Analog-to-Digital Converter *(ADC)*

Based on *dr. inż. Maciej Dzieniakowski's knowledge*

Adrian Chmiel

23 may 2024

1 Task

The signal transmitted by the ADC converter controlled by switch A1 is fed *(with a delay)* to the DAC converters and displayed on the oscilloscope screen.

2 Configuration

The configuration is divided into three main steps:

1. **DAC converters configuration**
2. **ADC converter configuration**
3. **Setting switch A1 as an input channel**

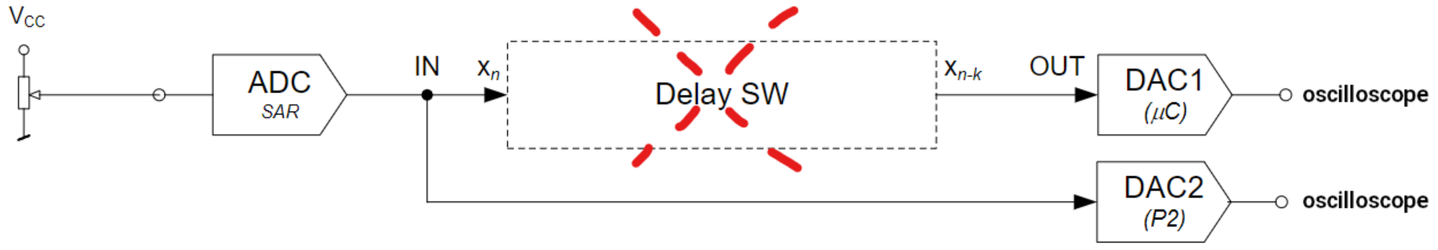
This division is also present in the rest of the document and it is recommended to follow this order to minimize encountered problems.

3 Other remarks

Firstly, we received a couple of advices that are worth mentioning at the beginning:

- Proper ADC converter configuration is not an easy task, especially if we do that for the very first time
- The available ADC converter configuration allows for a lot of control over the signal through many available parameters
- The default values of the bits for each parameter are **0** which means that we do not always have to set them

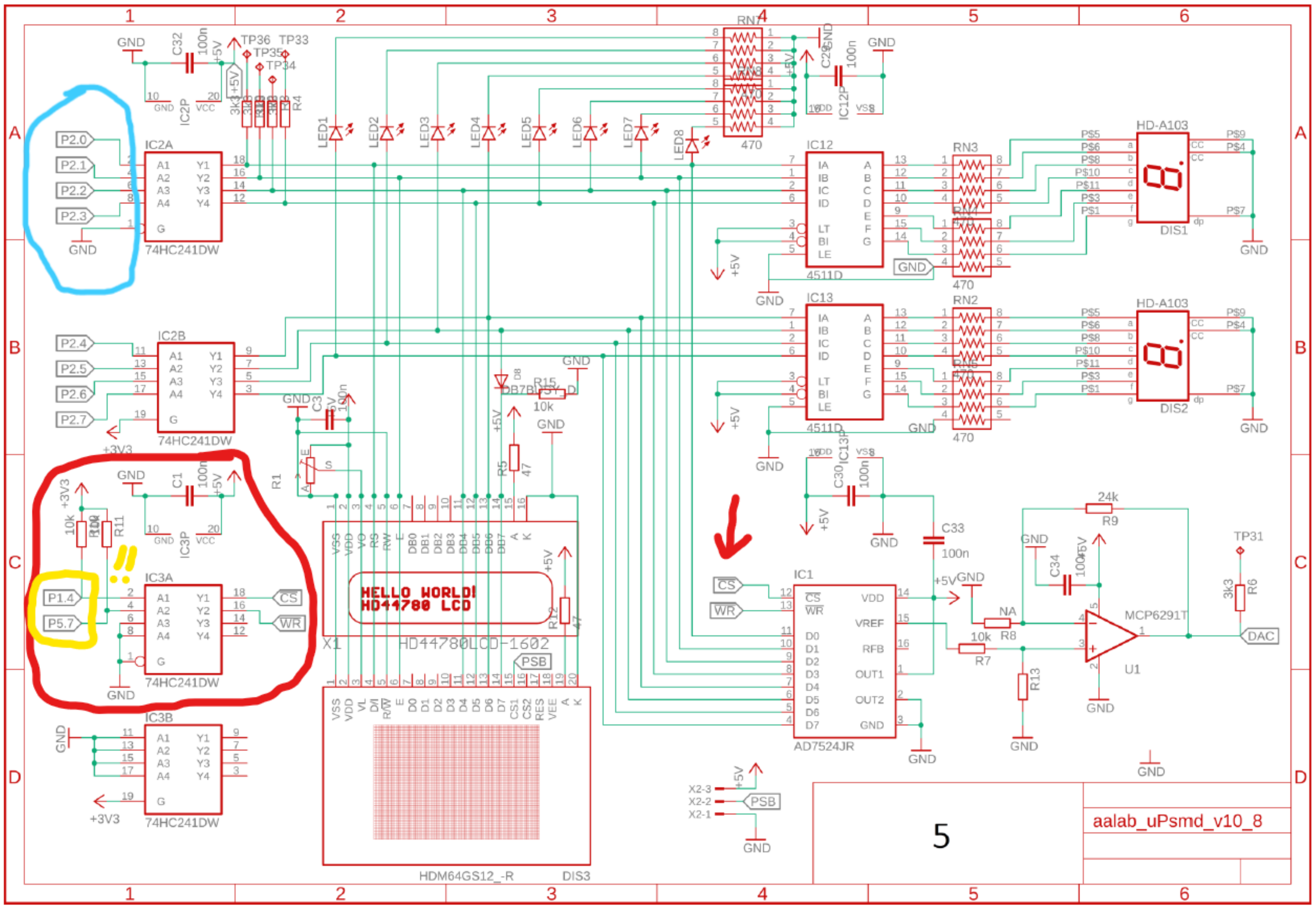
It is recommended to follow the following scheme:



Source: ADC_delay_DAC.pdf, page: 1

Delay SW means software (in this case a code fragment) delaying the signal. I marked this element with a dashed line to indicate that for now it is not important for us. However, this does not mean that this software should not be included in the final version of the program.

