AN5695

Application note

GTM and ADC

Introduction

This document is intended to give information, method and guideline to use the GTM submodule as source trigger for the SAR or SD ADC.

1 Scope

This document describes all steps to configure and use the GTM submodules in conjunction with SAR or SD ADC, using the GTM as source trigger for the Analog IPs.

The devices under analysis are listed in Table 1.

Table 1. Device list

Device	Part Number
SPC574Kx	SPC574K70E5, SPC574K70E7, SPC574K72E5, SPC574K72E7
SPC572Lxx	SPC572L60F2, SPC572L60E3, SPC572L64F2, SPC572L64E3
SPC58xEx	SPC58EE80E7, SPC58EE80C3, SPC58EE84E7, SPC58EE84C3, SPC58NE80E7, SPC58NE80C3, SPC58NE84E7, SPC58NE84C3, SPC58NE84H0
SPC58xNx	SPC584N80E7, SPC584N80C3, SPC58EN80E7, SPC58EN80C3, SPC58EN84E7, SPC58EN84C3, SPC58NN84E7, SPC58NN84C3

The GTM-IP version are listed in Table 2.

Table 2. GTM-IP list

Device	Part Number	Specification Revision
SPC574Kx	GTM-IP 122	1.5.5.1
SPC572Lxx	GTM-IP 101	1.5.5.1
SPC58xEx	GTM-IP 343	3.1.5.1
SPC58xNx	GTM-IP 344	3.1.5.1

2 Overview

The Successive Approximation Register Analog-to-Digital Converter (SAR ADC) are a versatile class of analog-to-digital converters that produce a digital (discrete time) representation of a continuous analog waveform providing accurate and fast conversion data for a wide range of applications.

Each SAR ADC analog block has a corresponding digital interface.

The ADC converter are used in an increasing number of automotive applications, from simply monitoring of DC voltages to most complex application following the innovation in the automotive area.

3 ADC converter

This document describes the ADC converter for the SPC574Kx MCU.

SPC574Kx contains the following analog-to-digital converters (ADCs):

- 5 independent 12-bit Successive Approximation Register Analog-to-Digital Converters (SAR ADCs)
- 2 independent 16-bit Sigma-Delta Analog-to-Digital Converters (Sigma-Delta, or SDA DCs)

These ADCs interface to:

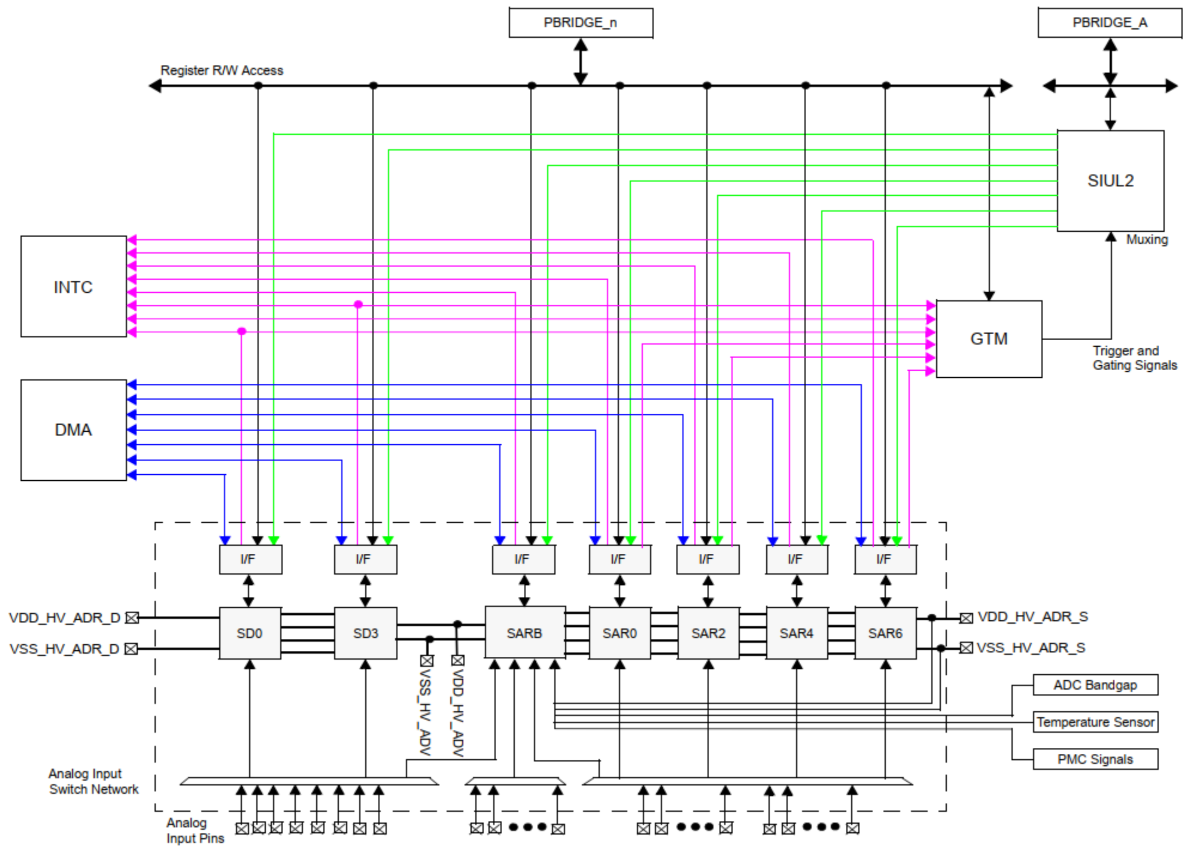
- 38 external analog input pins in the 144-pin package
- 48 external analog input pins in the 176-pin QFP package

Each SAR or Sigma-Delta ADC instance has independent analog input pin multiplexing, register controls, trigger and clock inputs, independent interrupt and DMA capability and a dedicated peripheral bridge (PBRIDGE) interface for CPU access to the registers.

