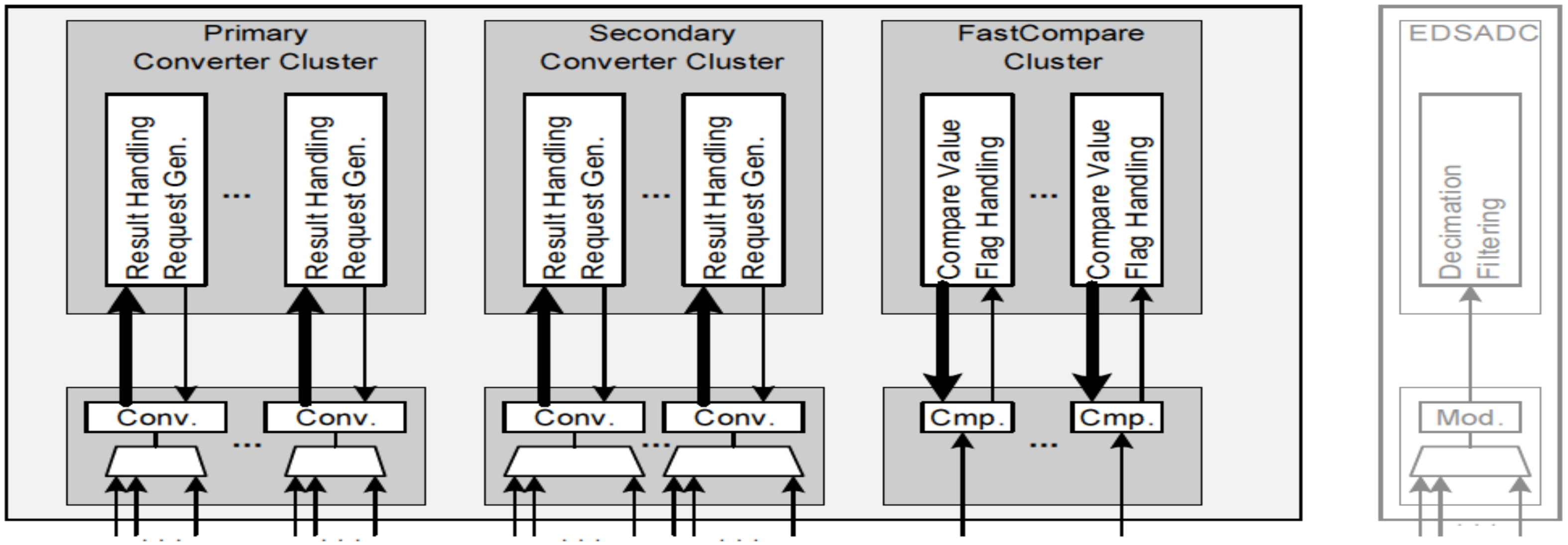
AURIX 2G EVADC

Enhanced Versatile Analog-to-Digital Converter

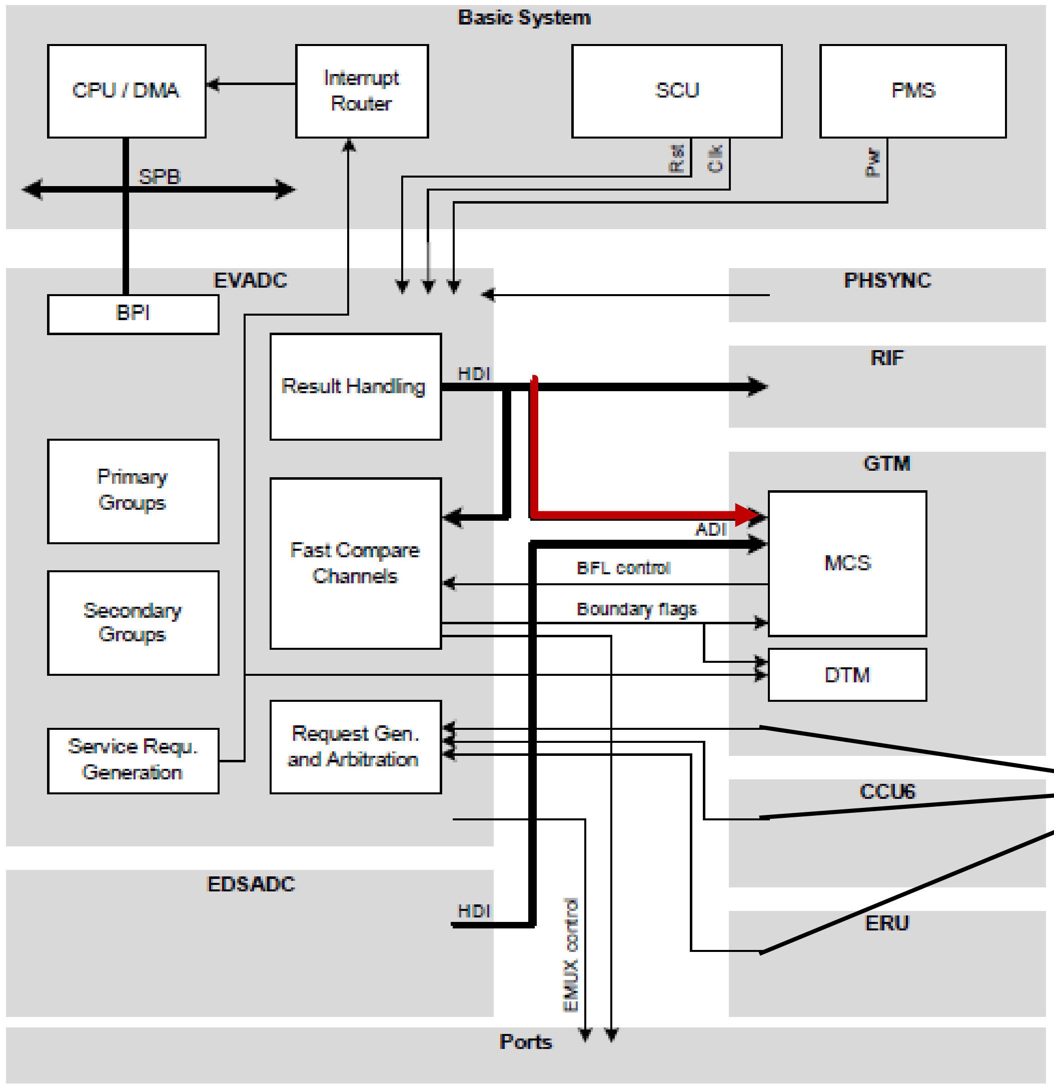
Hansen Chen
IFCN ATV SMD GC SAE MC
2018/5/2

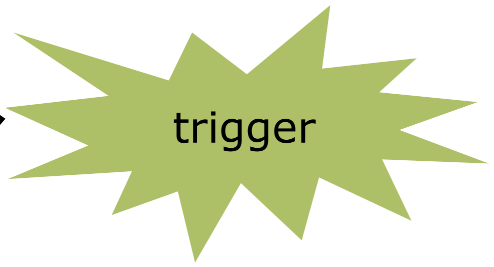
AURIX™ - TC3xx
EVADC – Feature Overview

- Each converter of the ADC clusters can operate independent of the others.
- The conversion result values of a certain group can be stored in one of the 16 associated group result registers (**GxRESy**) or in the common global result register(**GLOBRES**).
- Three clusters with different functionality are available:
  - **Primary converter cluster**
    - Equipped with 8:1 multiplexers and 8-stage queues, conversion time down to **400** ns.
  - **Secondary converter cluster**
    - Equipped with 16:1 multiplexers and 16-stage queues, conversion time down to **714** ns.
  - **Fast compare cluster**
    - Single channels, update rate down to **200** ns.


Primary Converter Cluster
Result Handling Request Gen.
Result Handling Request Gen.
Conv.
Conv.
Secondary Converter Cluster
Result Handling Request Gen.
Result Handling Request Gen.
Conv.
Conv.
FastCompare Cluster
Compare Value Flag Handling
Compare Value Flag Handling
Cmp.
Cmp.
EDSADC
Decimation Filtering
Mod.

AURIX™ - TC3xx Integration of EVADC

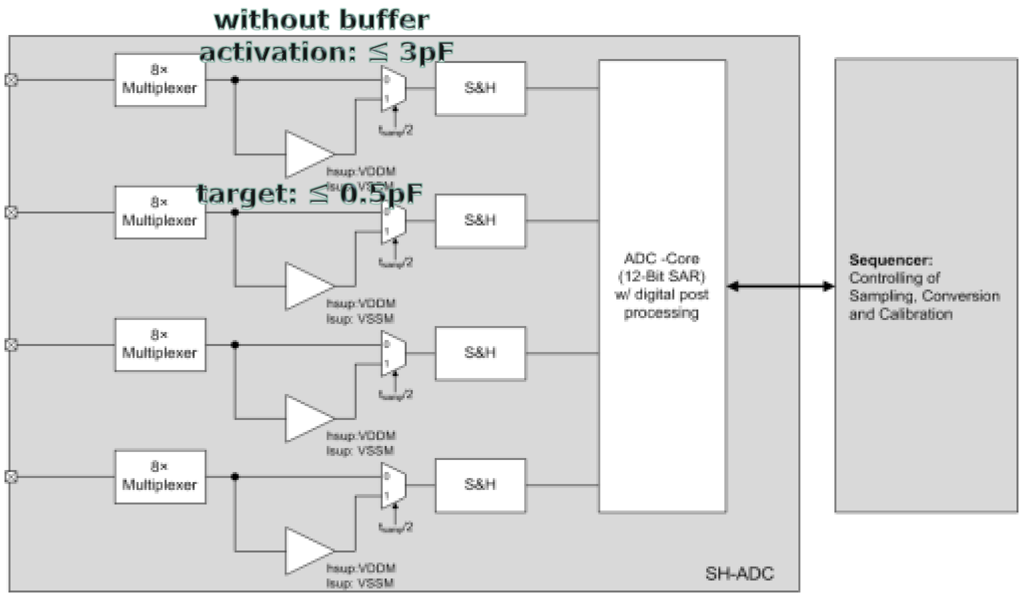




› System Integration and Communication
  – EVADC is directly connected to other hardware modules of the AURIX™ TC3xx Platform.

AURIX™ - TC3xx– Enhancement!
EVADC – Buffer for Analog Input (Pre-charge Control)

- Integrated amplifier to support much faster sampling with lower charge consumption from the external blocking capacitor (from default max switching capacitance of 3pF down to 0.5pF)
  - limited to 4.5V (VDDM-0.5V)
  - When higher than 4.5V, be charged from the external blocking capacitor
- The analog input buffer boosts the selected analog input signal for a certain time, when enabled. The time during which the input buffer is active can be adapted to the configured sample time by bitfields **AIPS**/**AIPE** in register **GxICLASS0** (x=0-7, 8-11) etc.
  - When the analog input buffer is activated (BE = 1 in register GxANCFG (x=0-7, 8-11)), it needs a certain setup time to settle before a conversion can be properly executed.
    - ● The input buffer setup time is typ 0.4µs max 1µs.


without buffer activation: ≤ 3pF
target: ≤ 0.5pF
8x Multiplexer
S&H
hsup: VDDM
Isup: VSSM
ADC -Core (12-Bit SAR) w/ digital post processing
Sequencer: Controlling of Sampling, Conversion and Calibration
SH-ADC

AIPS

Analog Input Precharge Control for Standard Conversions

00_B	No precharge
01_B	Precharge for 8 clock cycles
10_B	Precharge for 16 clock cycles
11_B	Precharge for 32 clock cycles

Note: Buffer must be enabled by BE = 1 (see ***GxANCFG (x=0-7, 8-11)****).*

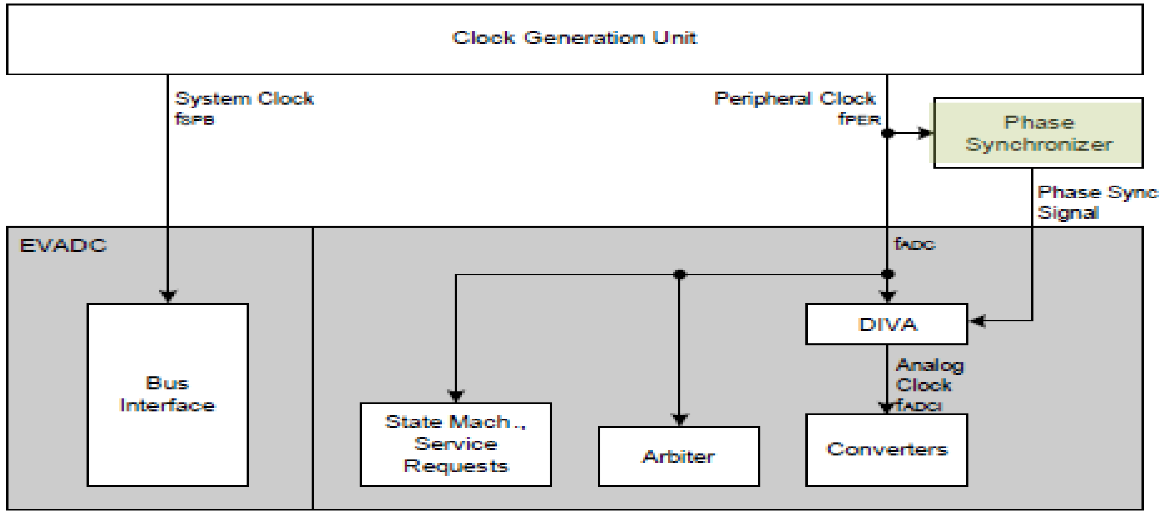
AIPE

Analog Input Precharge Control for EMUX Conversions

Note: Buffer must be enabled by BE = 1 (see GxANCFG).

00_B	No precharge
01_B	Precharge for 8 clock cycles
10_B	Precharge for 16 clock cycles
11_B	Precharge for 32 clock cycles

AURIX™ - TC3xx
EVADC – Clock System with new Phase Synchronizer



Attention: ***To ensure proper synchronization between the clock domains of the EVADC, the peripheral clock must never be slower than the bus clock, i.e.*** $f_{ADC} >= f_{SPB}$.

› The EVADC is operated with the peripheral clock signal ($f_{ADC} = f_{PER}$).