4 DAC converters configuration



Source: Blocks_scheme.pdf, page: 7

According to the above scheme, it is necessary to configure the *P1.4* and *P5.7* ports as **outputs** so that it is possible to view the results on the oscilloscope screen after connecting it to the appropriate pins. In addition, it is necessary to set all pins from port *P2* as **outputs**. This can be done with the following code fragment in the initialization section:

```
BIS.B #00010000b, &P1DIR ; set P1.4 as out
BIS.B #10000000b, &P5DIR ; set P5.7 as out
BIC.B #00010000b, &P1OUT ; clear bit P1.4
BIC.B #10000000b, &P5OUT ; clear bit P5.7
MOV.B #255, P2DIR        ; set all pins from port 2 as outputs
MOV.B #0, P2OUT          ; set port 2 to low
```

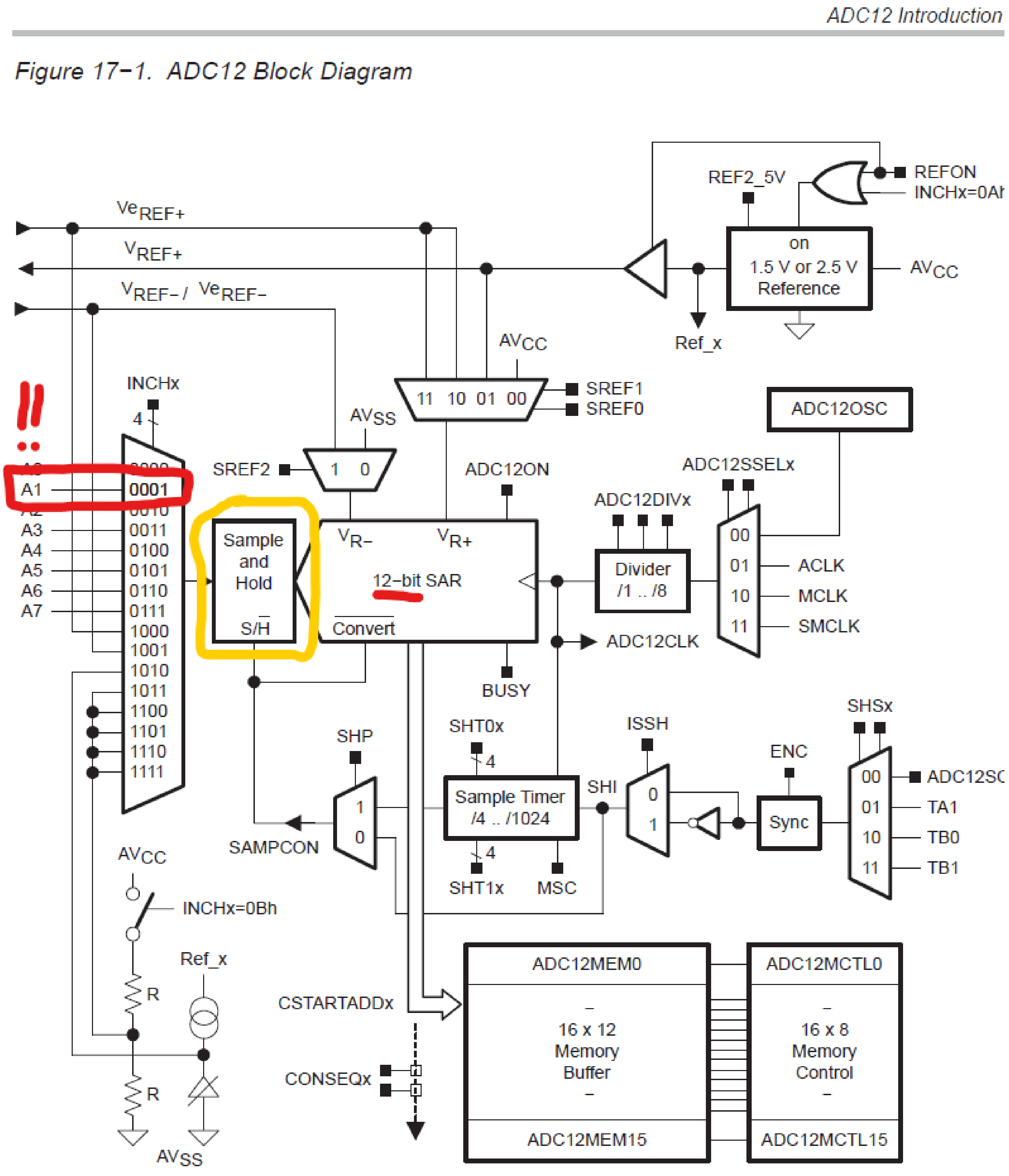
The rest of the configuration can be done using the code we received during the **timer task**:

```
     ;---------- Basic Clock Module Initialisation --------------------------------------
     ; - switch from DCO to XT2
     ; - MCLK & SMCLK supplied from XT2, ACLK = n/a
     ; - the DCO is left runing
     bis.b #OSCOFF,SR ;turn OFF osc.1
     bic.b #XT2OFF,BCSCTL1 ;turn ON osc.2
BCM0 bic.b #OFIFG,&IFG1 ;clear OFIFG
     mov #0FFFFh,R15 ;delay (waiting for oscilator start)
BCM1 dec R15 ;delay
     ;jnz BCM1 ;delay -> commented out as it was ocasionally leading to infinite loop
     bit.b #OFIFG,&IFG1 ;test OFIFG
     jnz BCM0 ;repeat test if needed
     ;MCLK
     bic.b #040h,&BCSCTL2 ;slelect XT2CLK as source
     bis.b #080h,&BCSCTL2 ;
     bic.b #030h,&BCSCTL2 ;MCLK=source/1 (8MHz)
     ;SMCLK
     bis.b #SELS,&BCSCTL2 ;slelect XT2CLK as source
     bic.b #006h,&BCSCTL2 ;SMCLK=source/1 (8MHz)
     ;---

     ;... ;DAC_0 initialisation
     bis.w #REFON+REF2_5V,&ADC12CTL0 ;Reference generator ON, VRef+=2.5V
     bic #DAC12SREF0,&DAC12_0CTL ;set Vref=VREF+
     bic #DAC12SREF1,&DAC12_0CTL ;
     bic #DAC12RES,&DAC12_0CTL ;12-bit resolution
     bic #DAC12LSEL0,&DAC12_0CTL ;Load mode 0
     bic #DAC12LSEL1,&DAC12_0CTL ;
     bis #DAC12IR,&DAC12_0CTL ;Full-Scale=1xVref
     bis #DAC12AMP0,&DAC12_0CTL ;High speed amplifier output
     bis #DAC12AMP1,&DAC12_0CTL ;
     bis #DAC12AMP2,&DAC12_0CTL ;
     bic #DAC12DF,&DAC12_0CTL ;Data format - straight binary
     bic #DAC12IE,&DAC12_0CTL ;Interrupt disabled
     bis #DAC12ENC,&DAC12_0CTL ;DAC_0 conversion enabled
     ;...

     ;... ;DAC_1 initialisation
     bis.w #REFON+REF2_5V,&ADC12CTL0 ;Reference generator ON, VRef+=2.5V
     bic #DAC12SREF0,&DAC12_1CTL ;set Vref=VREF+
     bic #DAC12SREF1,&DAC12_1CTL ;
     bic #DAC12RES,&DAC12_1CTL ;12-bit resolution
     bic #DAC12LSEL0,&DAC12_1CTL ;Load mode 0
     bic #DAC12LSEL1,&DAC12_1CTL ;
     bis #DAC12IR,&DAC12_1CTL ;Full-Scale=1xVref
     bis #DAC12AMP0,&DAC12_1CTL ;High speed amplifier output
     bis #DAC12AMP1,&DAC12_1CTL ;
     bis #DAC12AMP2,&DAC12_1CTL ;
     bic #DAC12DF,&DAC12_1CTL ;Data format - straight binary
     bic #DAC12IE,&DAC12_1CTL ;Interrupt disabled
     bis #DAC12ENC,&DAC12_1CTL ;DAC_1 conversion enabled
     ;...
```