3.1 ADC subsystem block diagram

The Figure 1 gives the block diagram for the high-level integration of the SPC574Kx ADC subsystem for, demonstrating how each SAR or Sigma-Delta ADC has independent register interfaces, interrupt/DMA requests, and trigger source selection.

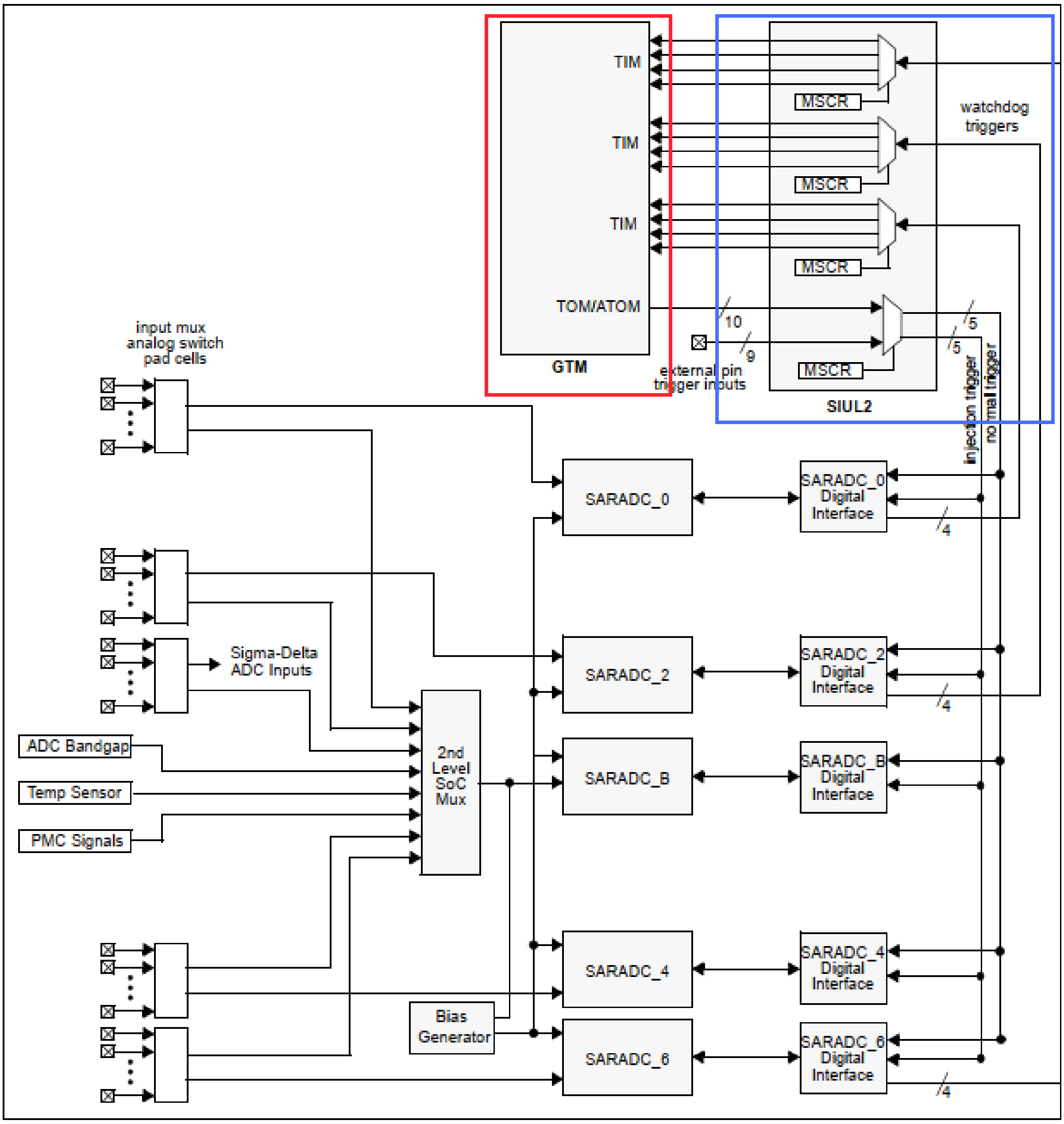
Figure 1. SPC574Kx ADC subsystem



3.1.1 SAR ADC integration diagram

The Figure 2 shows the block diagram of the integration of the five SAR ADCs on SPC574Kx.

Figure 2. SAR ADC Integration diagram



GTM and its submodules is fully integrated with ADC, in the GTM → ADC direction and vice versa.

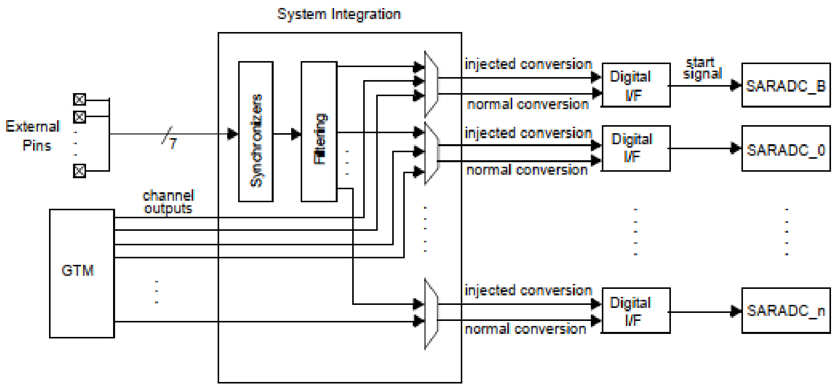
3.1.2 SAR ADC triggering

Each SAR ADC has two independent trigger inputs for starting conversion on an analog input: one is related to injected conversion and the other is related to normal conversion.

The trigger sources for each SAR ADC on the device are either external input pins or GTM channels.

Each external input can selectively trigger any of the five SAR ADCs on the device.