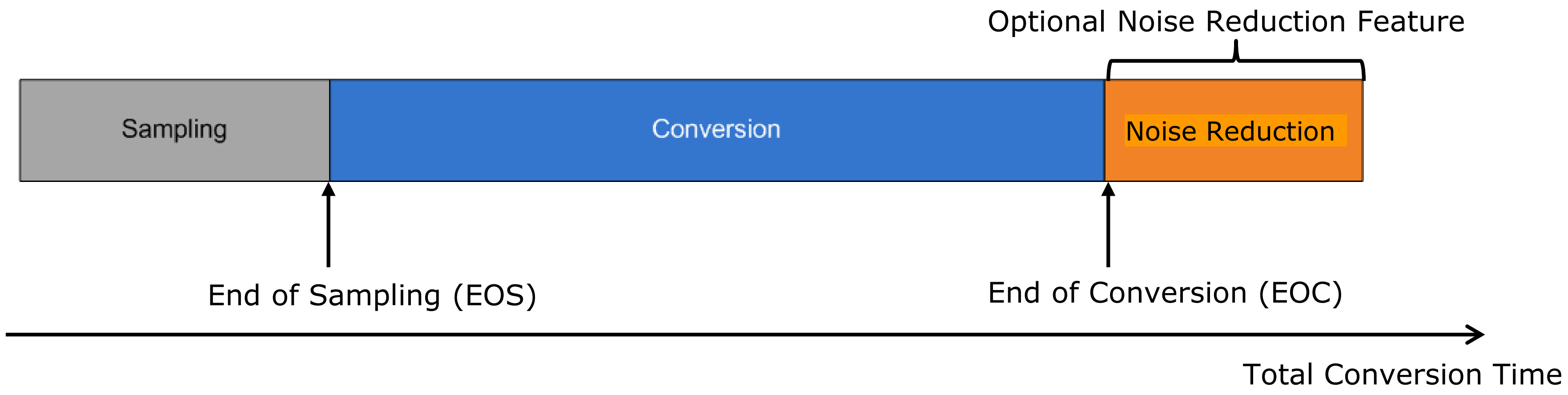
› The converters are operated with the analog clock f_{ADCI}

› The global configuration register defines common clock bases for all converters of all clusters.

  – This ensures deterministic behavior of converters that shall operate in parallel.

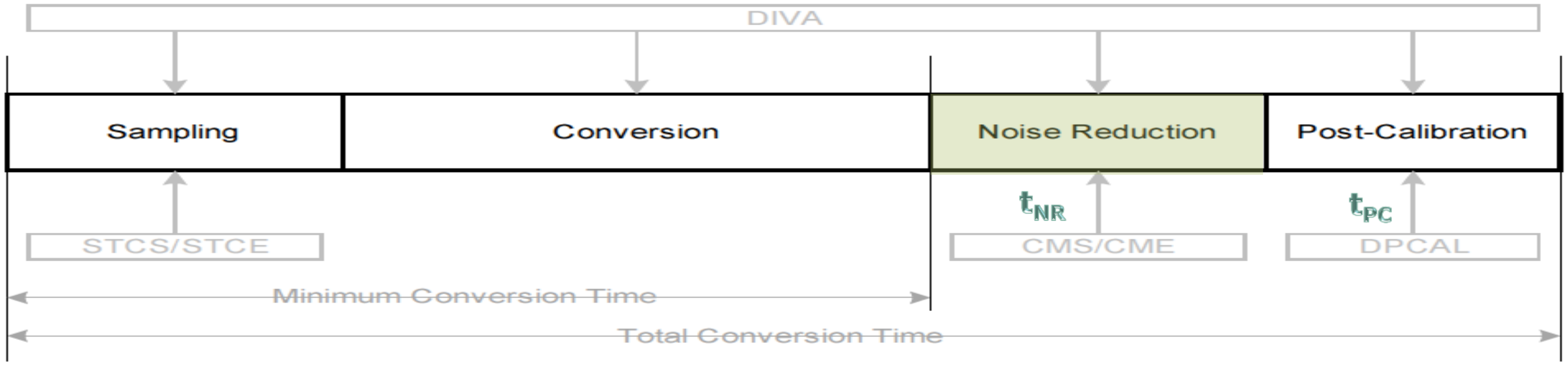
› The analog clock is based on the analog **Phase Synchronizer.**

AURIX™ - TC3xx– Noise Improvements! Noise Reduction Feature



- Generally the ADC accuracy can be affected:
  - Deterministic noise source: Periodic signal caused by the SOC part of the µC
  - Stochastic noise source: Gaussian distributed noise sources caused by the ADC implementation
- Gaussian distributed noise can be reduced generally by oversampling
  - Four-fold oversampling increases accuracy by one effective number of bit (ENOB)
  - Total conversion time is increased by $t_{NR} = 4 \times t_{ADCI} + 3 \times t_{ADC}$ per conversion step
  - Noise Reduction feature generates the arithmetic average of the four last significant bits
  - Analog input signal needs to be bandwidth limited

AURIX™ - TC3xx
EVADC – Conversion Timing & Noise Reduction Mode



› The conversion time is the sum of sample time, conversion steps, and internal steps. It can be computed with the following formula with $f_{ADCI} = f_{ADC}/(DIVA+1)$:

– $\mathbf{t_{C12} = [(2+STC)\times t_{ADCI}] + [13\times t_{ADCI}] + [NRS\times t_{NR}] + [t_{PC}] + [3\times t_{ADC}]}$, with:

- $t_{NR} = 4\times t_{ADCI} + 3\times t_{ADC}$ per conversion step
- $t_{PC} = (4 + 2\times CALSTC)\times t_{ADCI} + 5\times t_{ADC}$ if enabled, otherwise 0

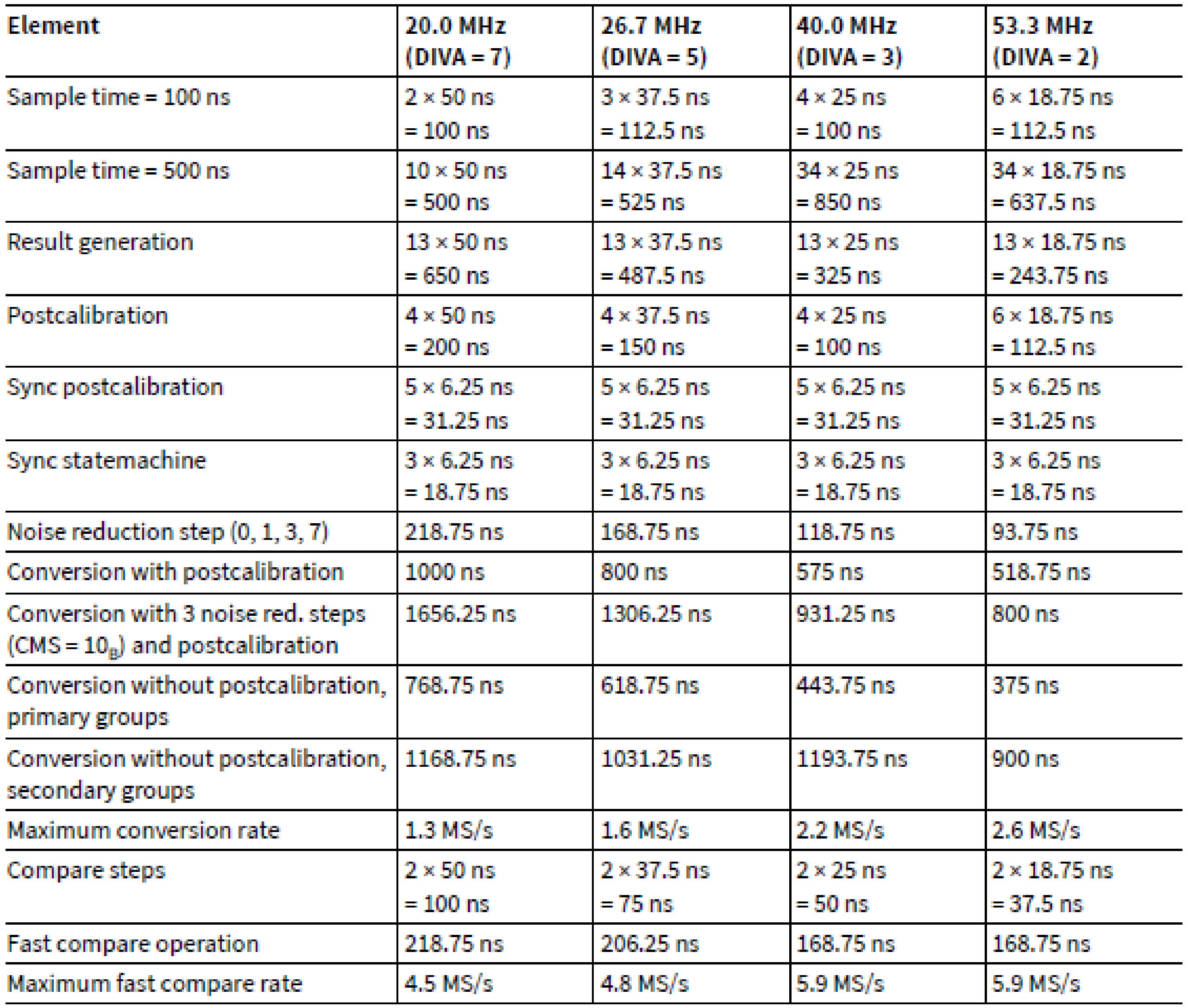
› Noise-Reduction Conversions

– Conversions can be extended by a selectable number of additional steps (1, 3, 7) to refine the generated result value. The noise reduction level is configured by bitfields CMS/CME in register GxICLASS0 (x=0-7, 8-11) etc.

Field	Bits	Type	Description
CMS	[9:8]	rw	**Conversion Mode for Standard Conversions** 00_B Standard conversion 01_B Noise reduction conversion level 1, 1 additional conversion step 10_B Noise reduction conversion level 2, 3 additional conversion steps 11_B Noise reduction conversion level 3, 7 additional conversion steps

NRS = additional conversion steps
NRS = 0/1/3/7

AURIX™ - TC3xx
EVADC – Conversion Timing & Noise Reduction Mode

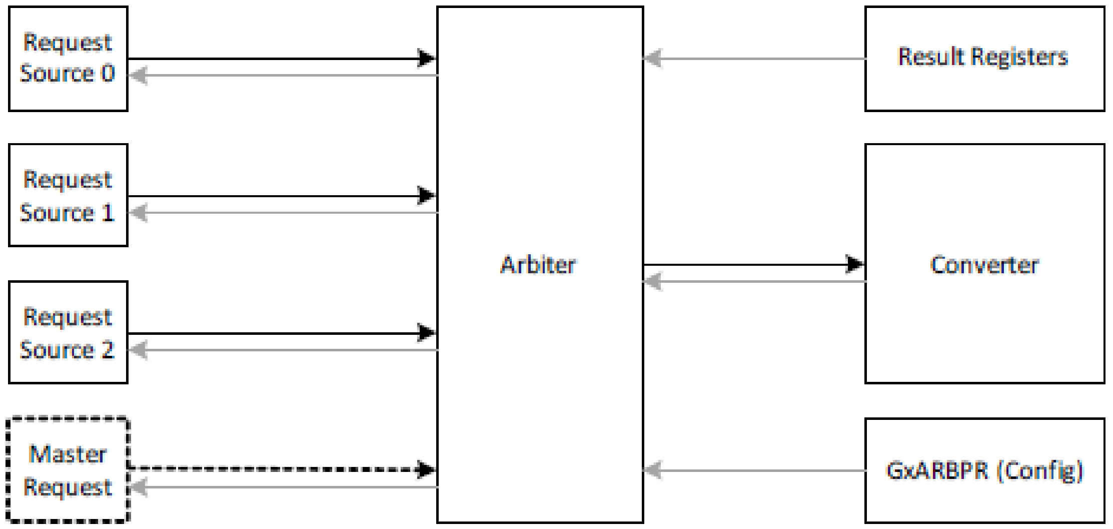


Element	20.0 MHz (DIVA = 7)	26.7 MHz (DIVA = 5)	40.0 MHz (DIVA = 3)	53.3 MHz (DIVA = 2)
Sample time = 100 ns	2 × 50 ns = 100 ns	3 × 37.5 ns = 112.5 ns	4 × 25 ns = 100 ns	6 × 18.75 ns = 112.5 ns
Sample time = 500 ns	10 × 50 ns = 500 ns	14 × 37.5 ns = 525 ns	34 × 25 ns = 850 ns	34 × 18.75 ns = 637.5 ns
Result generation	13 × 50 ns = 650 ns	13 × 37.5 ns = 487.5 ns	13 × 25 ns = 325 ns	13 × 18.75 ns = 243.75 ns
Postcalibration	4 × 50 ns = 200 ns	4 × 37.5 ns = 150 ns	4 × 25 ns = 100 ns	6 × 18.75 ns = 112.5 ns
Sync postcalibration	5 × 6.25 ns = 31.25 ns	5 × 6.25 ns = 31.25 ns	5 × 6.25 ns = 31.25 ns	5 × 6.25 ns = 31.25 ns
Sync statemachine	3 × 6.25 ns = 18.75 ns	3 × 6.25 ns = 18.75 ns	3 × 6.25 ns = 18.75 ns	3 × 6.25 ns = 18.75 ns
Noise reduction step (0, 1, 3, 7)	218.75 ns	168.75 ns	118.75 ns	93.75 ns
Conversion with postcalibration	1000 ns	800 ns	575 ns	518.75 ns
Conversion with 3 noise red. steps (CMS = 10_B) and postcalibration	1656.25 ns	1306.25 ns	931.25 ns	800 ns
Conversion without postcalibration, primary groups	768.75 ns	618.75 ns	443.75 ns	375 ns
Conversion without postcalibration, secondary groups	1168.75 ns	1031.25 ns	1193.75 ns	900 ns
Maximum conversion rate	1.3 MS/s	1.6 MS/s	2.2 MS/s	2.6 MS/s
Compare steps	2 × 50 ns = 100 ns	2 × 37.5 ns = 75 ns	2 × 25 ns = 50 ns	2 × 18.75 ns = 37.5 ns
Fast compare operation	218.75 ns	206.25 ns	168.75 ns	168.75 ns
Maximum fast compare rate	4.5 MS/s	4.8 MS/s	5.9 MS/s	5.9 MS/s

AURIX™ - TC3xx
EVADC – Arbiter and Request Source Arbitration

› The request source arbiter resolves the different request sources in a combinatorial way which means there is no sampling jitter anymore.

