAPCOM4 (CCU40/../43) is a multi-purpose timer unit for many use cases. It is designed with repetitive structures with multiple timer slices that have all the same base functionalities. This allows the development of re-usable SW low level drivers. The main function blocks of 1 of the 4 CC4y slices on a CCU4x is shown here. See Figure 1.

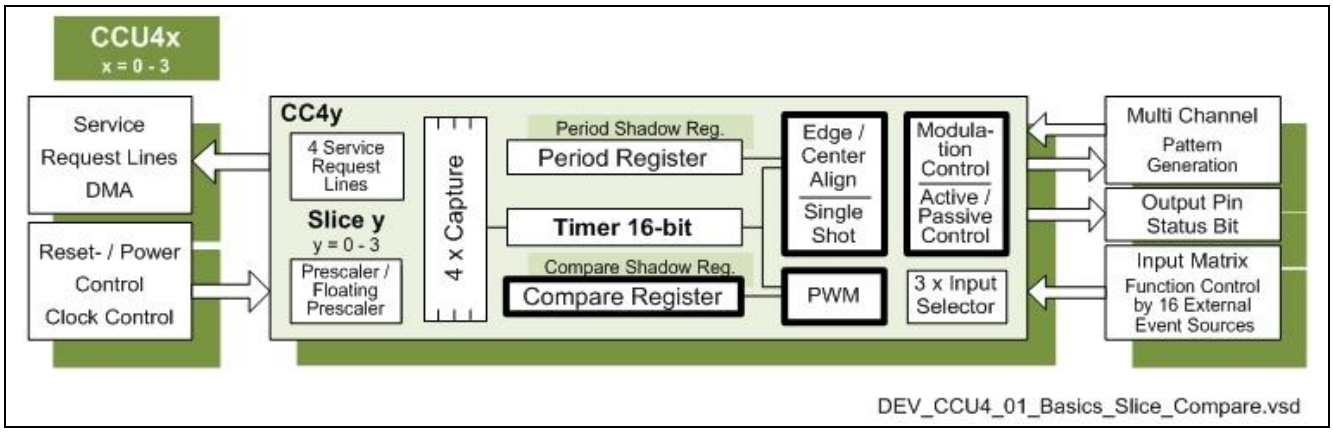

Figure 1 Timer Slice Compare Registers and PWM related Blocks

1.1 The Principle Compare Blocks

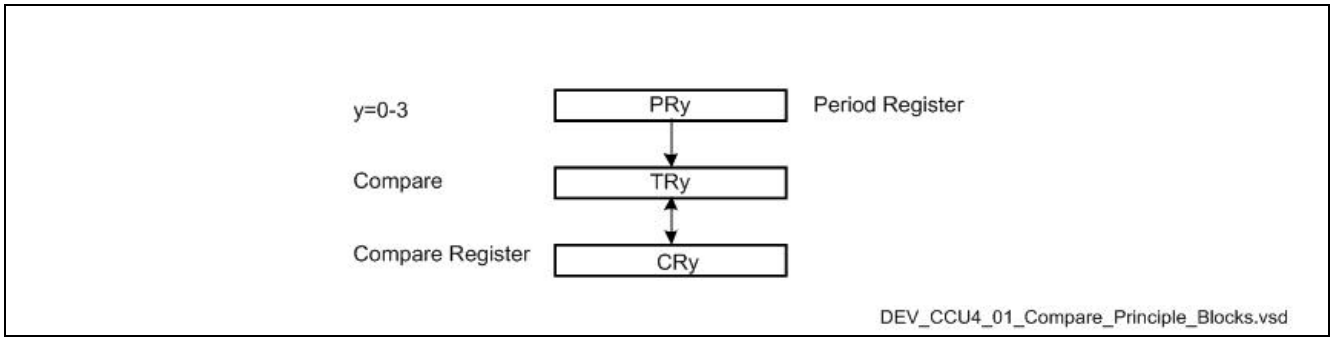

Figure 2 The Compare Blocks

1.2 PWM Range 0 – 100% in Up-Count Mode

The Compare Rule in Up-Count Mode is very simple: As long as the timer register value is equal or greater than the compare register value, the Status Bit (CCST or even named CCU4xSTy) will be set to one (=1). Otherwise it is set to zero (=0). The dynamic PWM range can be set to any value from 0 up to 100%. See Figure 3.