Figure 3. SAR ADC trigger diagram



4 GTM and ADC

GTM submodule can be used to trigger a start conversion of one ADC converter.

This is useful in an automotive application because it can be executed from MCS subsystem due to a specific event.

For example, in an Engine motor control application, an event could be a specific crank angular position.

The benefit of this architecture is that the CPU is not involved, giving more performance result, resource allocation and speed.

The first step is detecting the pins, in the Input Multiplexing table, with the ADC triggering capability.

Table 3. Input trigger

Instance	Input	SIUL2 MSCR Register	Source Signal Select (MSCR[SSS])	Source Instance	Source Signal
GTM	TIM0_0	512	0010_0000	SARADC_0	WATCHDOG_LIMIT_TRIGGER 3
GTM	TIM0_1	513	0010_0000	SARADC_2	WATCHDOG_LIMIT_TRIGGER 3
GTM	TIM0_2	514	0010_0000	SARADC_6	WATCHDOG_LIMIT_TRIGGER 3
GTM	TIM0_3	515	0010_0000	SARADC_0	WATCHDOG_LIMIT_TRIGGER 0
GTM	TIM0_4	516	0010_0000	SARADC_2	WATCHDOG_LIMIT_TRIGGER 0
GTM	TIM0_5	517	0010_0000	SARADC_6	WATCHDOG_LIMIT_TRIGGER 0
GTM	TIM1_0	520	0010_0000	SARADC_0	WATCHDOG_LIMIT_TRIGGER 3
GTM	TIM1_1	521	0010_0000	SARADC_2	WATCHDOG_LIMIT_TRIGGER 3
GTM	TIM1_2	522	0010_0000	SARADC_6	WATCHDOG_LIMIT_TRIGGER 3
GTM	TIM1_3	523	0010_0000	SARADC_0	WATCHDOG_LIMIT_TRIGGER 1
GTM	TIM1_4	524	0010_0000	SARADC_2	WATCHDOG_LIMIT_TRIGGER 1
GTM	TIM1_5	525	0010_0000	SARADC_6	WATCHDOG_LIMIT_TRIGGER 1
GTM	TIM2_0	528	0010_0000	SARADC_0	WATCHDOG_LIMIT_TRIGGER 2
GTM	TIM2_1	529	0010_0000	SARADC_2	WATCHDOG_LIMIT_TRIGGER 2
GTM	TIM2_2	530	0010_0000	SARADC_6	WATCHDOG_LIMIT_TRIGGER 2
GTM	TIM2_3	531	0010_0000	SARADC_0	WATCHDOG_LIMIT_TRIGGER 2
GTM	TIM2_4	532	0010_0000	SARADC_2	WATCHDOG_LIMIT_TRIGGER 2
GTM	TIM2_5	533	0010_0000	SARADC_6	WATCHDOG_LIMIT_TRIGGER 2

GTM triggering to SAR ADC:

Figure 4. ATOM & TOM trigger start

Instance	Input	SIUL2 MSCR Register	Source Signal Select (MSCR[SSS])	Source Instance	Source Signa
SARADC_0	IND_START_TRG	624	0001_0000	GTM	TOM0_6
SARADC_0	IND_START_TRG	624	0001_0001	GTM	TOM0_7
SARADC_0	IND_START_TRG	624	0001_0010	GTM	TOM0_13
SARADC_0	IND_START_TRG	624	0001_0011	GTM	TOM0_14
SARADC_0	IND_START_TRG	624	0001_0100	GTM	ATOM0_4
SARADC_0	IND_START_TRG	624	0001_0101	GTM	ATOM0_5
SARADC_0	IND_START_TRG	624	0001_0110	GTM	ATOM0_6
SARADC_0	IND_START_TRG	624	0001_0111	GTM	ATOM0_7
SARADC_0	IND_START_TRG	624	0001_1000	GTM	TOM1_6
SARADC_0	IND_START_TRG	624	0001_1001	GTM	TOM1_7
SARADC_0	IND_START_TRG	624	0001_1010	GTM	TOM1_13
SARADC_0	IND_START_TRG	624	0001_1011	GTM	TOM1_14
SARADC_0	IND_START_TRG	624	0001_1100	GTM	ATOM1_4
SARADC_0	IND_START_TRG	624	0001_1101	GTM	ATOM1_5
SARADC_0	IND_START_TRG	624	0001_1110	GTM	ATOM1_6
SARADC_0	IND_START_TRG	624	0001_1111	GTM	ATOM1_7
SARADC_2	IND_START_TRG	626	0001_0000	GTM	TOM0_6
SARADC_2	IND_START_TRG	626	0001_0001	GTM	TOM0_7
SARADC_2	IND_START_TRG	626	0001_0010	GTM	TOM0_13
SARADC_2	IND_START_TRG	626	0001_0011	GTM	TOM0_14
SARADC_2	IND_START_TRG	626	0001_0100	GTM	ATOM0_4
SARADC_2	IND_START_TRG	626	0001_0101	GTM	ATOM0_5
SARADC_2	IND_START_TRG	626	0001_0110	GTM	ATOM0_6
SARADC_2	IND_START_TRG	626	0001_0111	GTM	ATOM0_7
SARADC_2	IND_START_TRG	626	0001_1000	GTM	TOM1_6
SARADC_2	IND_START_TRG	626	0001_1001	GTM	TOM1_7
SARADC_2	IND_START_TRG	626	0001_1010	GTM	TOM1_13
SARADC_2	IND_START_TRG	626	0001_1011	GTM	TOM1_14
SARADC_2	IND_START_TRG	626	0001_1100	GTM	ATOM1_4

Title | IO Signal Table | **Input Muxing** | Supply Pins | Misc Pins | QFP176 | FQ216

*Note: Filter the Input Muxing table about SAR ADC **Instance** and TOM/ATOM **Source Signal**.*

4.1 Example application

For the example application will be used the SPC5Studio.

Platform used will be SPC574KE5 144 pin.