– **Input 0:**	Queued source **Q0**, 8/16-stage sequences in arbitrary order
– **Input 1:**	Queued source **Q1**, 8/16-stage sequences in arbitrary order
– **Input 2:**	Queued source **Q2**, 8/16-stage sequences in arbitrary order, intra-group concatenation, test
– **Input 3:**	Synchronization source, synchronized conversion requests from another ADC kernel (always handled with the highest priority in a synchronization slave kernel)



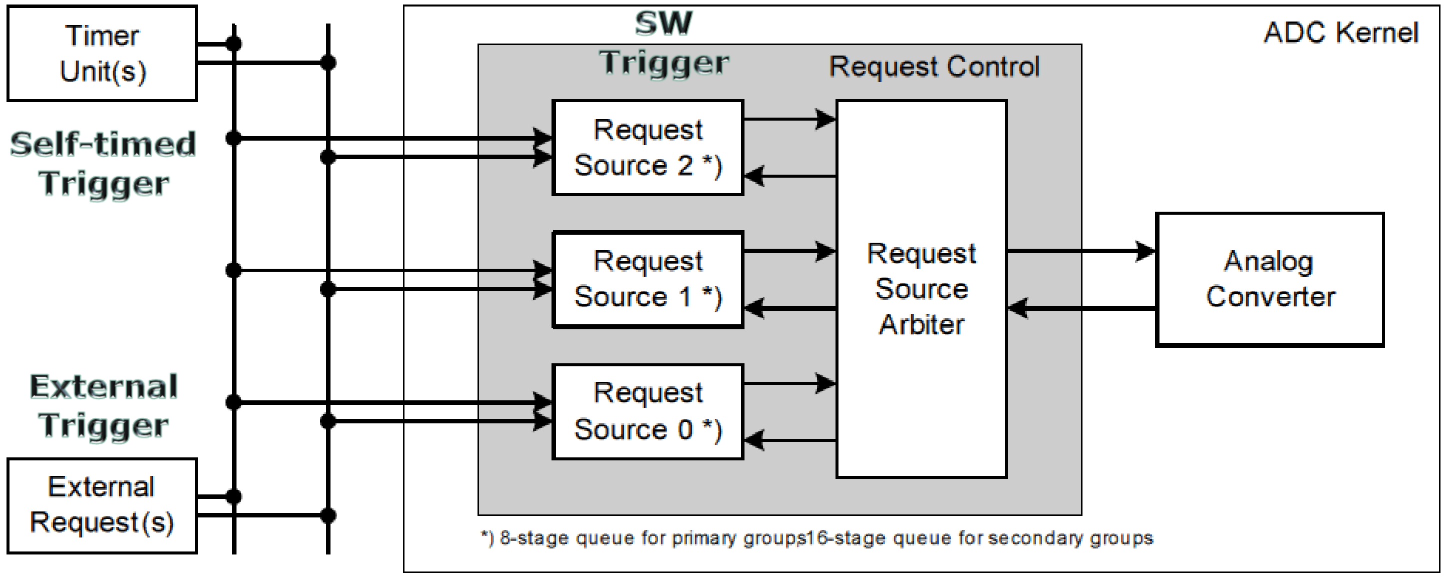
› **From SW point of view, the single cycle arbiter is compatible to the multi cycle arbiter of TC39x-A**

› **No change on the arbiter configuration registers**

› **Only sampling jitter has been resolved**

AURIX™ - TC3xx
EVADC – Conversion Request Generation

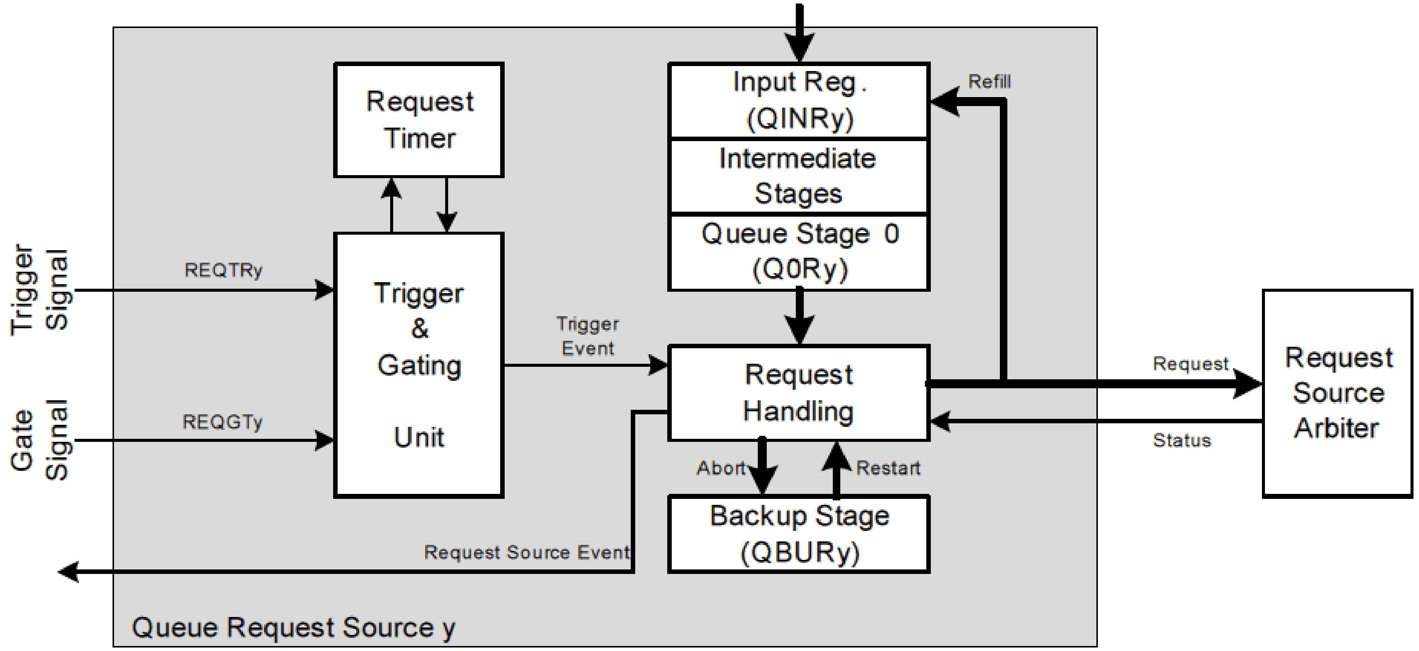
› Upon a trigger event, the request source requests the conversion of a certain analog input channel or a sequence of channels.
  - **Software triggers** directly activate the respective request source.
  - **Self-timed triggers** are generated by the request timer of the respective source.
  - **External triggers** synchronize the request source activation with external events, such as a trigger pulse from a timer generating a PWM signal or from a port pin.



AURIX™ - TC3xx
EVADC – 3x Queued Request Sources

› Each request source can operate in single-shot or in continuous mode:

- In **single-shot mode**, the programmed conversion (sequence) is requested once after being triggered. A subsequent conversion (sequence) must be triggered again.
- In **continuous mode**, the programmed conversion (sequence) is automatically requested repeatedly after being triggered once.

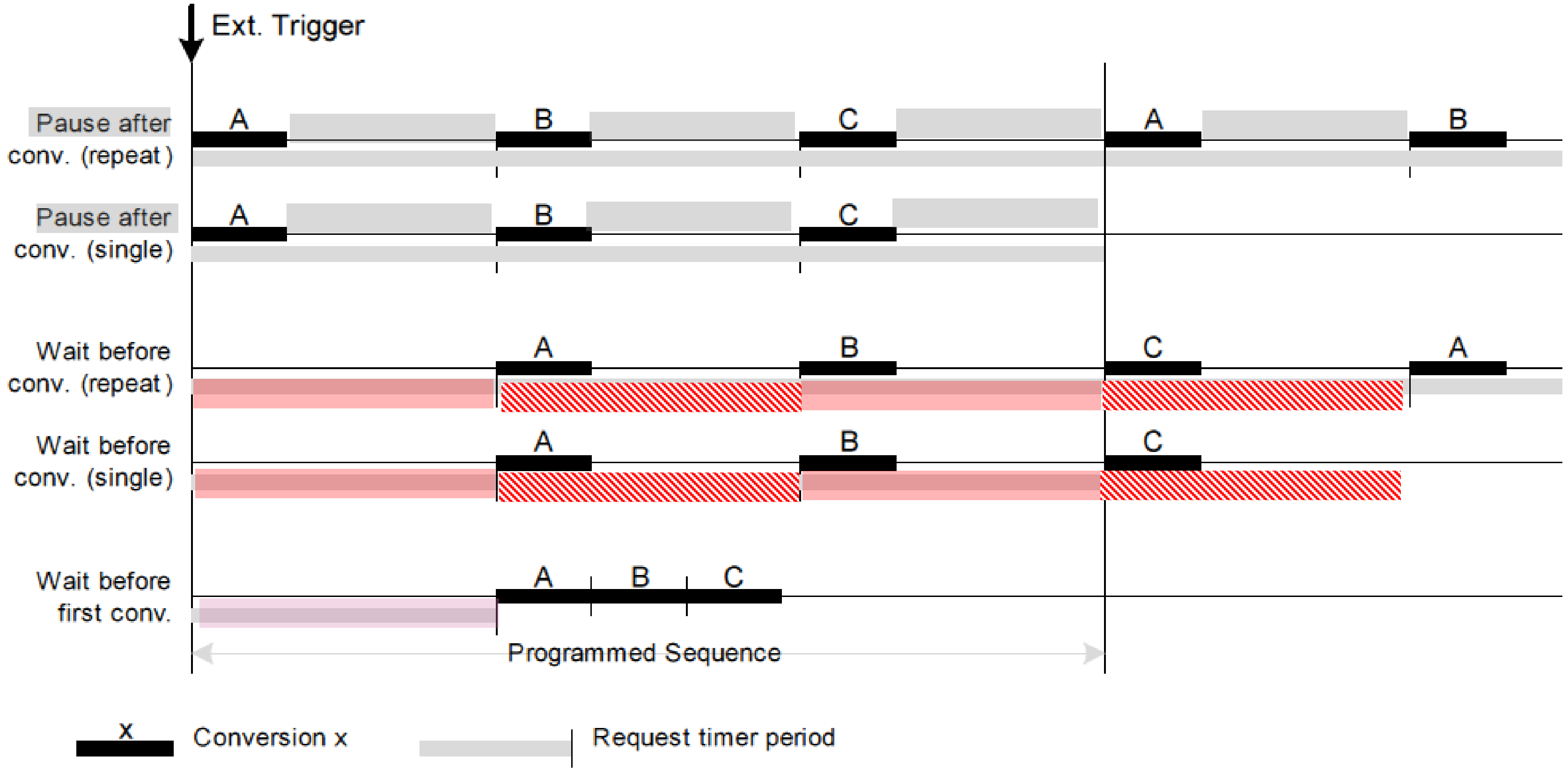


AURIX™ - TC3xx– Enhancement!
EVADC – Self-Timed Execution of Conversion Sequences

- The built-in request timer can request the conversions of a programmed sequence in configurable intervals.
  - The 16:1 prescaler generates a time base of 0.1 µs (for fADC = 160 MHz). The 10-bit timer creates intervals up to 102.4 µs.
- The timer is started by a trigger event (generated by hardware or by software), it is stopped when the queue runs empty. Several operating modes can be selected:
  - **Pause after each conversion**
  - **Wait before first conversion**
  - **Wait before each conversion**

AURIX™ - TC3xx– Enhancement!

EVADC – Self-Timed Execution of Conversion Sequences

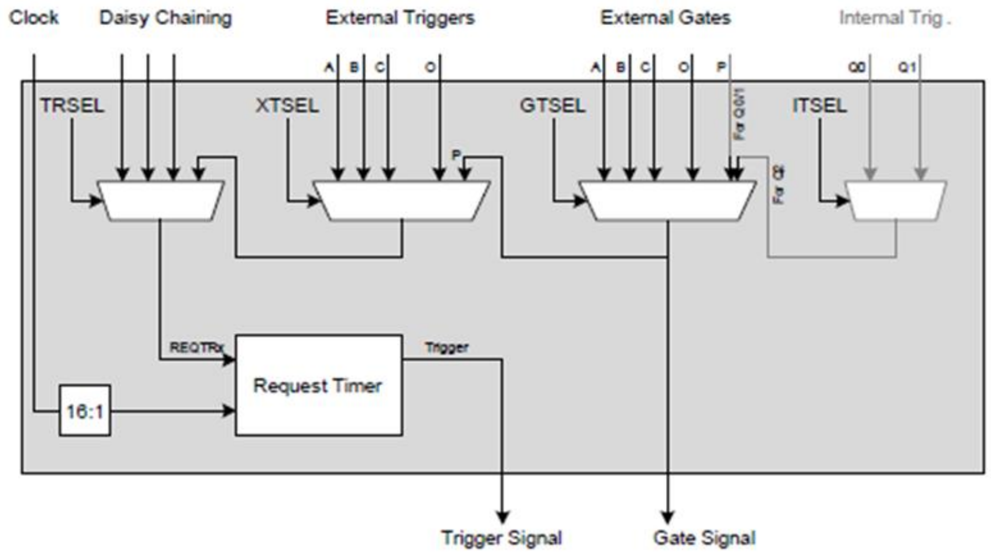


AURIX™ - TC3xx– Enhancement!