5 ADC converters configuration



Source: slau049f.pdf, page: 346

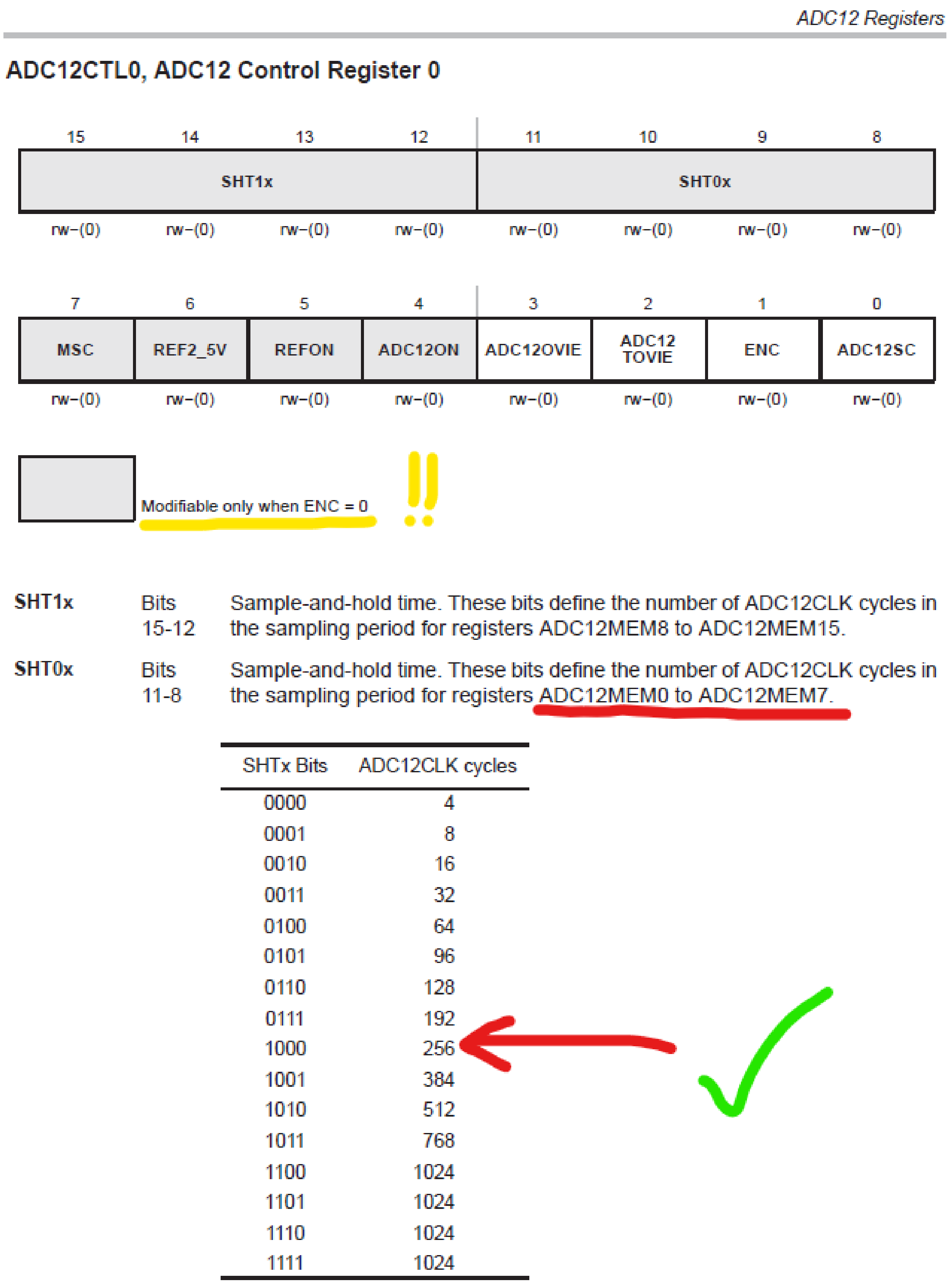
The ADC converter configuration is a bit more complicated than the DAC converters configuration. It is worth paying attention to a couple of aspects:

- The ADC converter is **12-bit**, but in reality we are only interested in the **8 least significant bits**, because one of the used DAC converters is only **8-bit**.
- The *Sample and Hold* parameter allows you to take a sample and hold it - this is necessary for the converter to work correctly with a delay.

The entire configuration process can be divided into three steps:

1. **Initial configuration of ADC12CTL0**

   The documentation corresponding to this step is below:



ADC12 Registers

ADC12CTL0, ADC12 Control Register 0

15	14	13	12	11	10	9	8
SHT1x				SHT0x			
rw–(0)	rw–(0)	rw–(0)	rw–(0)	rw–(0)	rw–(0)	rw–(0)	rw–(0)

7	6	5	4	3	2	1	0
MSC	REF2_5V	REFON	ADC12ON	ADC12OVIE	ADC12 TOVIE	ENC	ADC12SC
rw–(0)	rw–(0)	rw–(0)	rw–(0)	rw–(0)	rw–(0)	rw–(0)	rw–(0)

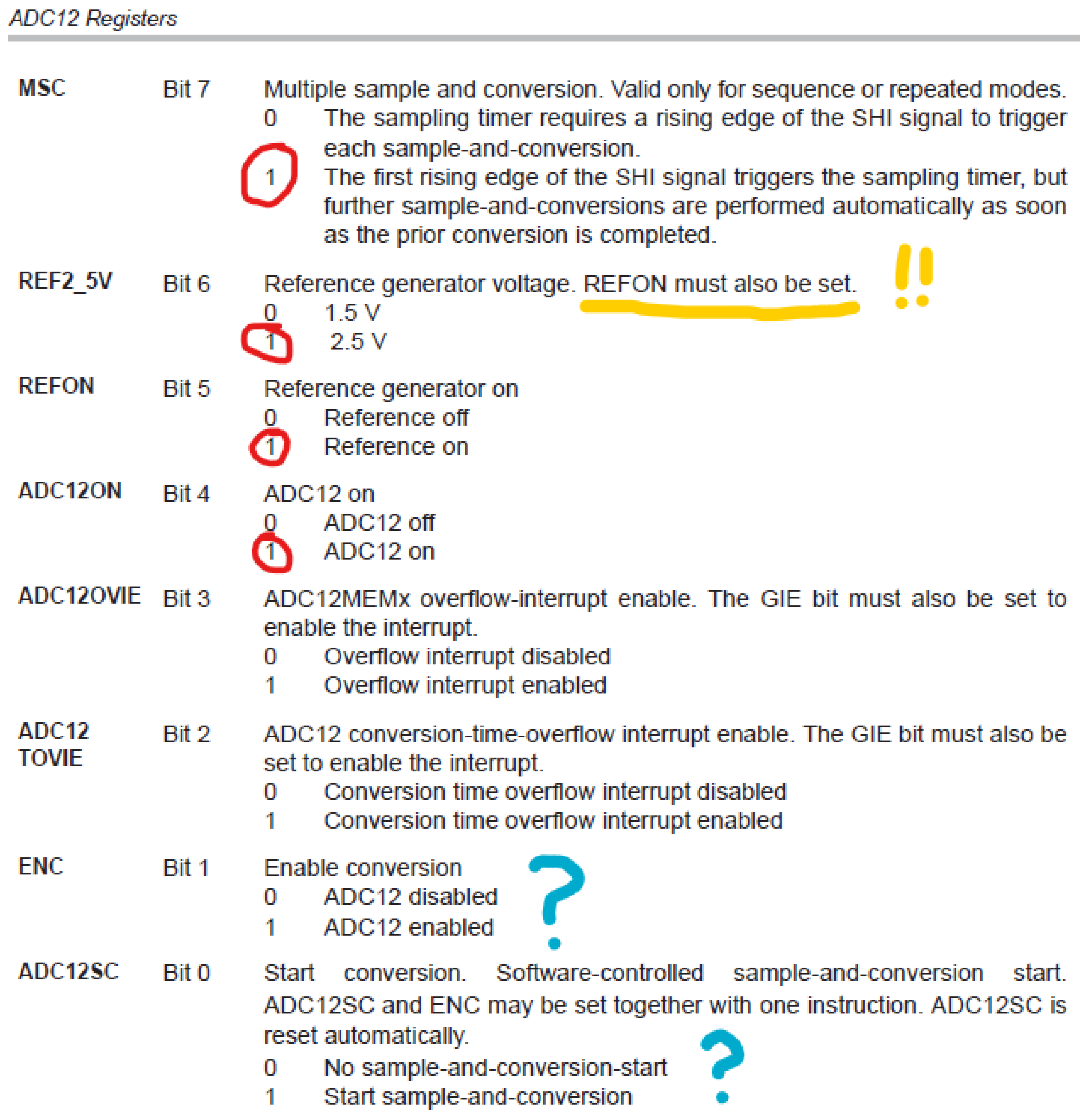
▨ Modifiable only when ENC = 0 !!