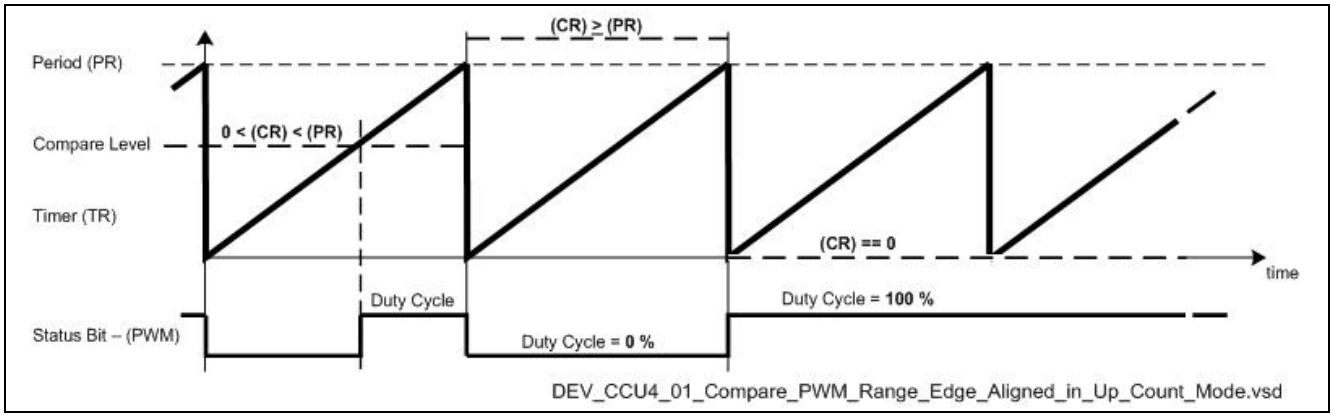

Figure 3 PWM Range in Up-Count Mode

1.3 PWM Range 0 – 100% in Down-Count Mode

The Compare Rule in Dowm-Count Mode is the same as in Up-Count Mode: When the timer register value is equal or greater than the compare register value, the Status Bit (CCST or CCU4xSTy) will be set to one (=1). Otherwise it is set to zero (=0). The dynamic PWM range can be set to any value 0 – 100%. See Figure 4.

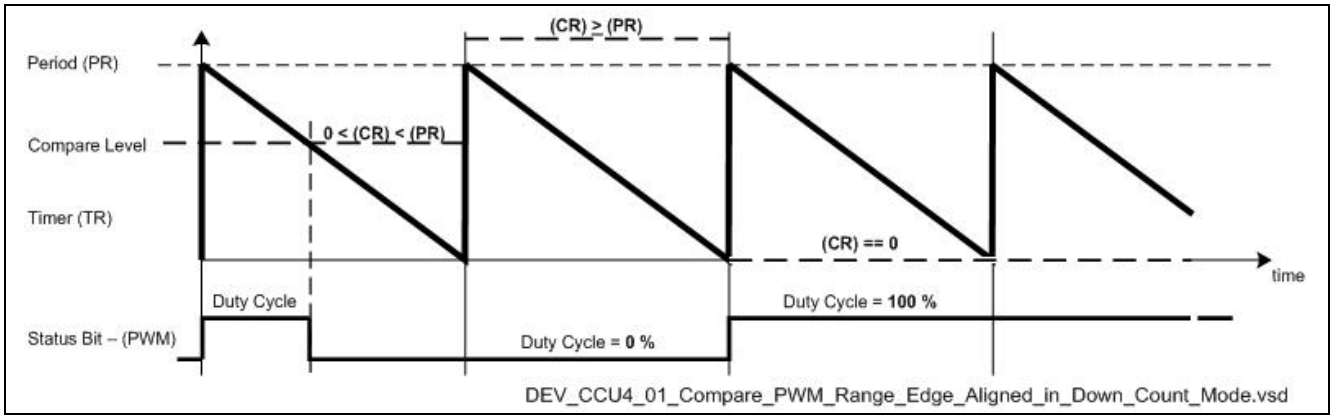

Figure 4 PWM Range in Down-Count Mode

1.4 PWM Range 0 – 100% in Center Aligned Mode

The Compare Rule in Center Aligned Mode is the same as in Up- or Down-Count Modes: When the timer register value is equal or greater than the compare register value, the Status Bit (CCST or CCU4xSTy) will be set to one (=1). Otherwise it is set to zero (=0). The PWM value may vary from 0 to 100%. See Figure 5-

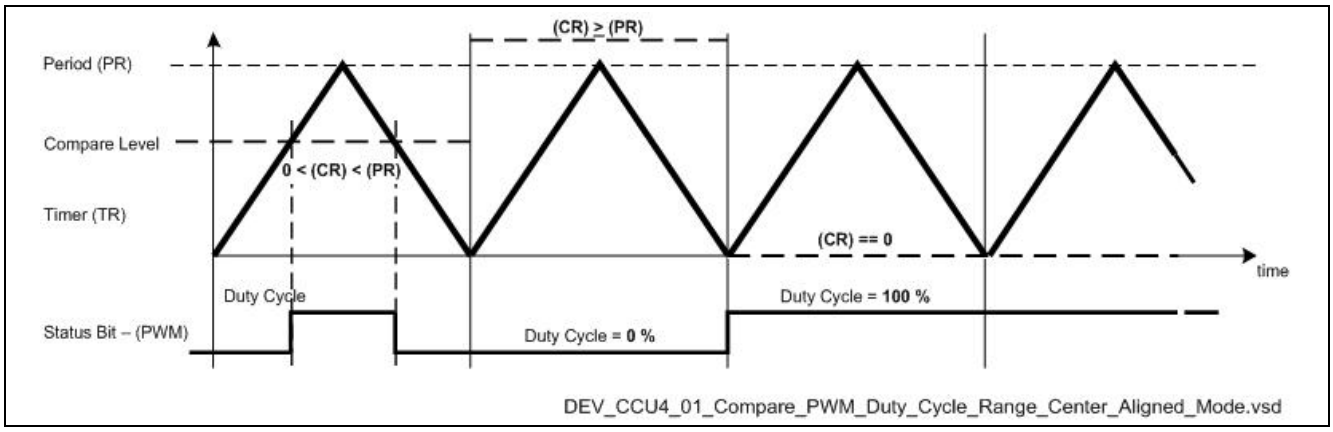

Figure 5 PWM Range 0 – 100% in Center Aligned Mode

2 Compare Counting in Detail

2.1 Counting Rules

See Figure 6.