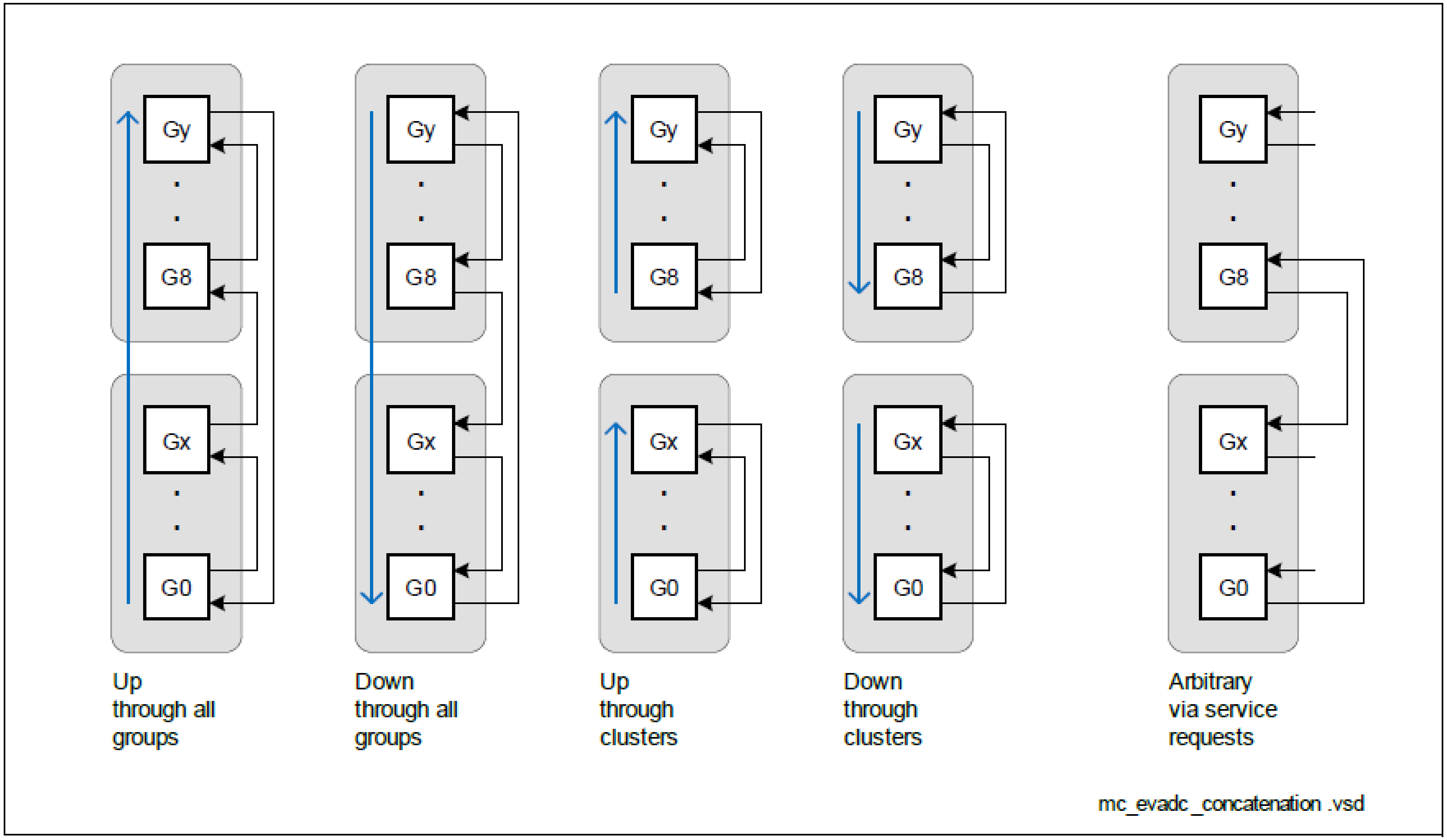
EVADC – Further Details for Request Source Queue

› **Extended Conversion Sequences through Concatenation**, Three concatenation modes are possible:

- **Within a group :** Request source 2 (Q2) can be internally triggered by Q0&Q1 of the same group.
- **Daisy chaining**: a request source can also be triggered by source events of an adjacent group. Daisy chains can be established within a cluster or spanning all groups
- **Multiple groups** : Triggers between groups can also be established by using the service request signal of another group.



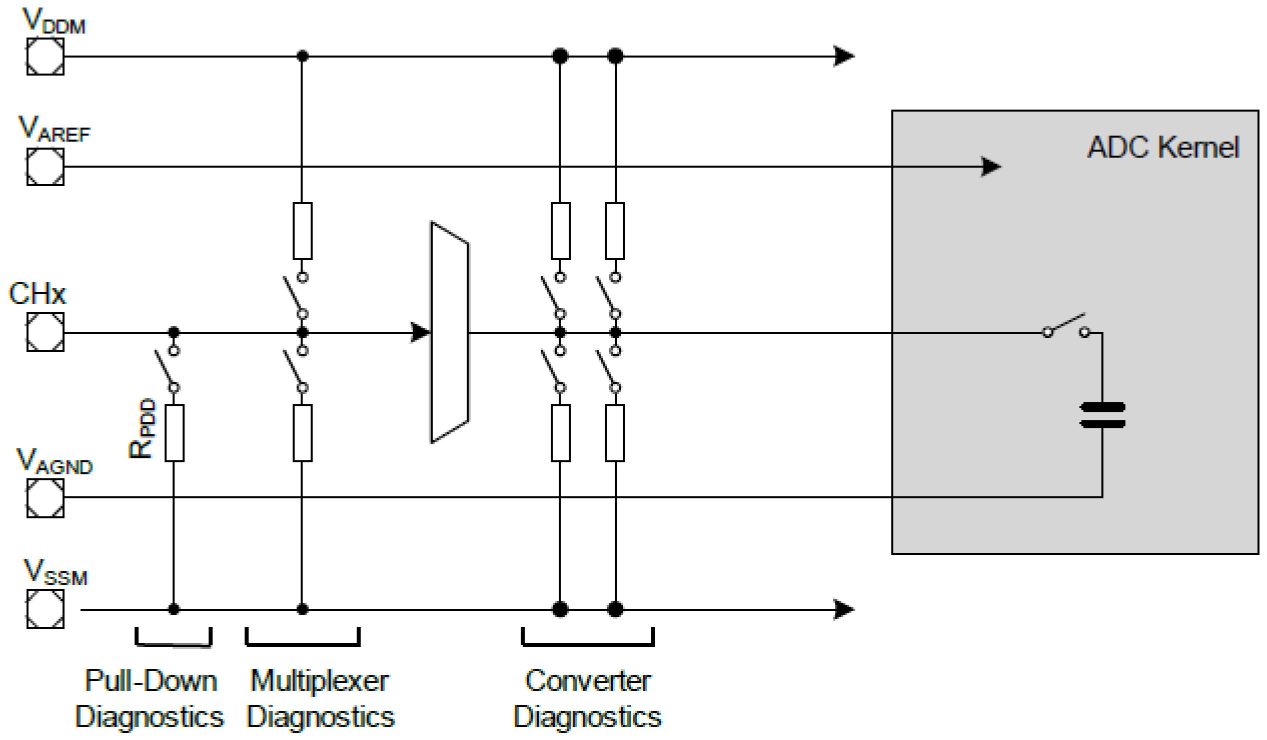
AURIX™ - TC3xx– Enhancement!
EVADC - Group Concatenation



AURIX™ - TC3xx– Safety Enhancement!

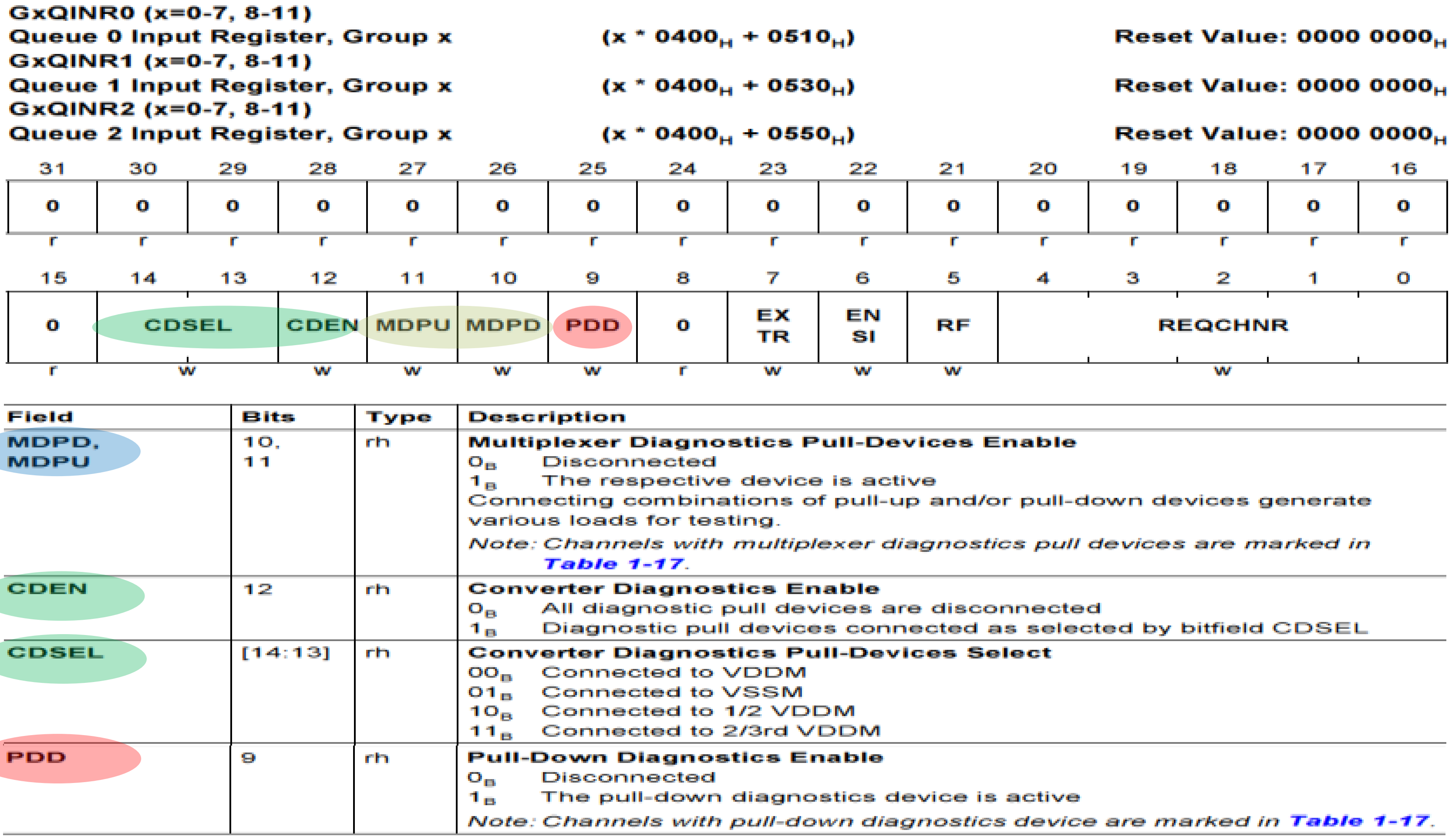
Safety feature – Analog Channel Diagnostics

- **Pull-Down Diagnostics** validates the connection of the external sensor
- **Multiplexer Diagnostics** validates the operation of the internal analog input multiplexer
- **Converter Diagnostics** validates the operation of the Analog/Digital converter itself
- **Broken Wire Detection** validates the connection from the sensor to the input pin
- **On-Chip Supervision Signals** enable additional monitoring



AURIX™ - TC3xx– Safety Enhancement! EVADC – SW Settings for Test Function

- **Testfunctions**
  Q2 can be used to automatically control safety-oriented test conversion sequences. It, therefore, provides the additional control bitfields CDSEL, CDEN, MDPU, MDPD, PDD, which are not available in Q0/Q1.



GxQINR0 (x=0-7, 8-11)
Queue 0 Input Register, Group x **(x * 0400_H + 0510_H)** **Reset Value: 0000 0000_H**
GxQINR1 (x=0-7, 8-11)
Queue 1 Input Register, Group x **(x * 0400_H + 0530_H)** **Reset Value: 0000 0000_H**
GxQINR2 (x=0-7, 8-11)
Queue 2 Input Register, Group x **(x * 0400_H + 0550_H)** **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	CDSEL		CDEN	MDPU	MDPD	PDD	0	EXTR	ENSI	RF	REQCHNR				
r	w		w	w	w	w	r	w	w	w	w				

Field	Bits	Type	Description
MDPD, MDPU	10, 11	rh	**Multiplexer Diagnostics Pull-Devices Enable** 0_B Disconnected 1_B The respective device is active Connecting combinations of pull-up and/or pull-down devices generate various loads for testing. *Note: Channels with multiplexer diagnostics pull devices are marked in Table 1-17.*
CDEN	12	rh	**Converter Diagnostics Enable** 0_B All diagnostic pull devices are disconnected 1_B Diagnostic pull devices connected as selected by bitfield CDSEL
CDSEL	[14:13]	rh	**Converter Diagnostics Pull-Devices Select** 00_B Connected to VDDM 01_B Connected to VSSM 10_B Connected to 1/2 VDDM 11_B Connected to 2/3rd VDDM
PDD	9	rh	**Pull-Down Diagnostics Enable** 0_B Disconnected 1_B The pull-down diagnostics device is active *Note: Channels with pull-down diagnostics device are marked in Table 1-17.*

1) Bitfields CDSEL, CDEN, MDPU, MDPD, PDD are only available in Q2, not in Q0/Q1.

AURIX™ - TC3xx– Safety Enhancement!