Details:

- SARADCB ANP[11]
- DMA Channel 3

Figure 5. DMAMUX peripheral DMA request to input source mapping

Source	Peripheral DMA requests			
	DMAMUX_0	DMAMUX_1	DMAMUX_2	DMAMUX_3
0	Reserved	Reserved	Reserved	Reserved
1	ADC_SAR_0 EOC	Reserved	ADC_SAR_2 EOC	Reserved
2	Reserved	Reserved	Reserved	DSPI_2 RX
3	ADC_SAR_B EOC	LINFlex_0 RX	DSPI_1 RX	DSPI_2 TX
4	ADC_SD_0 EOC	LINFlex_0 TX	DSPI_1 TX	LINFlex_2 RX

- ATOM0 Channel4 as ADC trigger

Table 4. SARADC_B Input Muxing

Instance	Input	SIUL2 MSCR Register	Source Signal Select (MSCR[SSS])	Source Instance	Source Signal
SARADC_B	IND_START_TRG	639	0001_0100	GTM	ATOM0_4

*Note: These values must be set in SPC5Studio Board Initialization Component, **Special Mux Settings**.*

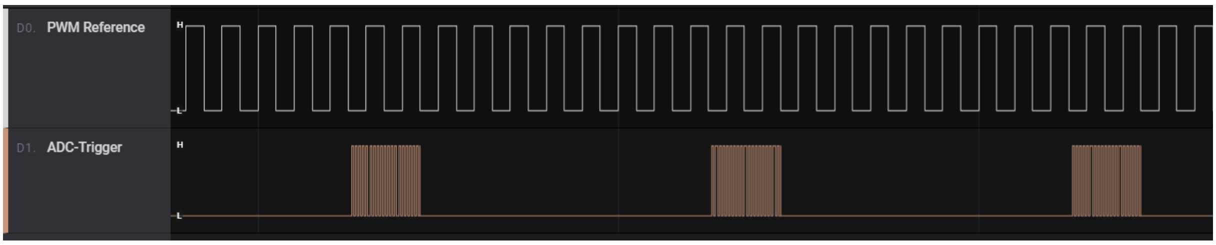
The application uses two different PWM.

- *ATOM0 Channel 0*, used as reference PWM to simulate external events. The PWM is routed internally with a TIM channel. Typically, the TIM is used in an Engine application to trigger events managed by the MCS. TIM channel is configured to raise interrupt each falling edge. These edges are notified via ARU to the MCS code.
- *ATOM0 Channel 4*, used for the ADC trigger. The ATOM channel starts its execution (in the MCS code) at 10th Falling Edge caught in the MCS code via ARU. Setting period, clock and duty parameters a PWM of *1ms* is generated. At each Rising Edge, the ADC perform one conversion.

End of Conversion is notified to the CPU in the DMA call back.

After 2 PWM the ADC trigger is stopped (*ATOM0 Channel 4* reset) by CPU.

Figure 6. PWM and ADC trigger



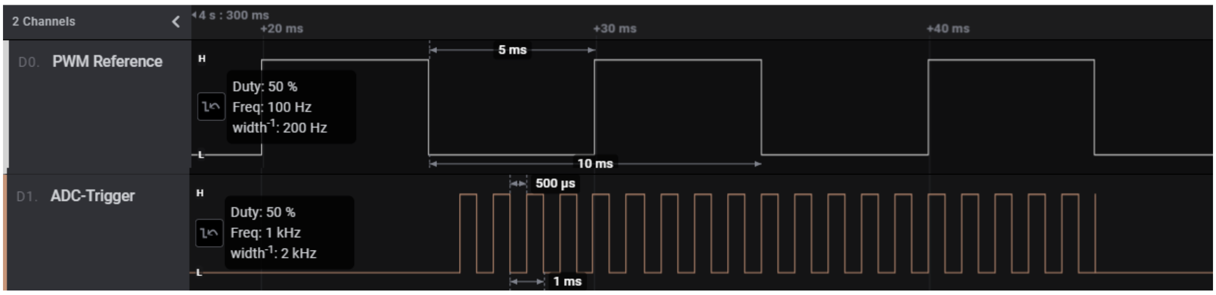


Figure 7. PWM and ADC trigger detail

4.2 SPC5Studio configuration

Component to configure:

- GTM
  - PWM - TIM - MCS
- Low Level Driver
  - SAR ADC
- Board
  - Pin definition