One count cycle after the Counting Direction status bit **CC8yTCST**.CDIR has been changed, the timer counting direction will have been changed at the next count. In Center Aligned Mode this bit will be changed from 0 to 1, one cycle after a Period-Match state is entered – and from 1 to 0, one count cycle after a One-Match is entered.

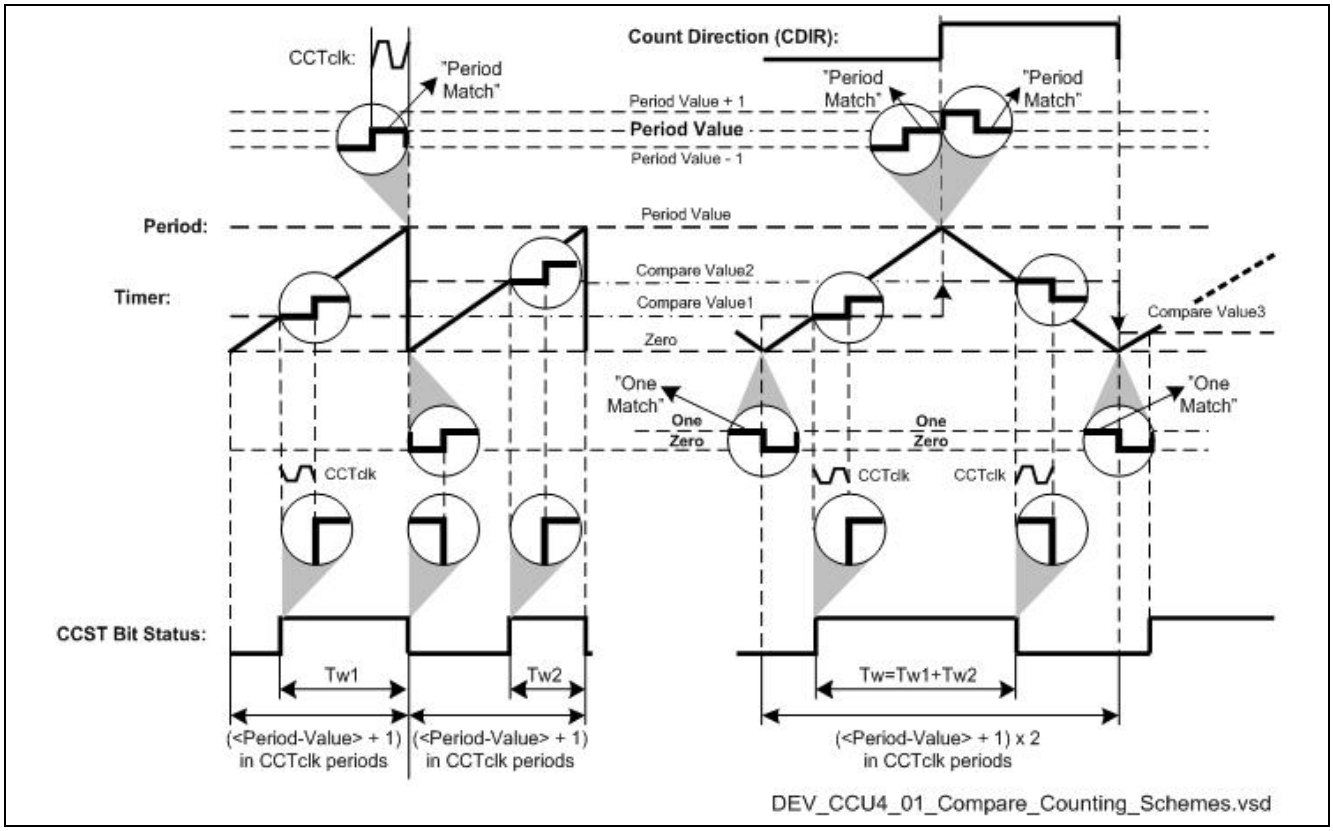

Figure 6 Compare Counting Schemes in Detail

3 Compare Reload in Detail

3.1 Reload by Shadow Transfer Rules

See Figure 7.

Reload of registers by *Shadow Transfers* from associated shadow registers will occur according to the following:

- In the next clock cycle after a Period Match while counting up
- In the next clock cycle after an One Match while counting down
- Immediately if the timer is stopped and a transfer request was triggered (not illustrated here though)

Note: This implies that it is still due time to Set Transfer Enable requests as late as on a Period Match event, by using a Period Match Interrupt Service with a shorter latency time than the last clock cycle before the Shadow Transfer occurs. Normally the Compare Event Interrupts are used for this purpose. However, it shows that even a compare register value equal to the period register value is accurate for Transfer Enable request settings – in other words: The entire timer period may be used to Set Transfer Enable bits.