Safety feature – Channel Test Function Distribution

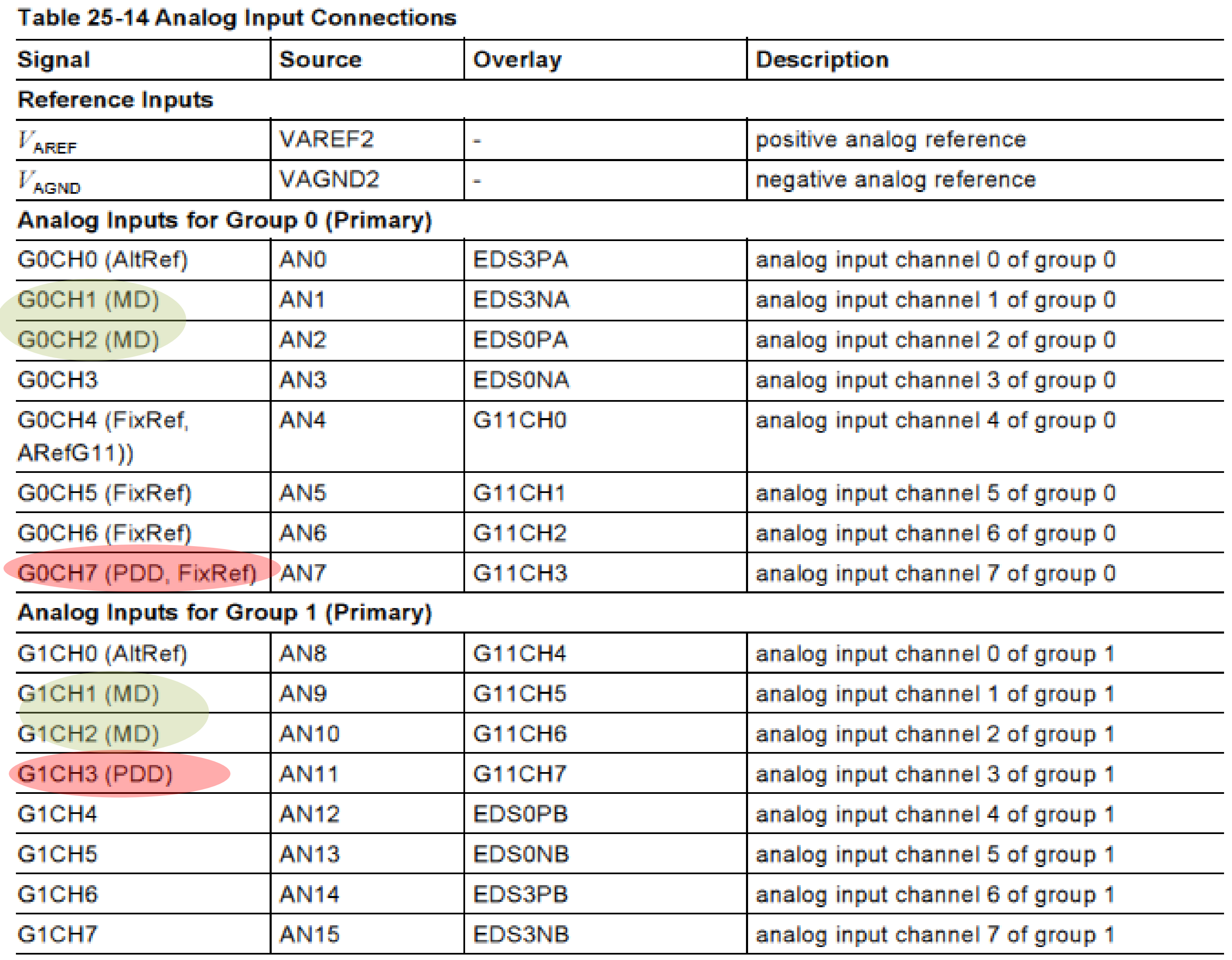


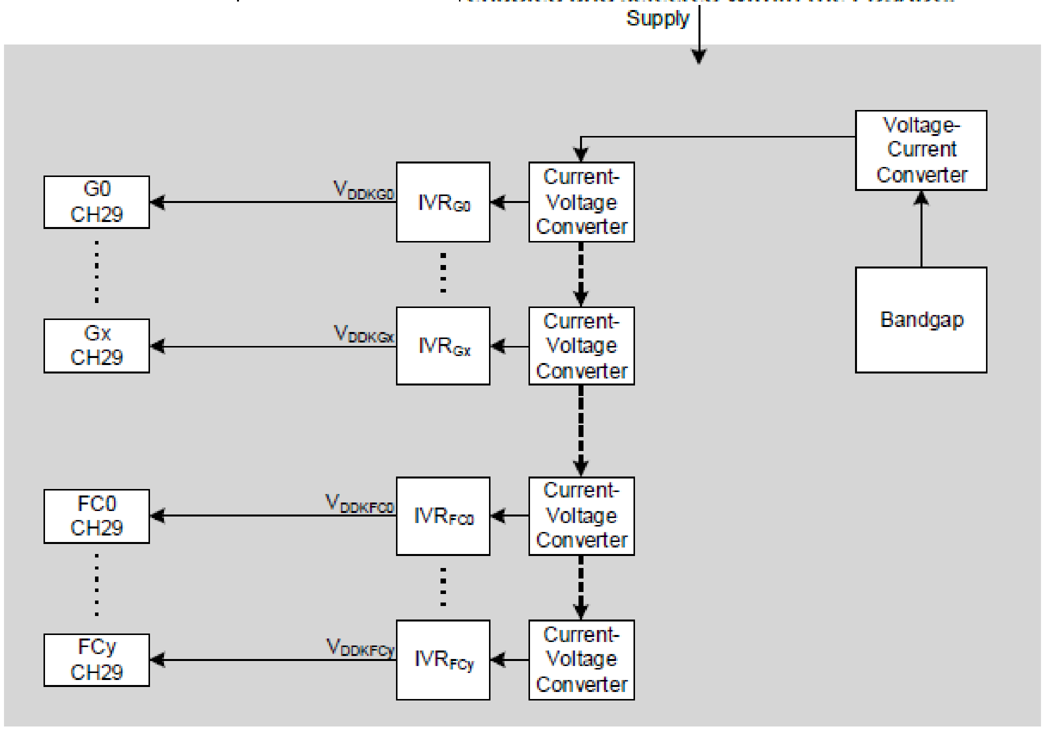
Table 25-14 Analog Input Connections

Signal	Source	Overlay	Description
Reference Inputs			
V_{AREF}	VAREF2	-	positive analog reference
V_{AGND}	VAGND2	-	negative analog reference
Analog Inputs for Group 0 (Primary)			
G0CH0 (AltRef)	AN0	EDS3PA	analog input channel 0 of group 0
G0CH1 (MD)	AN1	EDS3NA	analog input channel 1 of group 0
G0CH2 (MD)	AN2	EDS0PA	analog input channel 2 of group 0
G0CH3	AN3	EDS0NA	analog input channel 3 of group 0
G0CH4 (FixRef, ARefG11))	AN4	G11CH0	analog input channel 4 of group 0
G0CH5 (FixRef)	AN5	G11CH1	analog input channel 5 of group 0
G0CH6 (FixRef)	AN6	G11CH2	analog input channel 6 of group 0
G0CH7 (PDD, FixRef)	AN7	G11CH3	analog input channel 7 of group 0
Analog Inputs for Group 1 (Primary)			
G1CH0 (AltRef)	AN8	G11CH4	analog input channel 0 of group 1
G1CH1 (MD)	AN9	G11CH5	analog input channel 1 of group 1
G1CH2 (MD)	AN10	G11CH6	analog input channel 2 of group 1
G1CH3 (PDD)	AN11	G11CH7	analog input channel 3 of group 1
G1CH4	AN12	EDS0PB	analog input channel 4 of group 1
G1CH5	AN13	EDS0NB	analog input channel 5 of group 1
G1CH6	AN14	EDS3PB	analog input channel 6 of group 1
G1CH7	AN15	EDS3NB	analog input channel 7 of group 1

AURIX™ - TC3xx– Safety Enhancement!

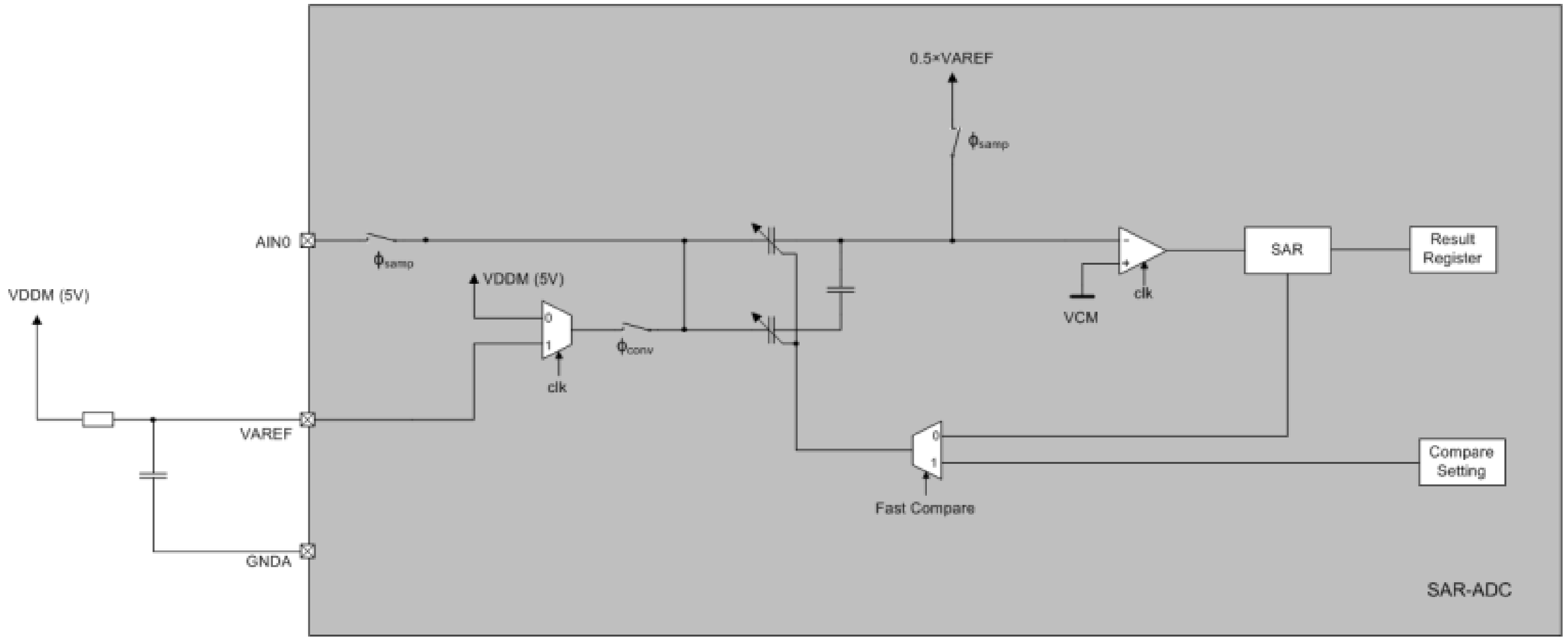
Safety feature – On-Chip Supervision Signals

Channel	Signal	Description
GxCH28[1)]	$V_{ANACOMM}$	Common reference signal, available to all converters. Can be fed to the converters through pin AN11.
GxCH29[1)]	V_{MTS}	Module test signal, provides the comparator supply voltage V_{DDK}, which is controlled by the bandgap in the power subsystem. See figure below.
GxCH30[1)]	V_{AGND}	Reference ground, external.
GxCH31[1)]	V_{AREF}	Reference voltage, external.
G10CH15	V_{EDSADC}	Supervision signal from module EDSADC. This supervision signal is enabled and selected within the EDSADC.[2)]



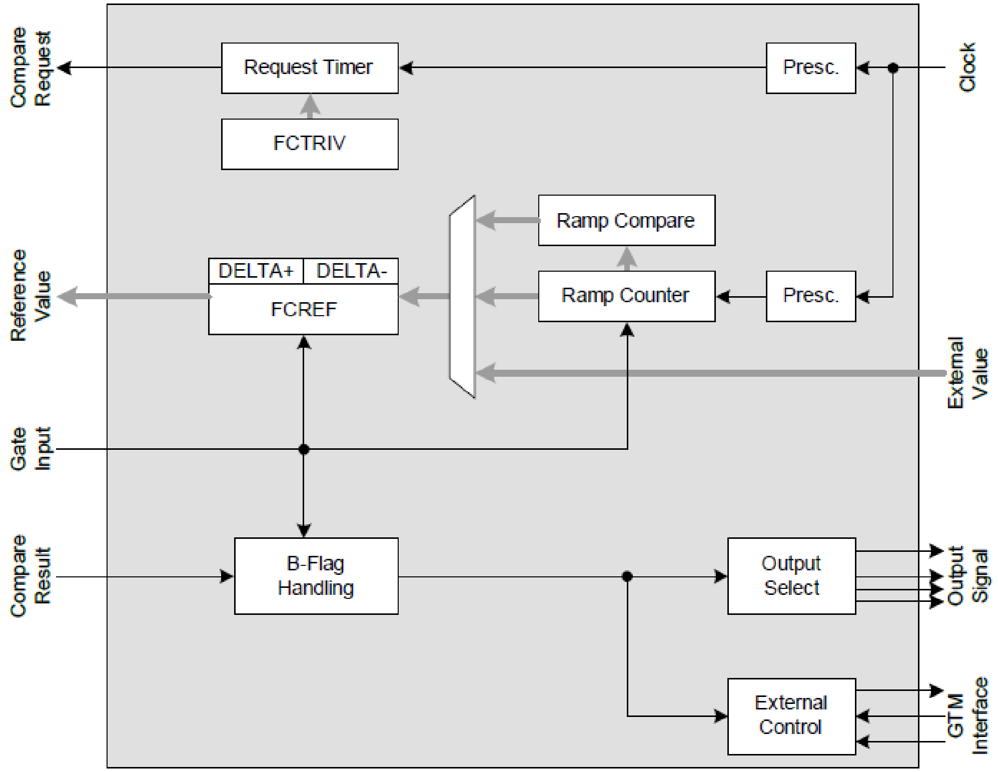
AURIX™ - TC3xx – New Feature Fast Compare Channel Operation

- Comparison time of 200ns (or **5Msamples/s**) which includes a sampling time of 100ns
  - therefore low impedance sensor output is required
- Compare value can be adjusted in a value range of 10 bit (**LSB10** …. or 1024 steps are supported)
- The intrinsic compare time is based on the 160 MHz clock
  - **Peak&Hold Operation** mode supported
  - Digital slope control to support **Peak Current controlled DC/DC** application



AURIX™ - TC3xx – New Feature Fast Compare – Channel Structure

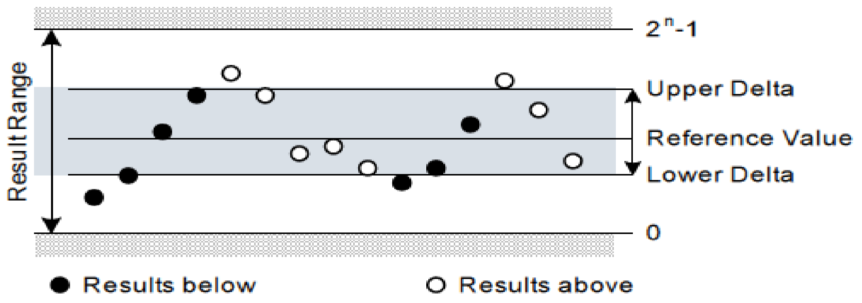
› A Fast Compare channel compares the input signal directly to a reference value stored in bitfield FCREF in register GxFCM (x = 12-19).
  – Delta values define a hysteresis.
  – This comparison returns a binary result (available in **bit FCR**) which indicates if the compared input voltage is above or below the given reference value.