SHT1x	Bits 15-12	Sample-and-hold time. These bits define the number of ADC12CLK cycles in the sampling period for registers ADC12MEM8 to ADC12MEM15.
SHT0x	Bits 11-8	Sample-and-hold time. These bits define the number of ADC12CLK cycles in the sampling period for registers ADC12MEM0 to ADC12MEM7.

SHTx Bits	ADC12CLK cycles
0000	4
0001	8
0010	16
0011	32
0100	64
0101	96
0110	128
0111	192
1000	256 ✓
1001	384
1010	512
1011	768
1100	1024
1101	1024
1110	1024
1111	1024

Source: slau049f.pdf, page: 364

Due to the fact that we will only deal with 8-bit numbers, we only need cells *ADC12MEM0 - ADC12MEM7*. Therefore, it is enough to set only the sample holding time *SHT0* corresponding to these cells. The recommended time is **256 clock cycles**!



ADC12 Registers

MSC	Bit 7	Multiple sample and conversion. Valid only for sequence or repeated modes. 0 The sampling timer requires a rising edge of the SHI signal to trigger each sample-and-conversion. [✓ 1] The first rising edge of the SHI signal triggers the sampling timer, but further sample-and-conversions are performed automatically as soon as the prior conversion is completed.
REF2_5V	Bit 6	Reference generator voltage. REFON must also be set. !! 0 1.5 V [✓ 1] 2.5 V
REFON	Bit 5	Reference generator on 0 Reference off [✓ 1] Reference on
ADC12ON	Bit 4	ADC12 on 0 ADC12 off [✓ 1] ADC12 on
ADC12OVIE	Bit 3	ADC12MEMx overflow-interrupt enable. The GIE bit must also be set to enable the interrupt. 0 Overflow interrupt disabled 1 Overflow interrupt enabled
ADC12 TOVIE	Bit 2	ADC12 conversion-time-overflow interrupt enable. The GIE bit must also be set to enable the interrupt. 0 Conversion time overflow interrupt disabled 1 Conversion time overflow interrupt enabled
ENC	Bit 1	Enable conversion ? 0 ADC12 disabled 1 ADC12 enabled
ADC12SC	Bit 0	Start conversion. Software-controlled sample-and-conversion start. ADC12SC and ENC may be set together with one instruction. ADC12SC is reset automatically. 0 No sample-and-conversion-start ? 1 Start sample-and-conversion

Source: slau049f.pdf, page: 365

This part requires a thorough understanding of the above documentation fragment, as we change quite a few parameters here.

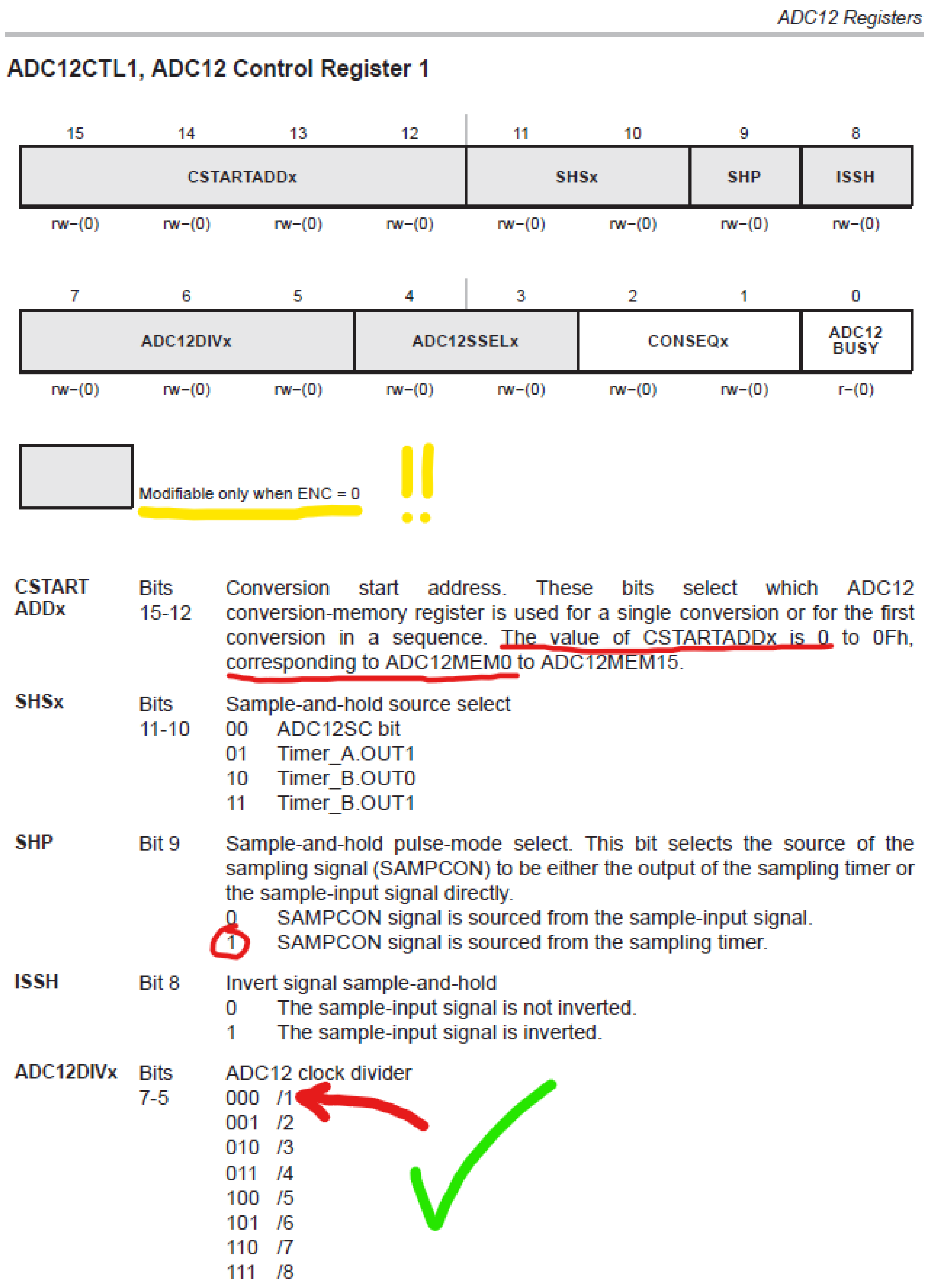
- *MSC* - allows the next conversions to be performed automatically one after the other
- *REF2_5V* - we set the reference voltage to 2.5V
- *REFON* - we turn on the reference voltage generator
- *ADC12ON* - we turn on the ADC converter
- *ADC12OVIE & ADC12TOVIE* - correspond to the interrupt, which we will not use, so the bit value remains unchanged
- *ENC* - allows conversion **(set only at the end of the entire configuration!)**
- *ADC12SC* - start conversion **(set only at the end of the entire configuration!)**