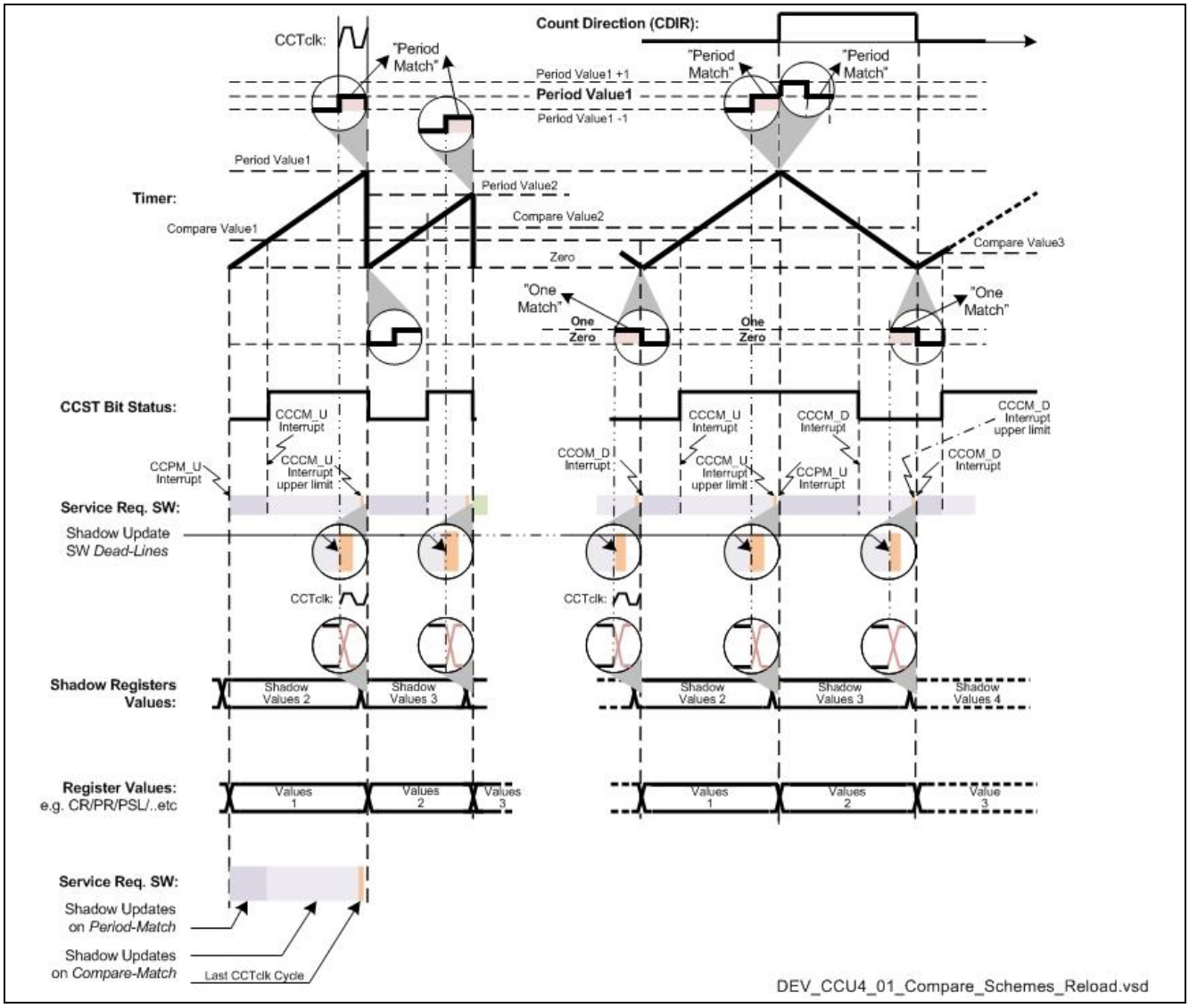

Figure 7 Compare Reload Schemes in Detail

3.2 CCU4 Shadow Transfers for Coherent Signal Pattern Update

All CAPCOM4 timers, in any slice configuration, no matter how complex or for what signal pattern, are assured *coherent update* by HW of all concerned timer function parameters – by update values in shadow registers that, on *a global* preset request, are transferred simultaneously to all function registers at a **Period-** or **One-Match**.

3.2.1 The Global Shadow Transfer Set Enable Register

There is one, global register, **GCSS**, carrying *all* enable-flags that have to be preset by SW to selectively activate the targeted Shadow Transfer Requests, which will be cleared by HW after the transfer. By this manner total realtime correctness is achieved also from the logic operations aspect – essential for safe power switching.

3.3 Shadow Transfer of Compare Register values

The compare values that are targeted for an update operation have to be written into the **CC4yCRS** shadow registers - *AND the corresponding* **Slice Transfer Set Enable** bits, e.g. **SySE** in **GCSS**, *must be preset – latest within the clock cycle of* **Period Match** (in *Edge Aligned Mode*) or **Period/One Match** (in *Center Aligned Mode*)

See Figure 8.