4.2.1 GTM configuration

Figure 8. GTM modules

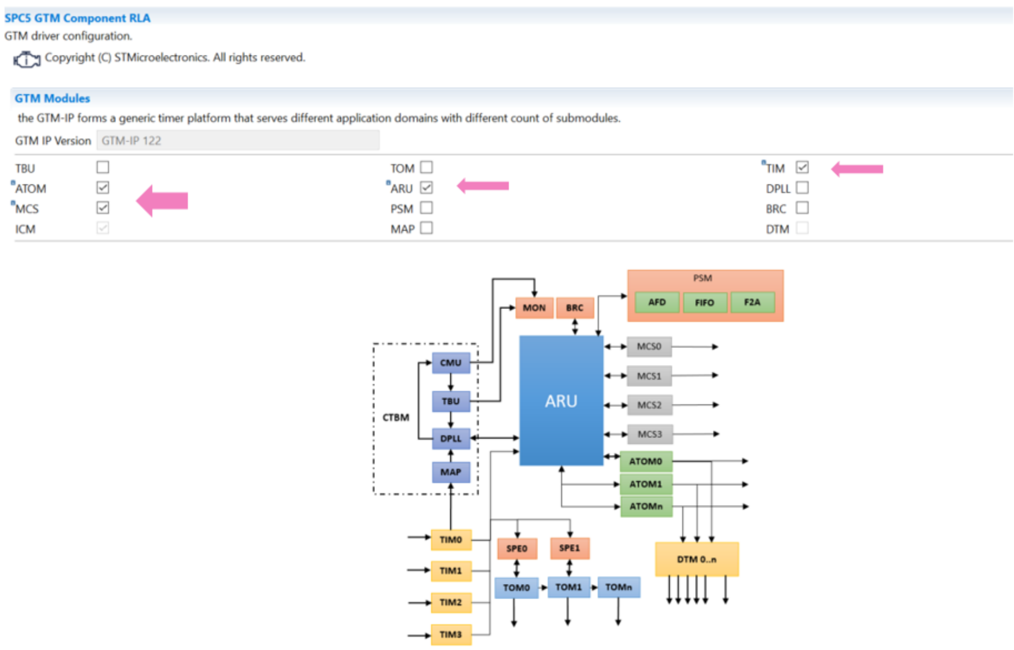


Figure 9. ATOM0 CH0 - REF. PWM

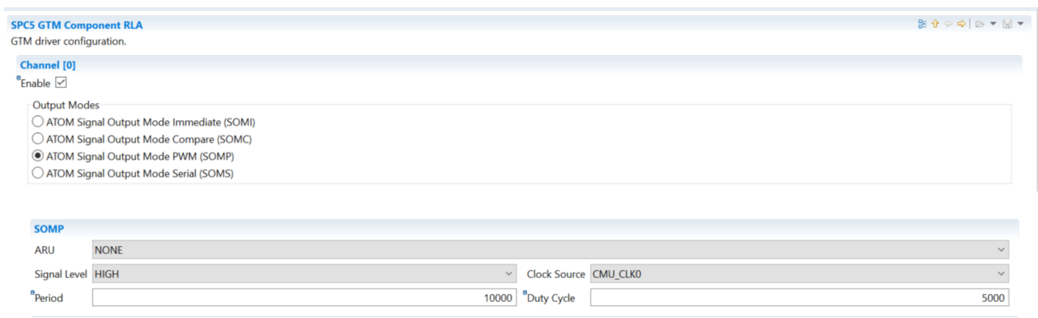


Figure 10. **ATOM0 CH4 - ADC trigger**

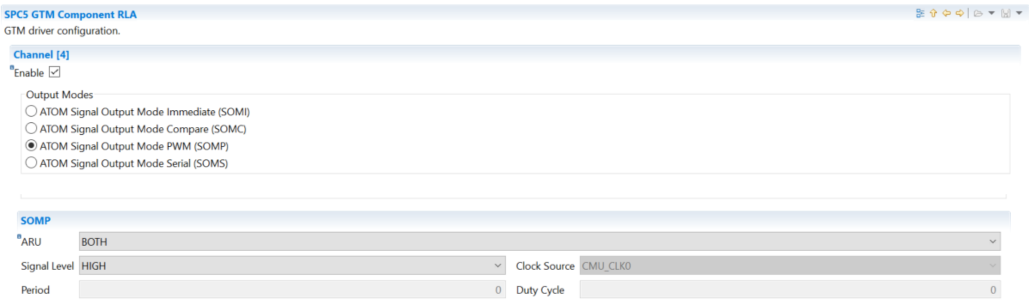
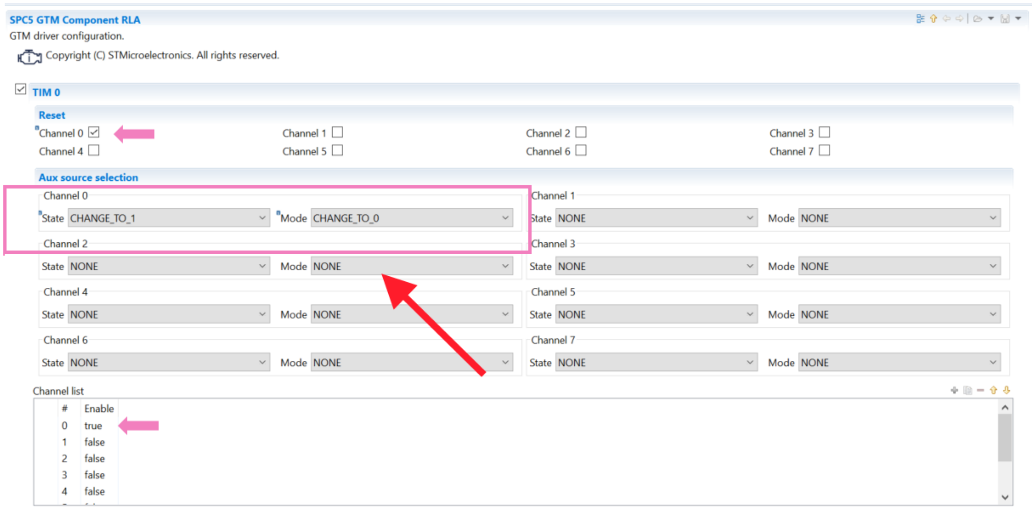
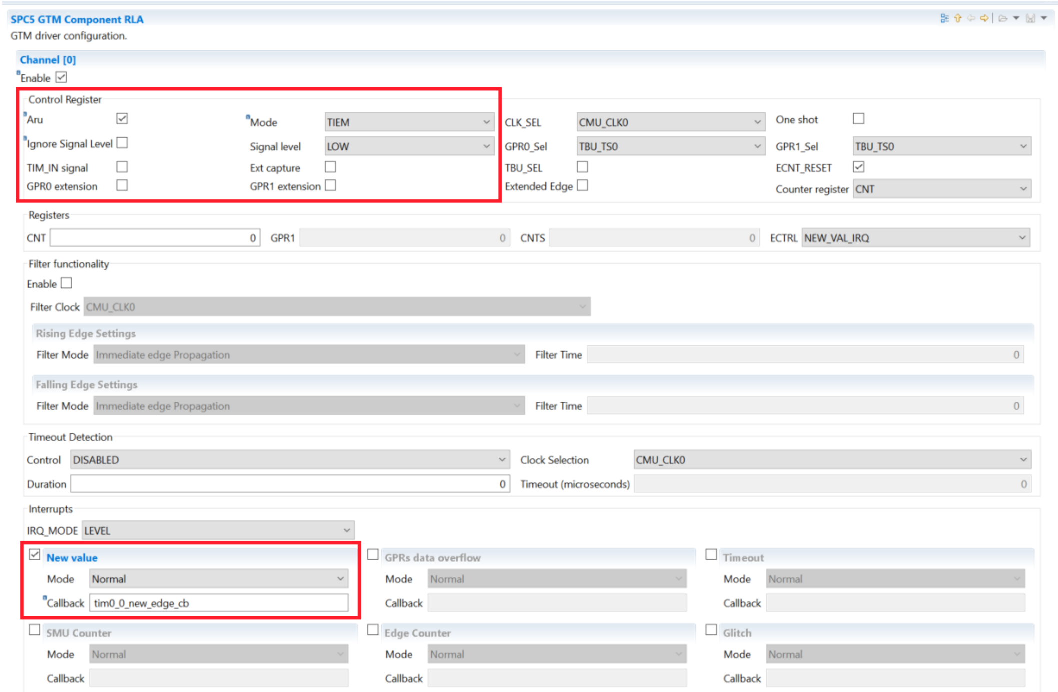