The code performing this step is available below:

```
BIS.W #0000100011110000b, &ADC12CTL0
; SHT0 = 1000b (256 cycles) | MSC = 1 | REF2_5V = 1 | REFON = 1 | ADC12ON = 1
```

2. **ADC12CTL1 configuration**

The documentation corresponding to this step is below:



ADC12 Registers

ADC12CTL1, ADC12 Control Register 1

15	14	13	12	11	10	9	8
CSTARTADDx				SHSx		SHP	ISSH
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

7	6	5	4	3	2	1	0
ADC12DIVx			ADC12SSELx		CONSEQx		ADC12 BUSY
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	r-(0)

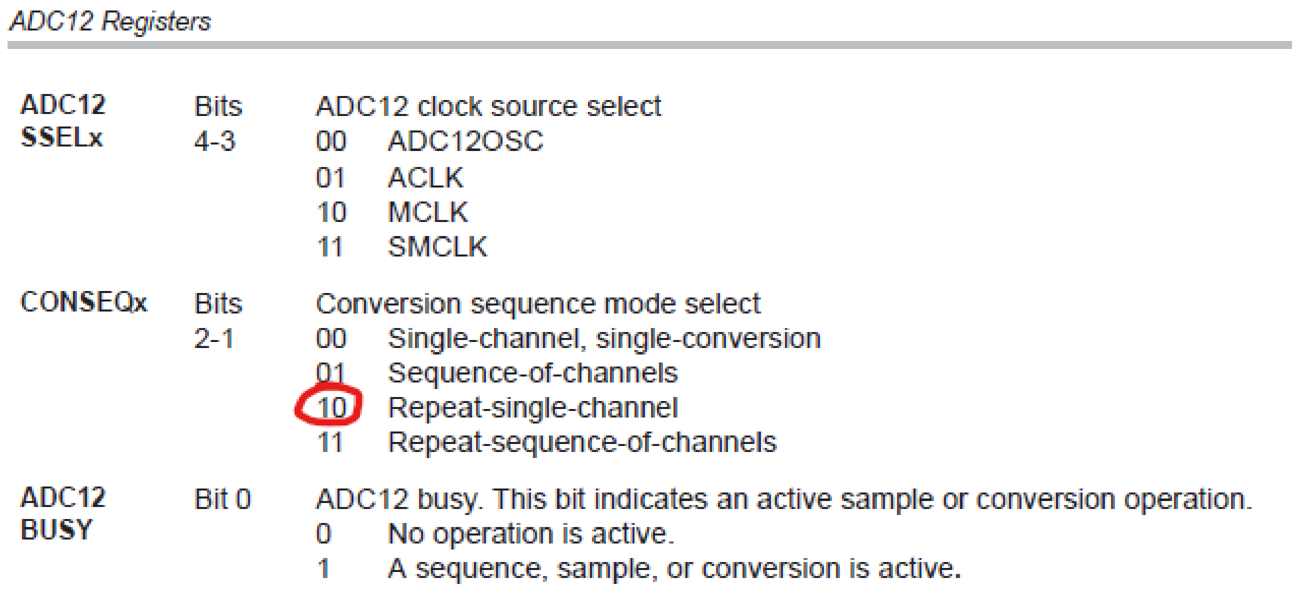
▨ Modifiable only when ENC = 0 !!