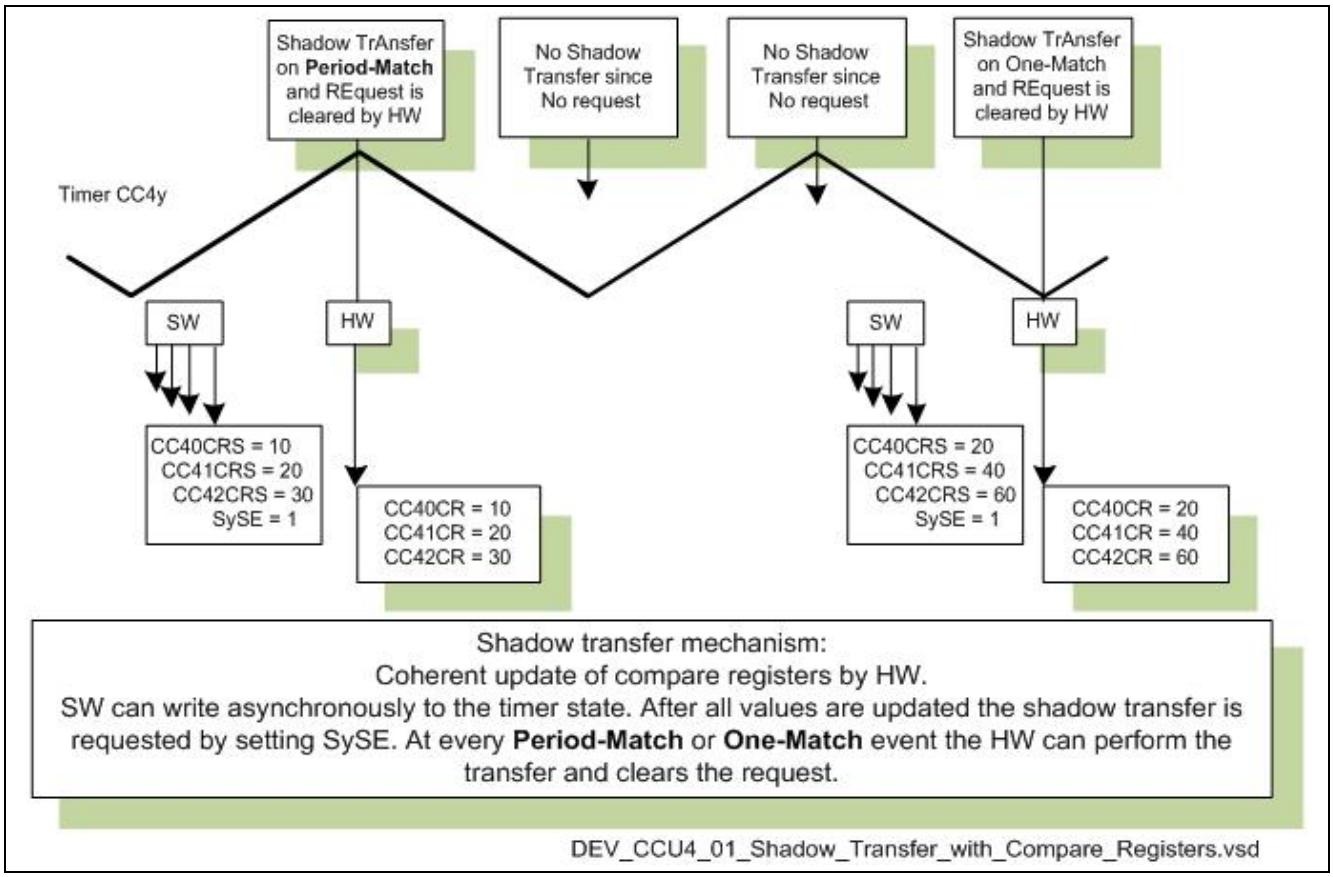

Figure 8 Shadow Transfer Mechanism with Compare Registers

3.3.1 Compound Shadow Transfers

Beside the *Compare* (CR) values, there are also the the timer *Period* register (PR) and the PWM *Active/Passive control* bit (PSL) that are updated simultaneously on the **SySE** flag. *Dithering* or *Floating Prescaler* values (not shown though) may get simultaneous update too by the **SyDSE** & **SyPSE** request flags. See Figure 9.