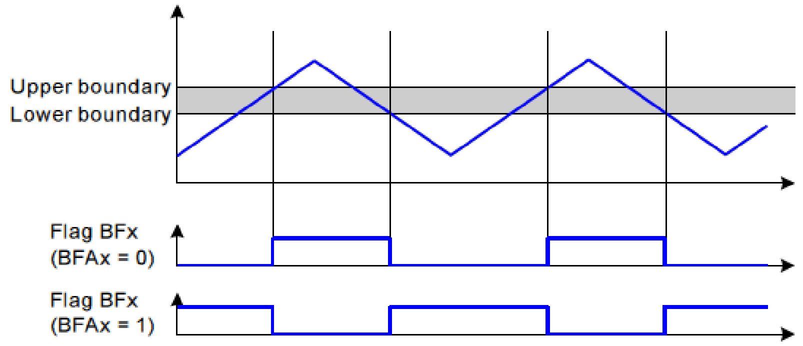


from Fast Compare Unit bit FCR

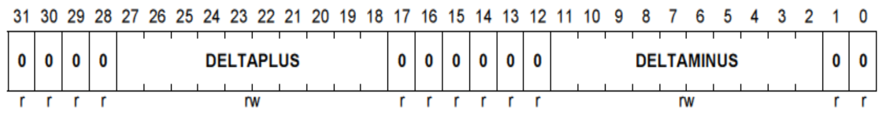
AURIX™ - TC3xx – New Feature
Fast Compare – Boundary Definition

- The actual boundaries are defined by the reference value **GxFCM** (x = 12-19).**FCREF** and the positive and negative delta values defining the hysteresis band.
- Bitfields **DELTAMINUS** and **DELTAPLUS** in register **GxFCHYST** (x = 12-19) store the delta values.
- The actual used compare value depends on the current result value FCR (see GxFCBFL (x = 12-19)):
  - FCR = 0: Reference value + upper delta (**FCREF** + **DELTAPLUS**)
  - FCR = 1: Reference value - lower delta (**FCREF** - **DELTAMINUS**)



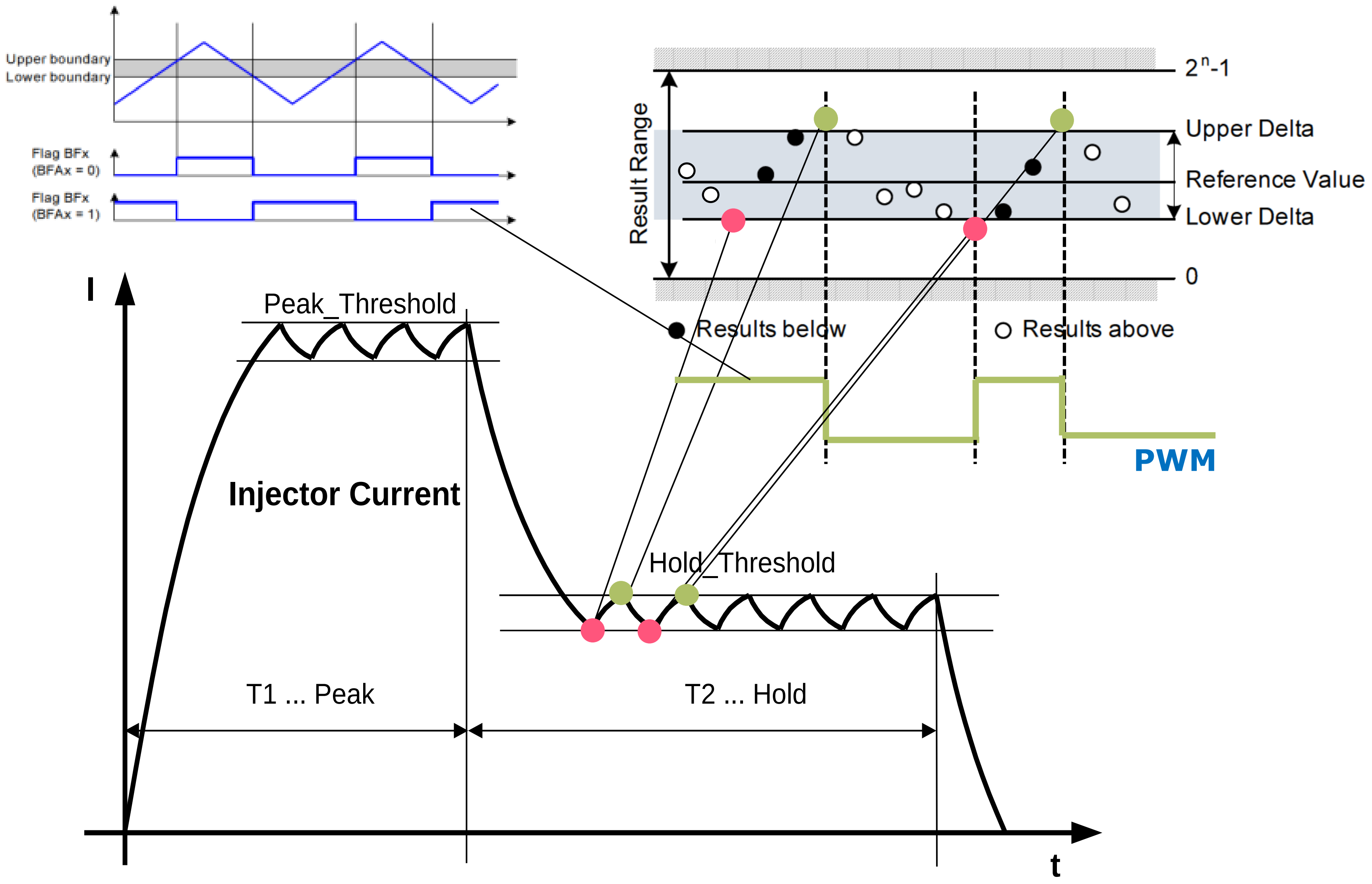


GxFCHYST (x = 12-19)
Fast Comp. Hysteresis Register, Group x (x * 0100_H + 2824_H) Reset Value: 0000 0000_H



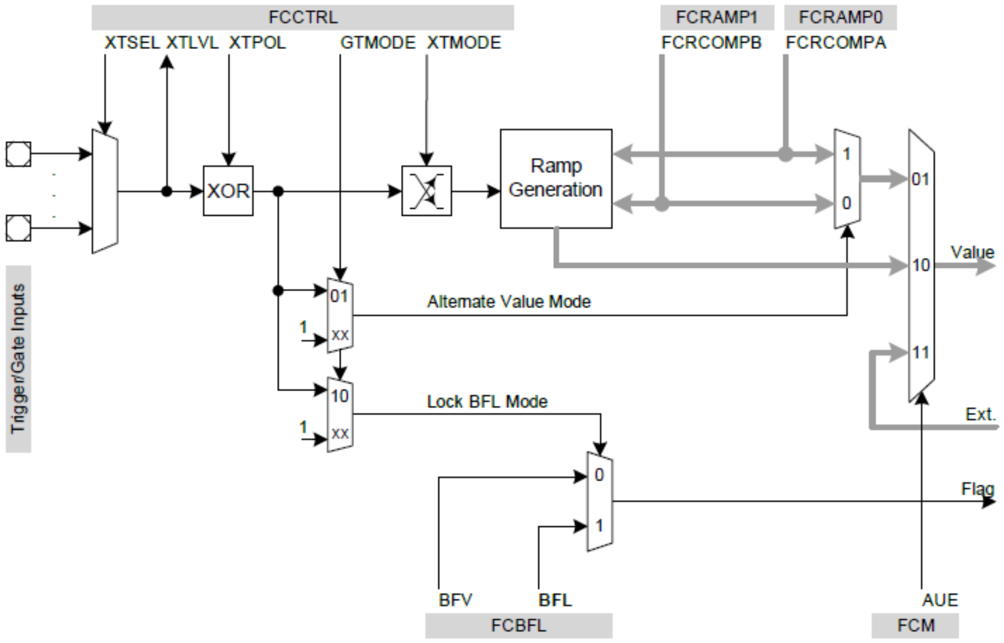
Field	Bits	Type	Description
0	[1:0]	r	**Reserved, write 0, read as 0** No flipflop required.
DELTAMINUS	[11:2]	rw	**Lower Delta Value** This value is subtracted from the reference value while the last result is 1
0	[17:12]	r	**Reserved, write 0, read as 0** No flipflop required.
DELTAPLUS	[27:18]	rw	**Upper Delta Value** This value is added to the reference value while the last result is 0
0	[31:28]	r	**Reserved, write 0, read as 0** No flipflop required.

AURIX™ - TC3xx – New Feature Fast Compare – Peak & Hold Operation (1)



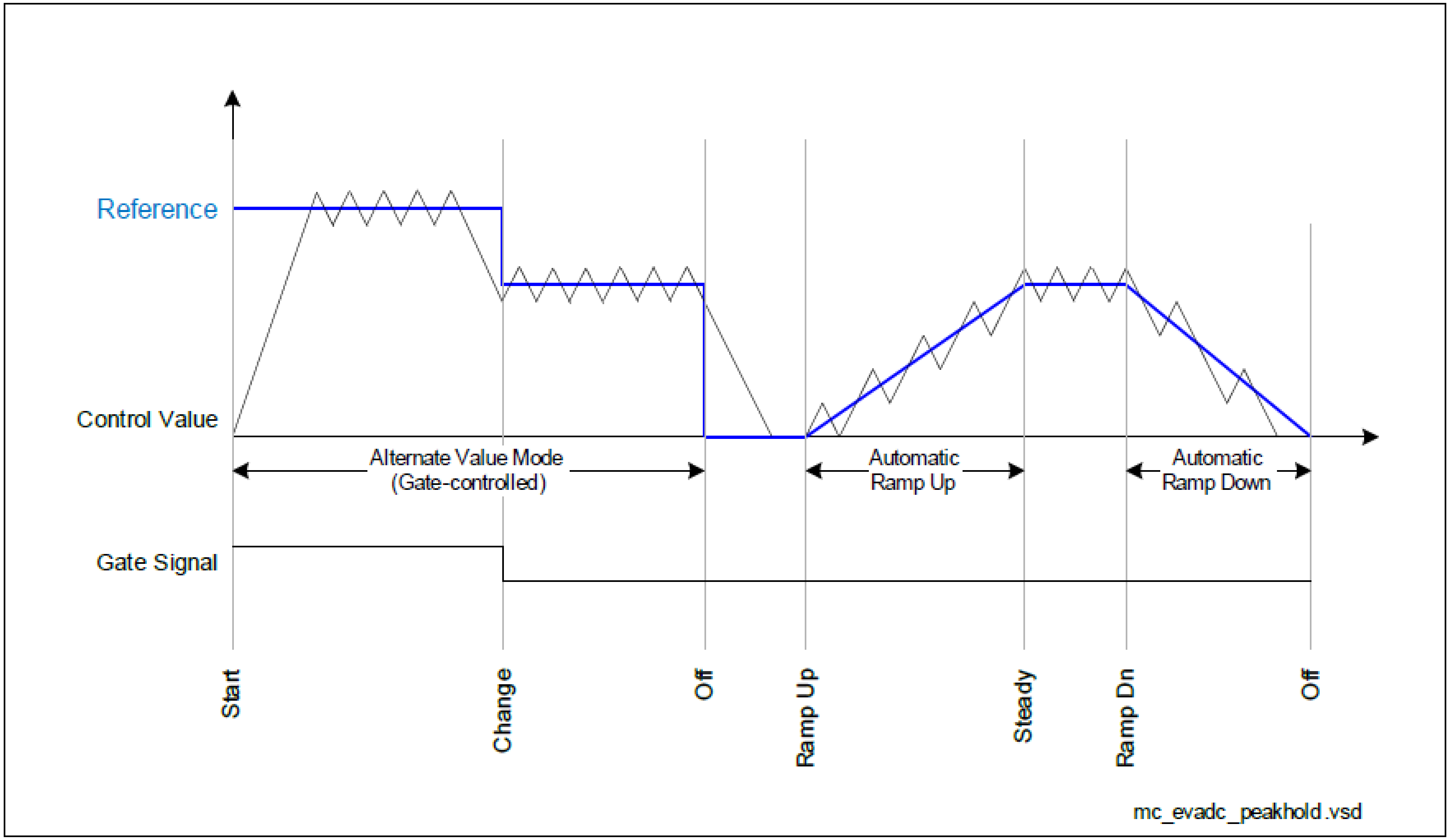
AURIX™ - TC3xx – New Feature Fast Compare – Peak & Hold Operation (2)

- The operation mode of Fast Compare channels is selected via bitfield AUE in register GxFCM (x = 12-19). four operating modes are available:
  - Software Mode
  - Alternate Value Mode
  - Ramp Mode
  - External Mode



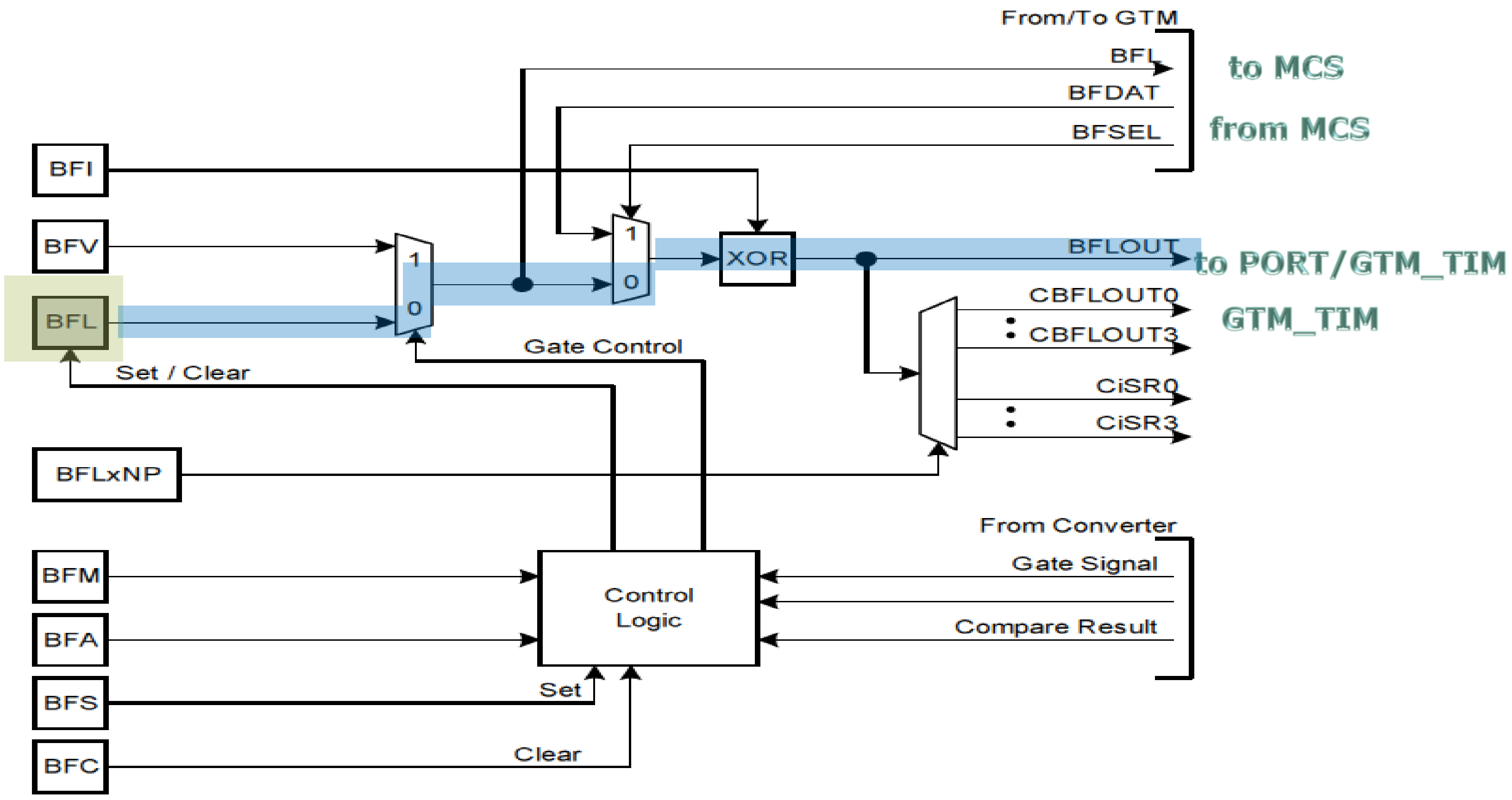
Field	Bits	Type	Description
AUE	19:18	rw	**Automatic Update Enable** Defines the source of the value(s) in bitfield FCREF. 00_B No automatic update value(s) written by software. 01_B Alternate value While gate is active (high): value copied from bitfield FCRCOMPA While gate is inactive (low): value copied from bitfield FCRCOMPB 10_B Ramp counter value(s) copied from ramp counter on ramp start or counter update. 11_B Analog source value(s) written by the associated converter (see product-specific appendix).