CSTART ADDx	Bits 15-12	Conversion start address. These bits select which ADC12 conversion-memory register is used for a single conversion or for the first conversion in a sequence. The value of CSTARTADDx is 0 to 0Fh, corresponding to ADC12MEM0 to ADC12MEM15.
SHSx	Bits 11-10	Sample-and-hold source select 00 ADC12SC bit 01 Timer_A.OUT1 10 Timer_B.OUT0 11 Timer_B.OUT1
SHP	Bit 9	Sample-and-hold pulse-mode select. This bit selects the source of the sampling signal (SAMPCON) to be either the output of the sampling timer or the sample-input signal directly. 0 SAMPCON signal is sourced from the sample-input signal. [✓ 1] SAMPCON signal is sourced from the sampling timer.
ISSH	Bit 8	Invert signal sample-and-hold 0 The sample-input signal is not inverted. 1 The sample-input signal is inverted.
ADC12DIVx	Bits 7-5	ADC12 clock divider 000 /1 001 /2 010 /3 011 /4 100 /5 101 /6 110 /7 111 /8

Source: slau049f.pdf, page: 366

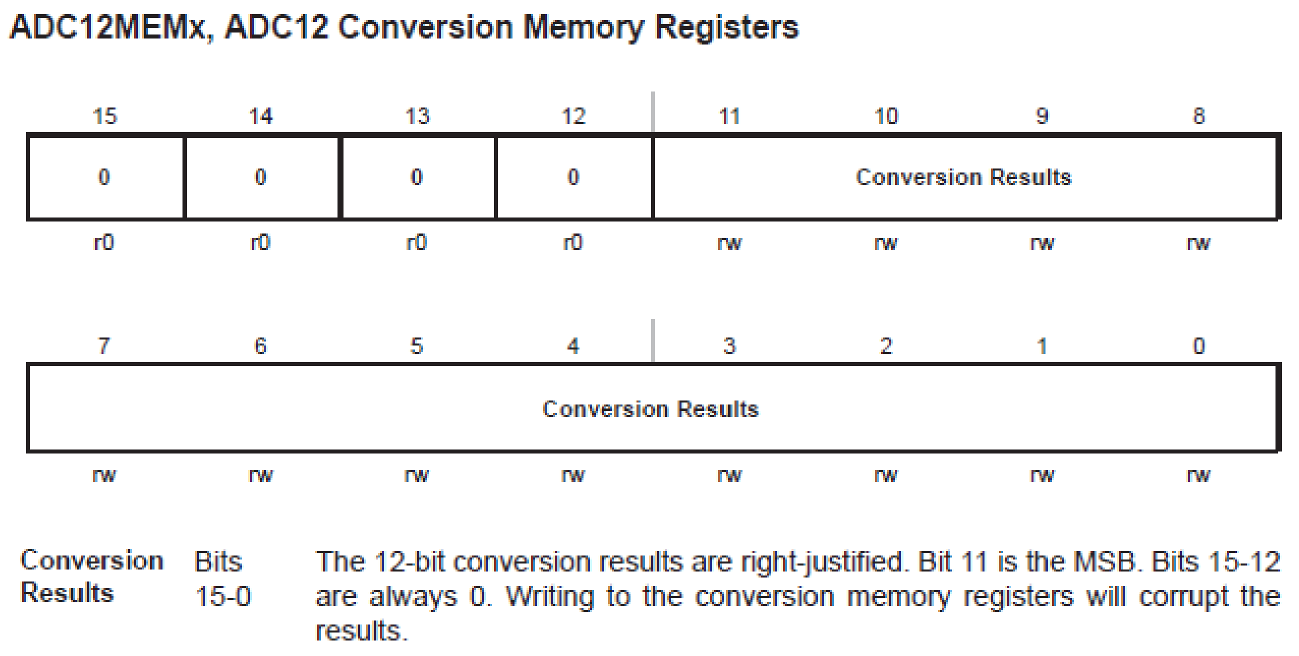
The following parameters are important in this step:

- *CSTARTADDx* - this is the address where the conversion result will start being saved - we leave it unchanged, so it will be *ADC12MEM0*
- *SHP* - allows an external signal to control the conversion (i.e. in our case **switch A1**)
- *ADC12DIVx* - this value divides the clock that the converter works with - we leave it unchanged to not change the clock frequency, but it is worth noting that this is a parameter that can be changed depending on the needs



ADC12 Registers

ADC12 SSELx	Bits 4-3	ADC12 clock source select 00 ADC12OSC 01 ACLK 10 MCLK 11 SMCLK
CONSEQx	Bits 2-1	Conversion sequence mode select 00 Single-channel, single-conversion 01 Sequence-of-channels [✓ 10] Repeat-single-channel 11 Repeat-sequence-of-channels
ADC12 BUSY	Bit 0	ADC12 busy. This bit indicates an active sample or conversion operation. 0 No operation is active. 1 A sequence, sample, or conversion is active.



ADC12MEMx, ADC12 Conversion Memory Registers

15	14	13	12	11	10	9	8
0	0	0	0	Conversion Results			
r0	r0	r0	r0	rw	rw	rw	rw

7	6	5	4	3	2	1	0
Conversion Results							
rw	rw	rw	rw	rw	rw	rw	rw

Conversion Results	Bits 15-0	The 12-bit conversion results are right-justified. Bit 11 is the MSB. Bits 15-12 are always 0. Writing to the conversion memory registers will corrupt the results.