Figure 11. **TIM configuration**



TIM and ATOM are routed internally using the Auxiliary source selection.

Figure 12. TIM0 CH0 configuration



4.2.2 Low level driver configuration

Figure 13. SAR ADC configuration

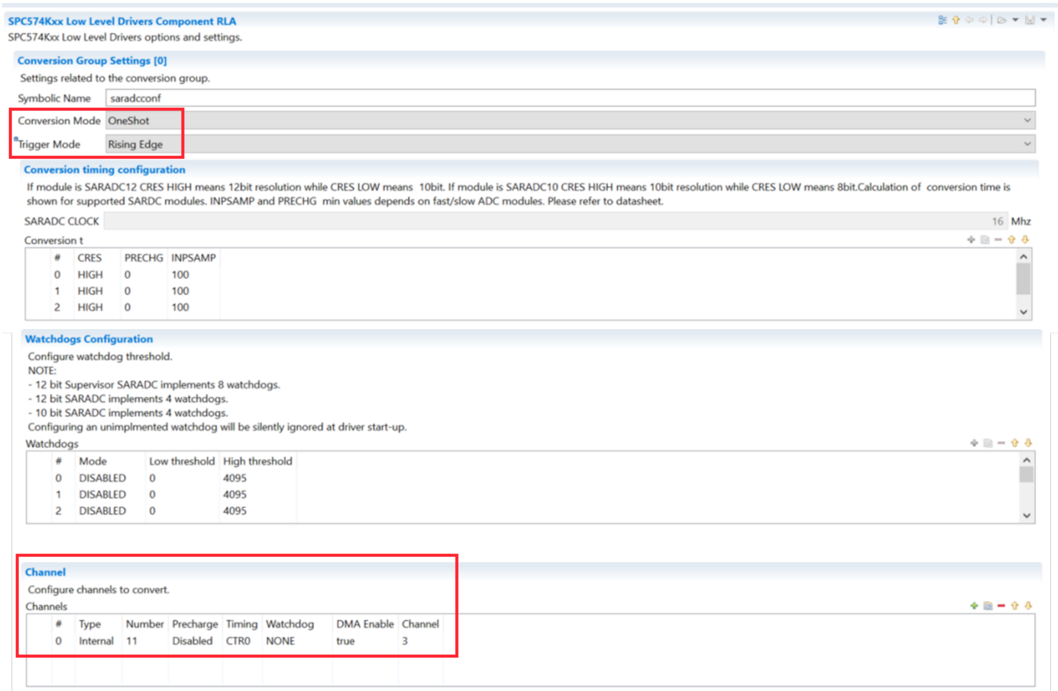
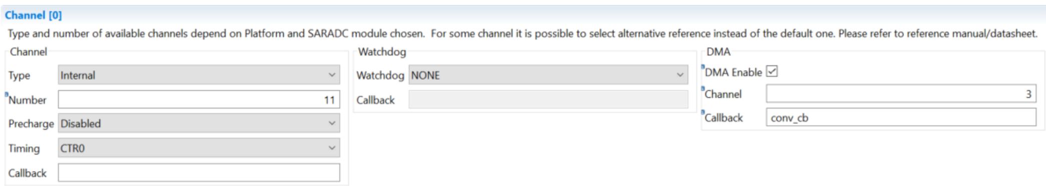


Figure 14. SAR ADC configuration details



4.2.3 Pin board configuration

In this section need to define pin used:

- ATOM0_0
- ATOM0_4
- ANP11

And the setting in Special Muxing for the ADC external trigger (ATOM0_4).