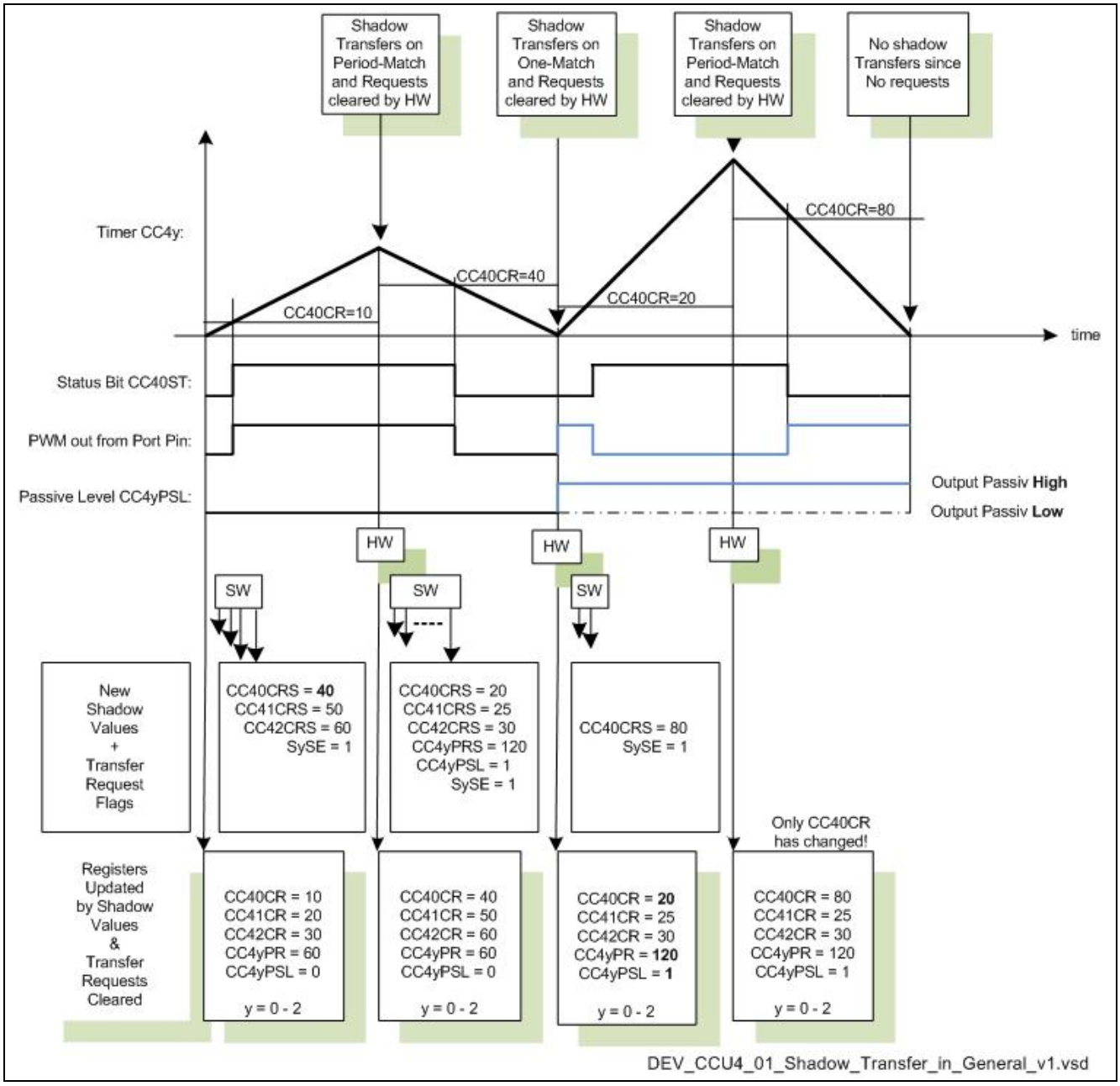

Figure 9 Compound Shadow Transfer Mechanism with Coherent Update of PWM

4 CCU4 Output Control

4.1 Output State and Output Pin PASSIVE / ACTIVE Level Control

The PASSIVE/ACTIVE state of a slice *internal* output **CCUxSTy** (i.e. status bit **CC4yST)** is controlled by the compare level and the External Modulation Mode. Whether the *external* output pin state **CCU4xOUTy** (e.g. the PWM) should be *Passive Low / Active High* or vice versa, is controlled by the **CC4yPSL** *Passive/Active* bit **PSL**.

4.2 External Control by Compare Events

A compare event may *trigger external actions* via the **Top-Level Interconnect** matrix or *request for an Interrupt*. Each CAPCOM4 has four **Service Request Lines** and each slice has a dedicated output signal **CC4ySR[3...0]** selectable to a line by **CC4ySRS.** E.g. compare events may request for immidiate ADC actions or for interrupts.

4.3 Top-Level Control of Event Requests to/from a Timer in Compare Mode

Top-Level control means also *conditional* control of events requests between a slice and other action providers. The Event Request Unit (**ERU1**) *plus* the **Top-Level Interconnect** matrix may *combine, control and link* event signals according to *user defined* request-to-action event patterns - e.g. ADC triggering combined on I/O events.

4.4 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 10.

4.5 CCU4 and ERU1 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.

4.5.1 Example – Using CCU4 and ERU1 for Delayed ADC-Start Controlled by an IO

See Figure 10.

The CCU4xCC4y is here a single shot *delay timer*. Its status bit (*red*) that is delayed and set by compare event, shall delay an ADC-start when triggered by a PWM timer (*blue*) on a GPIO state (*orange*). The ERU1 *combines*, *detects* and *links* it all as a trigger (*green*) via the delay timer and the Top-Level Interconnect matrix to the ADC.