AURIX™ - TC3xx – New Feature
Fast Compare – Peak & Hold Operation (3)

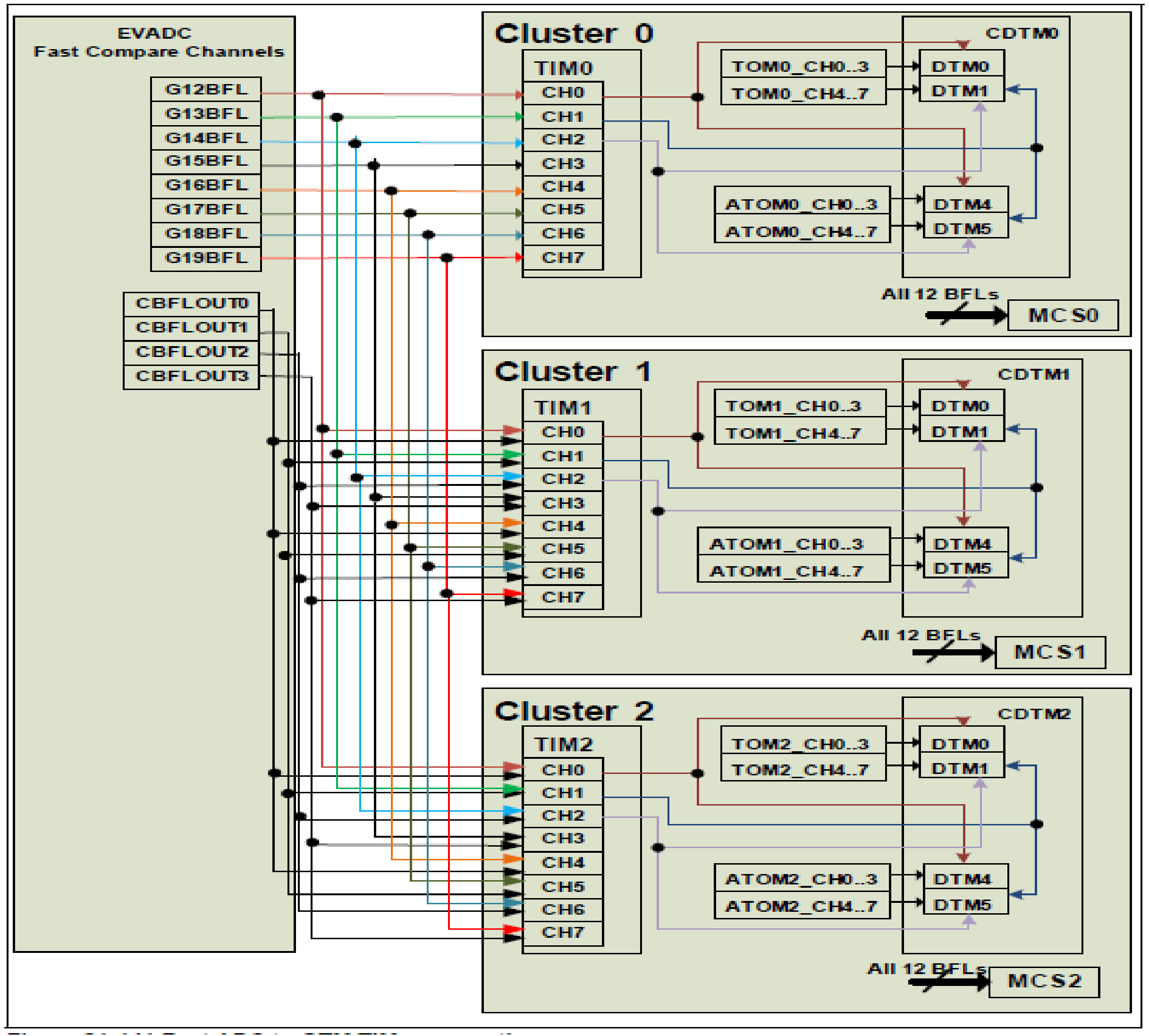


AURIX™ - TC3xx – New Feature Fast Compare – Boundary Flag Switching (1)

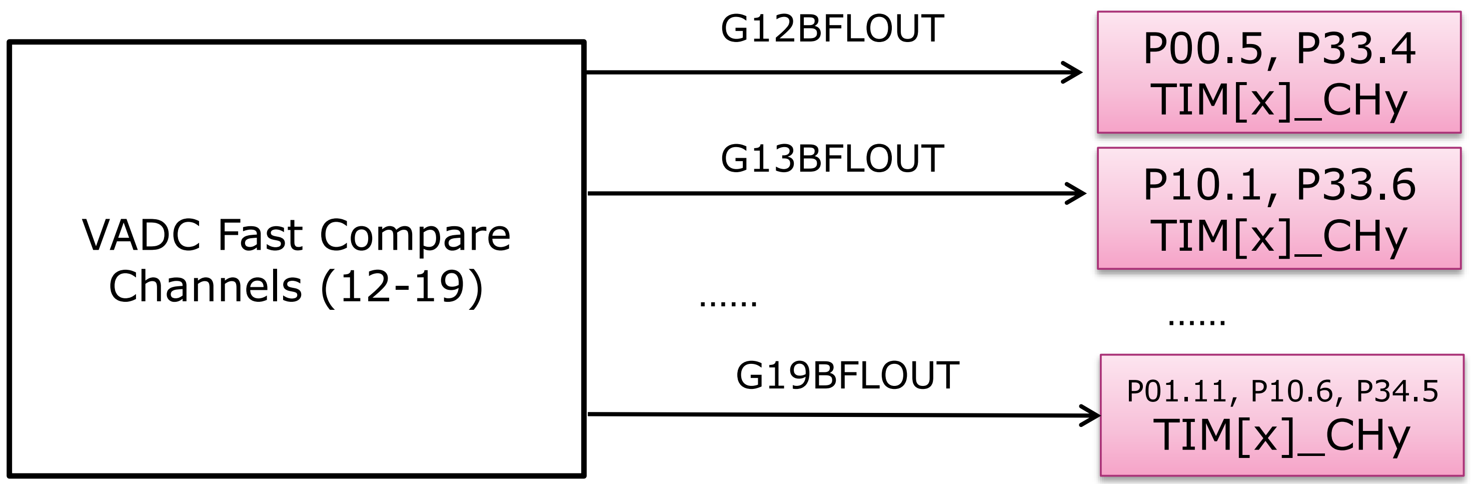
- The output signal derived from a boundary flag can be controlled by the GTM.
- A select input and a data input are available to temporarily replace the boundary flag signal before sending it to the output pin

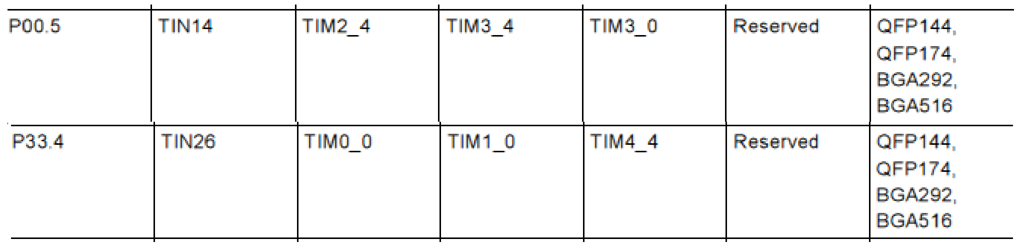


AURIX™ - TC3xx – New Feature Connections between EVADC and GTM

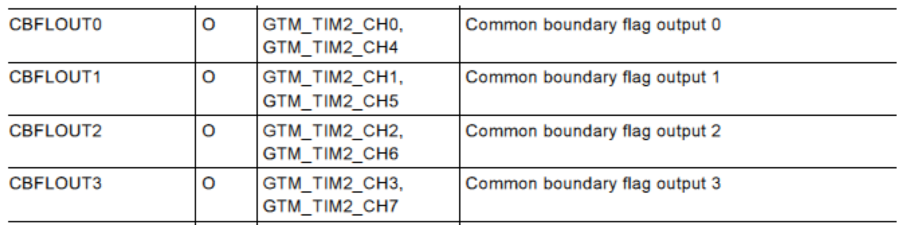


AURIX™ - TC3xx
Fast Compare to PORTS & GTM TIMx_CHy

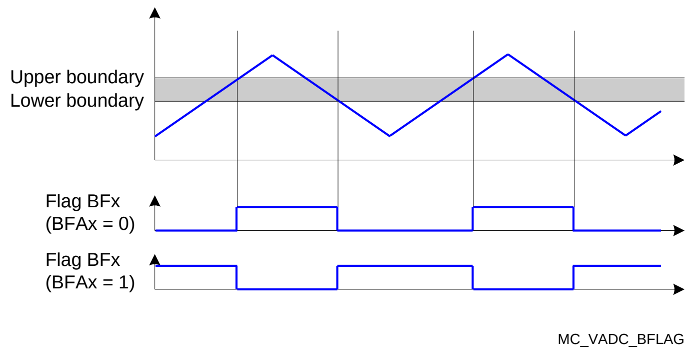




P00.5	TIN14	TIM2_4	TIM3_4	TIM3_0	Reserved	QFP144, QFP174, BGA292, BGA516
P33.4	TIN26	TIM0_0	TIM1_0	TIM4_4	Reserved	QFP144, QFP174, BGA292, BGA516



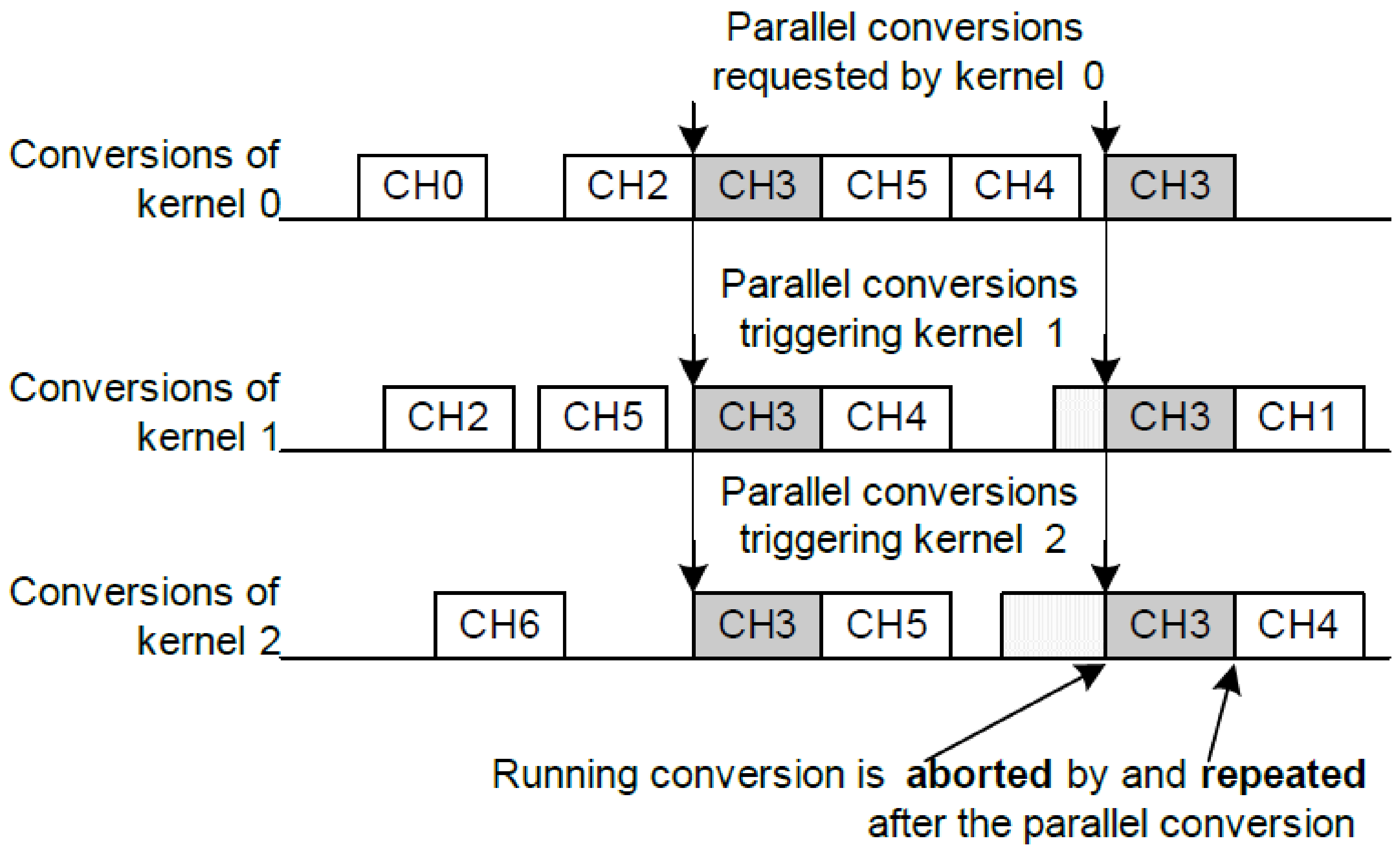
CBFLOUT0	O	GTM_TIM2_CH0, GTM_TIM2_CH4	Common boundary flag output 0
CBFLOUT1	O	GTM_TIM2_CH1, GTM_TIM2_CH5	Common boundary flag output 1
CBFLOUT2	O	GTM_TIM2_CH2, GTM_TIM2_CH6	Common boundary flag output 2
CBFLOUT3	O	GTM_TIM2_CH3, GTM_TIM2_CH7	Common boundary flag output 3