Figure 15. **Pin map settings**

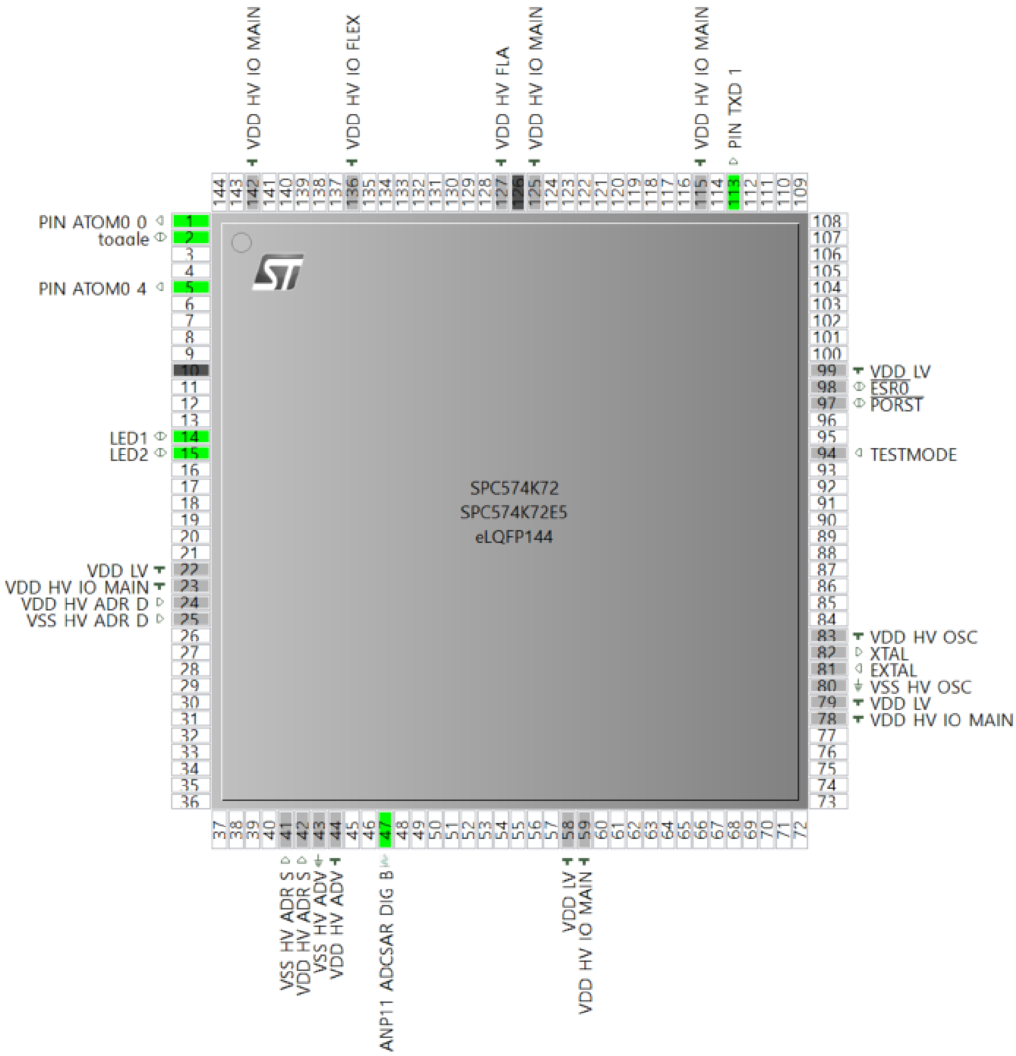
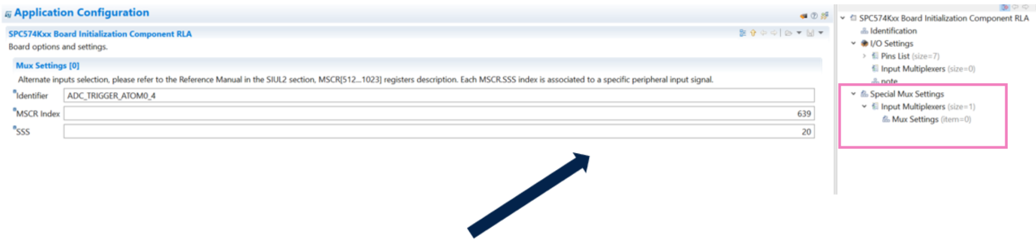


Figure 16. ADC trigger settings



Instance	Input	SIUL2 MSCR Register	Source Signal Select (MSCR[SSS])	Source Instance	Source Signal
SARADC_B	IND_START_TRG	**639**	**0001_0100**	GTM	ATOM0_4

From Input Multiplexing Table (excel file)

5 Code

5.1 Main code

Refer to the Complex PWM application.

Figure 17. Main application loop

```
/*
 * Application entry point.
 */
int main(void) {

        /* Initialization of all the imported components in the order specified in
     the application wizard. The function is generated automatically.*/
        componentsInit();

        /* Enable Interrupts */
        irqIsrEnable();

        /* Start SARADC driver */
        saradc_lld_start(&SARADC12DSV, &saradc_config_saradcconf);

        /* Enable Clock Management Unit */
        gtm_cmuStart(&CMUD1);

        /* Initialize the MCS */
        initMCS();

        /* Connect ATOM0_CH0 to TIM0_CH0 */
        gtm_SetAuxIN(&GTMD, TIM_0, TIM_CHANNEL0, SPC5_GTM_IP_TIM_AUXIN_ATOM);

        gtm_atomStart(&ATOMD1, ATOM_CHANNEL0);

        /* Define the source trigger for the ATOM0 CHANNEL0
         * In this case the port 0 of MCS0 is used as ARU interface
         * (24 port for each MCS are available)
         */
        gtm_atomSetDataSource(&ATOMD1, ATOM_CHANNEL4, SPC5_GTM_WRITE_ADDRESS_MCS0_MCS0_0);

        //PWM used as ADC Trigger
        gtm_atomStart(&ATOMD1, ATOM_CHANNEL4);

        /* set the Duty, Period and clock for the PWM in the MCS code,
         * MCS code start the execution when Trigger bit "0" is set */
        gtm_mcsSetTriggerBit(&MCSD1, MCS_CHANNEL0);

        /*
         * Start TIM0_0, connected with ATOM0_CH0 (routing)
         */
        gtm_timStart(&TIMD1, TIM_CHANNEL0);

        /* Application main loop.*/
        for ( ; ; ) {
                pal_lld_togglepad(PORT_E, LED1);
                osalThreadDelayMilliseconds(100);
        }

         }
```