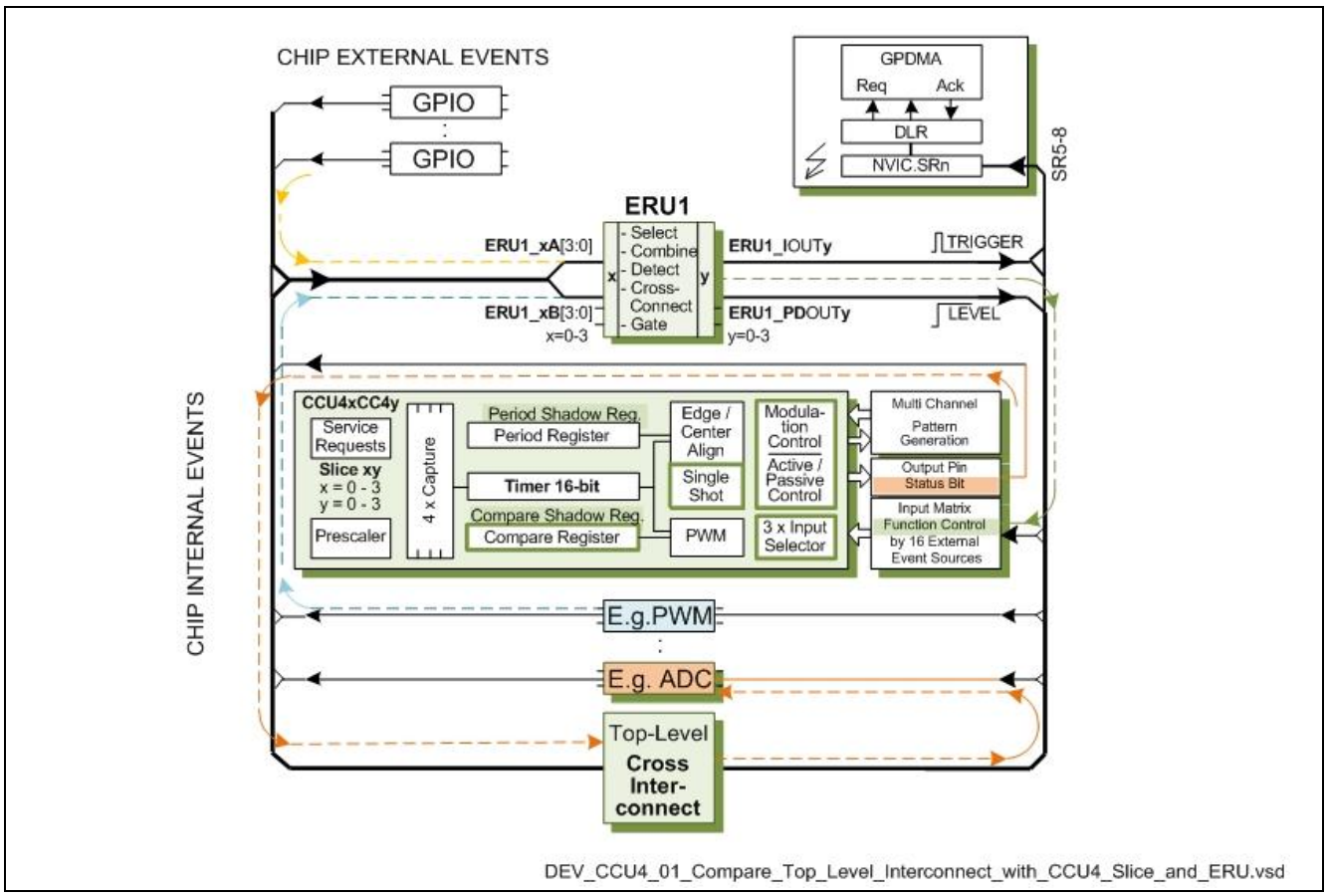

Figure 10 Using CCU4 and ERU1 for Delayed ADC Start Controlled by an IO

5 Getting Started with CCU4 and Compare Mode Features

5.1 Ultimate Initialization Sequence

Before start and execution of a timer slice SW first time, the CAPCOM4 must have been initialized appropriately according to a specific **CAPCOM4 Initialization Sequence:** 1. *Apply Reset*, 2. *Release Reset*, 3. *Enable Clock*, 4. *Enable Prescaler Block*, 5. *Configure Global Control*, 6. *Configure Slice(s) Functions*, *Interrupts* and *Start-up*.

5.1.1 Ultimate Start-Up Enable

In the last part of the ultimate **CAPCOM4 Initialization Sequence** the startup value(s) for a specific Compare Channel Status of the Timer Slice(s) may be configured by the respective **GCSS.SyTS** bit. After that the default *IDLE mode has to be removed* from the Timer Slice(s) in register **GIDLC** and Start or Global Start can be done!

5.2 Setting Compare Mode Control

Compare Mode is the default state of a CAPCOM timer, i.e. **CC4yTC.**CMOD=0_B. The *compare level* will be set in the read-only compare register **CC4yCR** by the value from the compare *shadow* register **CC4yCRS** on each *shadow transfer, if requested within the current period cycle* by the **Slice Transfer Set Enable** bit **GCSS.**SySE.

5.2.1 The Shadow Transfer Events for all registers concerned

- In the next clock cycle after a Period Match while counting up
- In the next clock cycle after an One Match while counting down
- Immediately if the timer is stopped and a transfer request was triggered

5.2.2 Setting the Output Pin PASSIVE / ACTIVE Level Control

The PASSIVE/ACTIVE state of a slice *internal* output **CCUxSTy** (i.e. status bit **CC4yST)** is controlled by the compare level and the External Modulation Mode. Whether the *external* output pin state **CCU4xOUTy** (e.g. the PWM) should be *Passive Low / Active High* or vice versa, is controlled by the **CC4yPSL** *Passive/Active* bit **PSL**.