› Note: The BFL Values can be read directly by MCS using data exchange registers

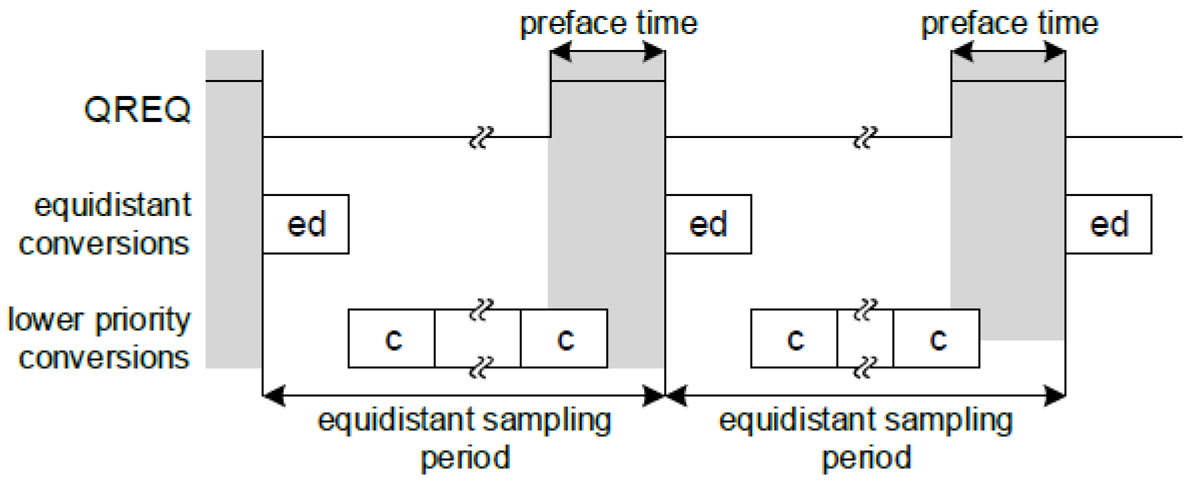
AURIX™ - TC3xx– Enhancement! Synchronization of Conversions (1)

› Synchronized Conversions for Parallel Sampling : Several independent ADC kernels1) can be synchronized for imultaneous measurements of analog input channels.


Parallel conversions requested by kernel 0
Conversions of kernel 0
CH0
CH2
CH3
CH5
CH4
CH3
Parallel conversions triggering kernel 1
Conversions of kernel 1
CH2
CH5
CH3
CH4
CH3
CH1
Parallel conversions triggering kernel 2
Conversions of kernel 2
CH6
CH3
CH5
CH3
CH4
Running conversion is **aborted** by and **repeated** after the parallel conversion

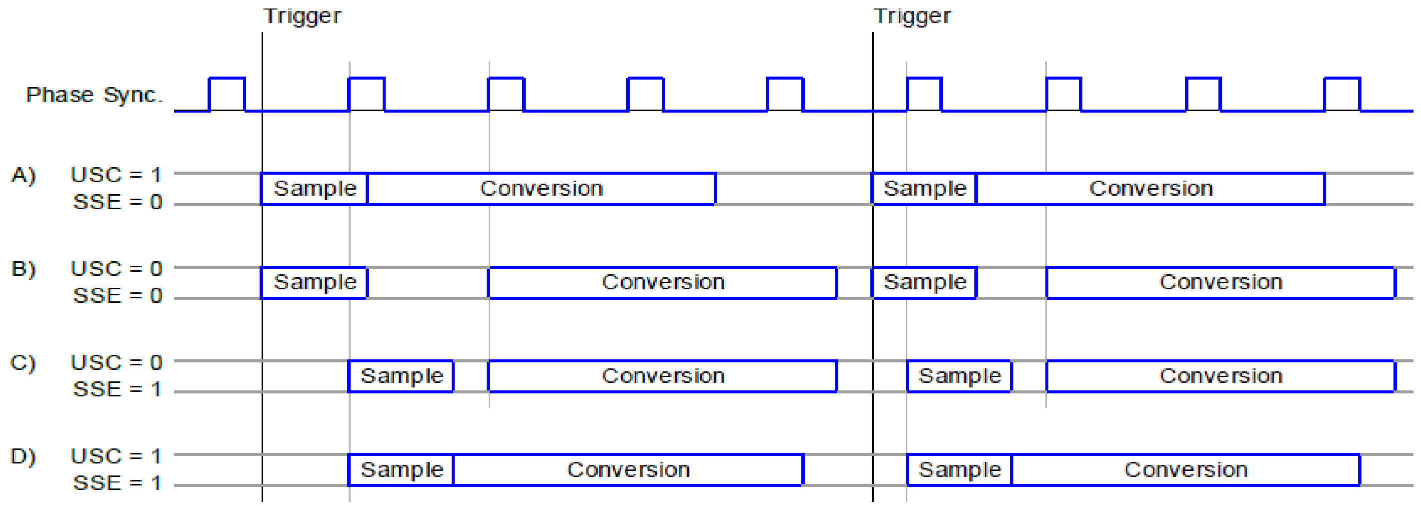
AURIX™ - TC3xx– Enhancement!
Synchronization of Conversions (2)

› Equidistant Sampling : conversions can be executed in a fixed timing raster. Conversions for equidistant sampling are triggered by a timer signal.


preface time
preface time
QREQ
equidistant conversions
ed
ed
ed
lower priority conversions
c
c
c
c
equidistant sampling period
equidistant sampling period

AURIX™ - TC3xx– Enhancement! Synchronization of Conversions (3)

› Synchronous Sampling

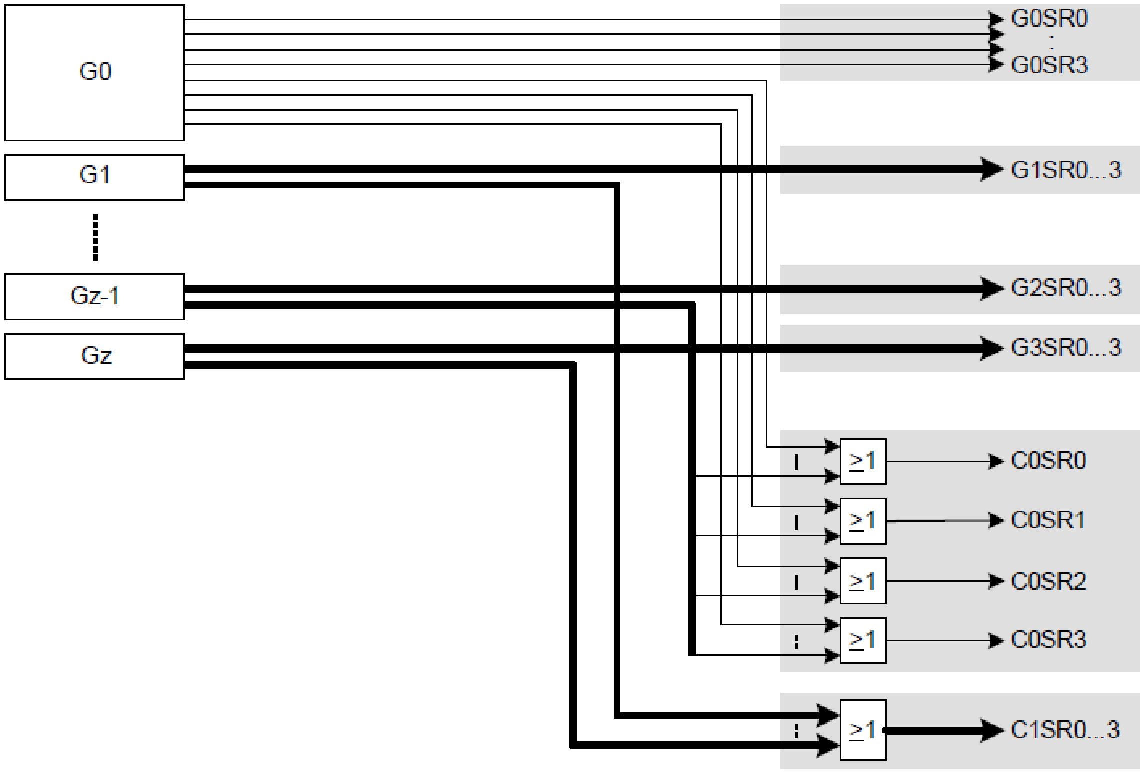


Example in Figure	Sampling Synchronization	Conversion Synchronization	Description	Note
A	SSE = 0, unsynchronized	USC = 1, unsynchronized	Sampling starts immed. after the trigger, conversion starts immed. after sampling	As in previous products
B [1)]	SSE = 0, unsynchronized	USC = 0, synchronized	Sampling starts immed. after the trigger, conversion starts upon next PHSYNC pulse	Default after Reset
C [2)]	SSE = 1, synchronized	USC = 0, synchronized	Sampling starts upon next PHSYNC pulse, conversion starts upon next PHSYNC pulse	Optimized synchronization
D	SSE = 1, synchronized	USC = 1, unsynchronized	Sampling starts upon next PHSYNC pulse, conversion starts immed. after sampling	Not recommended

AURIX™ - TC3xx Service Request Generation

Each A/D Converter can activate up to 4 group-specific service request output signals and up to 4 shared service request output signals to issue an interrupt or to trigger a DMA channel External Mode:

- **Request source events**: indicate that a request source completed the requested conversion sequence and the application software can initiate further actions.
- **Channel events**: indicate that a conversion is finished. Optionally, channel events can be restricted to result values within a programmable value range.
- **Result events**: indicate a new valid result in a result register.



AURIX™ - TC3xx
EVADC - use case 1

```
App_EvadcQueueTransfer g_EvadcQueueTransfer; /**< \brief Demo information */
void EvadcQueueTransferDemo_init(void)
{
  /* create configuration*/
   IfxEvadc_Adc_Config adcConfig;
   IfxEvadc_Adc_initModuleConfig(&adcConfig, &MODULE_EVADC);

   /* initialize module*/
   IfxEvadc_Adc_initModule(&g_EvadcQueueTransfer.evadc, &adcConfig);

   /* create group config*/
   IfxEvadc_Adc_GroupConfig adcGroupConfig;
   IfxEvadc_Adc_initGroupConfig(&adcGroupConfig, &g_EvadcQueueTransfer.evadc);

   /* with group 0 */
   adcGroupConfig.groupId = IfxEvadc_GroupId_0;
   adcGroupConfig.master  = adcGroupConfig.groupId;

   /* enable all queue source */
   adcGroupConfig.arbiter.requestSlotQueue0Enabled  = TRUE; // enable Queue0 mode
   adcGroupConfig.arbiter.requestSlotQueue1Enabled  = TRUE; // enable Queue1 mode
   adcGroupConfig.arbiter.requestSlotQueue2Enabled  = TRUE; // enable Queue2 mode

   /* enable all gates in "always" mode (no edge detection)*/
   adcGroupConfig.queueRequest[0].triggerConfig.gatingMode = IfxEvadc_GatingMode_always;
   adcGroupConfig.queueRequest[1].triggerConfig.gatingMode = IfxEvadc_GatingMode_always;
   adcGroupConfig.queueRequest[2].triggerConfig.gatingMode = IfxEvadc_GatingMode_always;

   /* initialize the group*/
   IfxEvadc_Adc_initGroup(&g_EvadcQueueTransfer.adcGroup, &adcGroupConfig);
}
```

EVADC module initialize

EVADC group initialize

AURIX™ - TC3xx
EVADC - use case 1(cont.)

```
void EvadcQueueTransferDemo_run(void)
{
        uint32                    chnIx;
        /* IMPORTANT: for deterministic results we have to disable the queue gate
         * while filling the queue, otherwise results could be output in the wrong order */
        IfxEvadc_RequestSource requestSource = IfxEvadc_RequestSource_queue0;
        IfxEvadc_GatingMode savedGate      = IfxEvadc_getQueueSlotGatingMode(g_EvadcQueueTransfer.adcGroup.group,requestSource);
        IfxEvadc_GatingSource gatingSource = IfxEvadc_getQueueSlotGatingSource(g_EvadcQueueTransfer.adcGroup.group, requestSource );
        /* create channel config */
        IfxEvadc_Adc_ChannelConfig adcChannelConfig[3];
        IfxEvadc_Adc_Channel       adcChannel[3];
        for (chnIx = 0; chnIx < 3; ++chnIx){
            IfxEvadc_Adc_initChannelConfig(&adcChannelConfig[chnIx], &g_EvadcQueueTransfer.adcGroup);
            adcChannelConfig[chnIx].channelId      = (IfxEvadc_ChannelId)(chnIx);
            adcChannelConfig[chnIx].resultRegister = IfxEvadc_ChannelResult_1; /* use result register #1 for all channels */
            /* initialize the channel */
            IfxEvadc_Adc_initChannel(&adcChannel[chnIx], &adcChannelConfig[chnIx]);
            /* Add channel to queue with refill enabled */
            IfxEvadc_Adc_addToQueue(&adcChannel[chnIx], requestSource,IFXEVADC_QUEUE_REFILL);
        }
        /* restore previous gate config */
        IfxEvadc_setQueueSlotGatingConfig(g_EvadcQueueTransfer.adcGroup.group, gatingSource, savedGate, requestSource );
        /* start the Queue */
        IfxEvadc_Adc_startQueue(&g_EvadcQueueTransfer.adcGroup,requestSource); /* the queue has already been started in previous test */
       /* get 10 results for all 3 channels and store in temporary buffer */
        Ifx_EVADC_G_RES resultTrace[3 * 10];
        uint32      i;
        for (i = 0; i < 3 * 10; ++i){
            unsigned    chnIx = i % 3;
            /* wait for valid result */
            Ifx_EVADC_G_RES conversionResult;
            do {
                conversionResult = IfxEvadc_Adc_getResult(&adcChannel[chnIx]);
            } while (!conversionResult.B.VF);
            /* store result */
            resultTrace[i] = conversionResult;
        }
        /* stop the queue */
        IfxEvadc_Adc_clearQueue(&g_EvadcQueueTransfer.adcGroup,requestSource);
}
```

EVADC channel initialize and add it to queque

SW trigger queque request

Read result

Part of your life. Part of tomorrow.