Figure 18. ISR routines

```
#include "components.h"
#define EDGE_MASK 0x01000000

static uint8_t  converted        = 0U;
static uint16_t converted_value  = 0U;
static uint8_t  ANP_channel      = 0U;
uint32_t adc_trigger = 0;
uint32_t edge_cnt = 0;


/* conversion callback when using DMA */
void saradcconf_ch11_conv_cb(SARADCDriver *saradcp, uint8_t channel, uint16_t value) {
        (void)saradcp;

        ANP_channel     = channel;
        converted_value = value;
        converted       = 1U;
}

void mcs0_cb(GTM_MCSDriver *mcsd, uint8_t channel) {
        (void)mcsd;
        (void)channel;
        pal_lld_togglepad(PORT_D, toggle);
        //Your code
        adc_trigger = 1;
}

void tim0_0_new_edge_cb(GTM_TIMDriver *timd, uint8_t channel) {
        (void) channel;

        if ((timd->tim->CH0_GPR0.R & EDGE_MASK) == 0x01000000) { /* rising edge detected */
                pal_lld_togglepad(PORT_E, LED2);
        }
        if ((timd->tim->CH0_GPR0.R & EDGE_MASK) == 0x0) { /* falling edge detected */
                pal_lld_togglepad(PORT_E, LED2);
                if (adc_trigger == 1) {
                        edge_cnt++;
                        if (edge_cnt == 2) {
                                gtm_atomStop(&ATOMD1, ATOM_CHANNEL4);
                                gtm_atomARUEnable(&ATOMD1, ATOM_CHANNEL4, 0);
                                gtm_atomSetDuty(&ATOMD1, ATOM_CHANNEL4, 0);
                                gtm_atomSetPeriod(&ATOMD1, ATOM_CHANNEL4, 0);
                                gtm_atomARUEnable(&ATOMD1, ATOM_CHANNEL4, 1);
                                gtm_atomStart(&ATOMD1, ATOM_CHANNEL4);
                                adc_trigger = 0;
                                edge_cnt = 0;
                        }
                }
        }
 }
```

ADC trigger (Atom0_CH4) is disabled after 2 "active" PWM pulse.

Figure 19. **MCS initialization**

```
/*
 * Initializes the MCS.
 */
static void initMCS(void) {

        /* Reset the RAM and wait for action complete */
        gtm_mcsResetRAM(&MCSD1);
        while (gtm_mcsGetResetRAM(&MCSD1) == 1U) {
                ;
        }

        /* Load the MCS program */
        gtm_mcsLoadProgram(&MCSD1, mcs0_mem, SIZE_MCS0_MEM, OFFSET_MCS0_MEM);

        /* Enable the channel */
        gtm_mcsEnableChannel(&MCSD1, MCS_CHANNEL0);

  }
```

5.2 MCS code

Refer to the Complex MCS application for the detail.

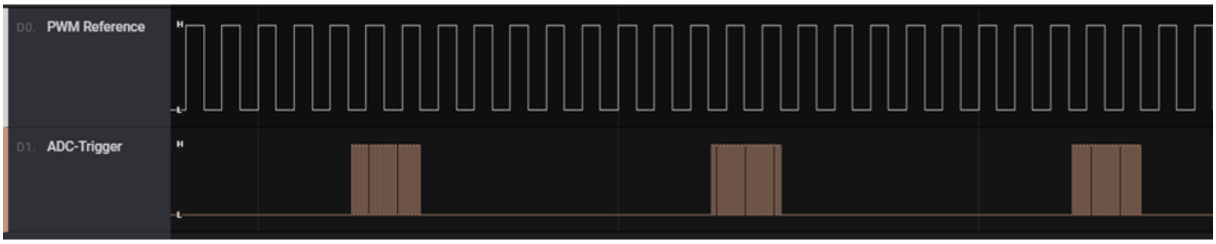
Figure 20. **mcs0.mcs - ch0 code**

```
tsk0_init:
    movl R7 (tsk0_stack-4); initialize stack pointer of task 0
        movl R0, 0
        movl R1, 0
        movl R2, 0
        movl R3, 0
        movl R4, 0
        movl R5, 0
        movl R6, 0

TASK0_START:
    movl CTRG, 1 ; Clear start trigger

TASK0_CHECK_START_SEQUENCE:
    movl  R0, 1
    wurm  R0, STRG, 1
    movl CTRG, 1 ; Clear start trigger

WAIT_TIM_EDGE:
   ard R0, R1, TIM0_WRADDR0    ; Wait trigger from TIM, notified by ARU only TIM Falling Edge
   xorl ACB 0                  ; double check
   jbc STA Z WAIT_TIM_EDGE

   addl R2, 1                  ; loop util 10 edges occurred
   atul R2, 10
   jbc STA Z WAIT_TIM_EDGE
   movl R2, 0                  ; reset counter (R2)

;Configure PWM                 ; Start PWM for ADC Trigger T = 1ms
   movl ACB, PWM_CLK_SRC
   movl R1, PMW_DUTY
   movl R0, PWM_PERIOD
   awr R0, R1, PWM_MCS0_0_PORT_IDX

   xorl STA, 2                 ; raise ISR

TASK0_JUMP_TO_NEXT_UPDATE:
   jmp   WAIT_TIM_EDGE

TASK0_END:
;Should not get here
                andl STA EN_L_MSK ; disable MCS channel 0
```

5.3 Result

The result of this application is a PWM that start and stop due to external event, TIM edges in our case. This PWM is used as external trigger for the SAR ADC.

Figure 21. Result



6 Conclusion

The document gives all the information and method to programming your application using the GTM to trigger an ADC device.

In this example have been used SAR ADC device but it is the same for the SD ADC.

This approach is very useful in terms of resource and speed in Automotive Engine Motor Control application.

Using SPC5Studio, the GUI approach is user friendly and very easy to use, few steps giving keys information to develop the project application.

A set of APIs are available to be used in the user code to customize the GUI configuration parameter and the application developing.

The online documentation is available to help the user in his project application development.

Appendix A Reference documents

Table 5. Reference documents

Doc Name	ID	Title
AN5552	034599	Generate PWM signals with GTM
RM0361	025070	Generic Timer Module specification revision 1.5.5.1

Revision history

Table 6. **Document revision history**

Date	Version	Changes
09-Sep-2021	1	Initial release.

Contents

List of tables

List of figures