5.2.3 Initialize and Start the Timer

After the Ultimate Start procedures have been executed, the SW may 1) initialize the slice timer **CC4yTIMER** or just clear it with a **CC4yTCCLR**.TCC bit=1 command and 2) start the timer with a Set Timer Run command in the **CC4yTCSET**.TRBS bit. The timer may be stopped at any time by SW with a **CC4yTCCLR**.TRBC command.

5.2.4 Using the Compare Mode / PWM Control Registers

1.	**CC4yTC.**CMOD	CAPCOM Mode selection bit should be set =0 (which is the default value)
2.	**CC4yTC.**TCM	Timer **Counting Mode selection** bit: 0 → *Edge-Aligned* / 1 → *Center-Aligned*
3.	**CC4yCRS.**CRS	Write the wanted **Compare Level** value into the compare *shadow* register
4.	**CC4yPR.**PR	Write the timer count cycle upper limit, **Period Match**, into the *period* register
5.	**CC4yPSL.**PSL	**PASSIVE/ACTIVE** control bit: (*CCU4xOUTy)* = (*CCUxSTy*) xor (*CC4yPSL.PSL*)
Optionally:		
6.	**CC4yTIMER**	Initialize a timer start value (or *flush* by the **CC4yTCCLR**.TCC bit=1 command)
7.	**GCSS.**SySE	Request at least once a Shadow Transfer to *Initialize the Compare Register*
8.	**CC4yTCSET**.TRBS	Start Timer by SW

5.2.5 Pseudo Code for Setting Compare Mode Control

```
// This is an independent 32-bit based Pseudo Code, regarding that there are no
connections to a certain CAPCOM4 Unit (CCU4x) nor specific Variable Declarations.
// step1
```

5.3 External Events Control Setup Registers

External Events Control would be useful e.g. for *Start/Stop* options. 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 any *Event[2:0] Input Line* to any of the 11 *Input Functions*, e.g. *Start*.

5.3.1 Using the External Start Control Register Bitfields

9. **CC4yCMC.**STRTS Input Function Selection and Event Number x=0-2 Input Line Selection bitfield
10. **CC4yINS.**<Event-*x* bitfields> Event-x Source EVxIS (**CC4yINS**[P:A]) and Event Edge Mode EVxEM bitfields
11. **CC4yTC.**<bitfields> Timer Mode selections for e.g. *Start, Extended start, Edge or Center Align, etc*

Note: If External Events Control of Up/Down Counting is used – then the counting scheme is always edge aligned and the Timer Counting Mode bit (TCM) is read-only!

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 STRTS Input Function is x == <STRTS>-1 (when <STRTS> != 0).

5.3.2 Using the External Stop Control Register Bitfields

Besides the Timer Single Shot Mode (by the **CC4yTC.**TSSM bit set), a timer may be stopped by External Events Control as well - even by its own events! The following Control Register Bitfields should be used then:

12. **CC4yCMC.**ENDS Input Function Selection and Event Number x=0-2 Input Line Selection bitfield
13. **CC4yINS.**<Event-*x* bitfields> Event-x Source EVxIS (**CC4yINS**[P:A]) and Event Edge Mode EVxEM bitfields
14. **CC4yTC.**<bitfields> Timer Mode selections for e.g. *Stop, Extended Stop*

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

5.3.3 Pseudo Code for the External Start/Stop Control Setup

```
// This is an independent 32-bit based Pseudo Code, regarding that there are no
connections to a certain CAPCOM4 Unit (CCU4x) nor specific Variable Declarations.
// step1
// Make a MASK OPERATION first for bitfield focusing - if not first time - else:
CC4yCMC |= STRTS_FUNCTION_AND_EVENT_NUMBER_BITFIELD //Initialize External Start
// step 2
CC4yINS &=!STRTS_SOURCE_AND_EVENT_MODE_BITFIELDS_MASK; //Focus on STRTS events
CC4yINS |= STRTS_SOURCE_AND_EVENT_MODE_BITFIELDS_PATTERN; //Paste setup pattern
// step 3
CC4yTC &=!STRTS_MODES_AND_EXTENDED_MODE_BITFIELDS_MASK; //Focus on STRTS modes
CC4yTC |= STRTS_MODES_AND_EXTENDED_MODE_BITFIELDS_PATTERN; //Paste setup pattern

// step4
// Make a MASK OPERATION first for bitfield focusing - if not first time - else:
CC4yCMC |= ENDS_FUNCTION_AND_EVENT_NUMBER_BITFIELD //Initialize External Start
// step 5
CC4yINS &=!ENDS_SOURCE_AND_EVENT_MODE_BITFIELDS_MASK; //Focus on STRTS events
CC4yINS |= ENDS_SOURCE_AND_EVENT_MODE_BITFIELDS_PATTERN; //Paste setup pattern
// step 6
CC4yTC &=!ENDS_MODES_AND_EXTENDED_MODE_BITFIELDS_MASK; //Focus on STRTS modes
CC4yTC |= ENDS_MODES_AND_EXTENDED_MODE_BITFIELDS_PATTERN; //Paste setup pattern
```

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

Published by Infineon Technologies AG