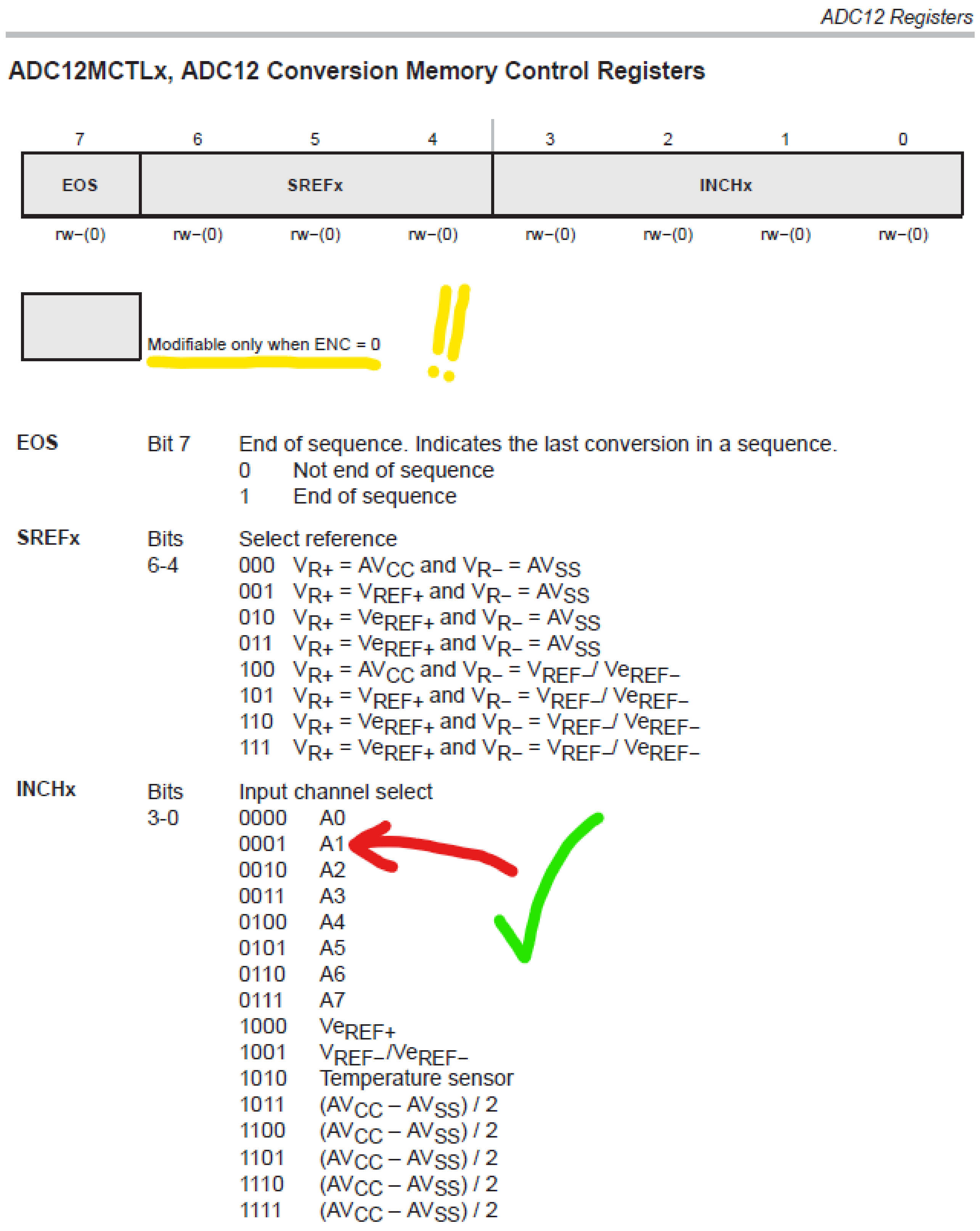
Source: slau049f.pdf, page: 367

We change the *CONSEQx* parameter to **10** so that the conversion is **repeated** based on a **single** channel with the A1 converter. The code performing the entire step with the *ADC12CTL1* configuration is below:

```
BIS.W #0000001000000010b, &ADC12CTL1
; SHP = 1 | CONSEQ2 = 1 -> A1 goes to MEM0
```

3. **ADC12MCTL0 configuration**

The documentation corresponding to this step is below:



ADC12 Registers

ADC12MCTLx, ADC12 Conversion Memory Control Registers

7	6	5	4	3	2	1	0
EOS	SREFx			INCHx			
rw–(0)	rw–(0)	rw–(0)	rw–(0)	rw–(0)	rw–(0)	rw–(0)	rw–(0)

Modifiable only when ENC = 0 !!

EOS	Bit 7	End of sequence. Indicates the last conversion in a sequence. 0 Not end of sequence 1 End of sequence
SREFx	Bits 6-4	Select reference 000 $V_{R+} = AV_{CC}$ and $V_{R-} = AV_{SS}$ 001 $V_{R+} = V_{REF+}$ and $V_{R-} = AV_{SS}$ 010 $V_{R+} = Ve_{REF+}$ and $V_{R-} = AV_{SS}$ 011 $V_{R+} = Ve_{REF+}$ and $V_{R-} = AV_{SS}$ 100 $V_{R+} = AV_{CC}$ and $V_{R-} = V_{REF-}/ Ve_{REF-}$ 101 $V_{R+} = V_{REF+}$ and $V_{R-} = V_{REF-}/ Ve_{REF-}$ 110 $V_{R+} = Ve_{REF+}$ and $V_{R-} = V_{REF-}/ Ve_{REF-}$ 111 $V_{R+} = Ve_{REF+}$ and $V_{R-} = V_{REF-}/ Ve_{REF-}$
INCHx	Bits 3-0	Input channel select 0000 A0 0001 A1 0010 A2 0011 A3 0100 A4 0101 A5 0110 A6 0111 A7 1000 Ve_{REF+} 1001 V_{REF-}/Ve_{REF-} 1010 Temperature sensor 1011 $(AV_{CC} - AV_{SS}) / 2$ 1100 $(AV_{CC} - AV_{SS}) / 2$ 1101 $(AV_{CC} - AV_{SS}) / 2$ 1110 $(AV_{CC} - AV_{SS}) / 2$ 1111 $(AV_{CC} - AV_{SS}) / 2$

Source: slau049f.pdf, page: 368

The only thing we do in this step is to set the input channel to **A1**, which will allow the switch to work correctly. This is achieved by setting the *INCHx* register to **0001**. The code performing this configuration is below:

```
BIS.B #00000001b, &ADC12MCTL0 ; set input channel as A1
```

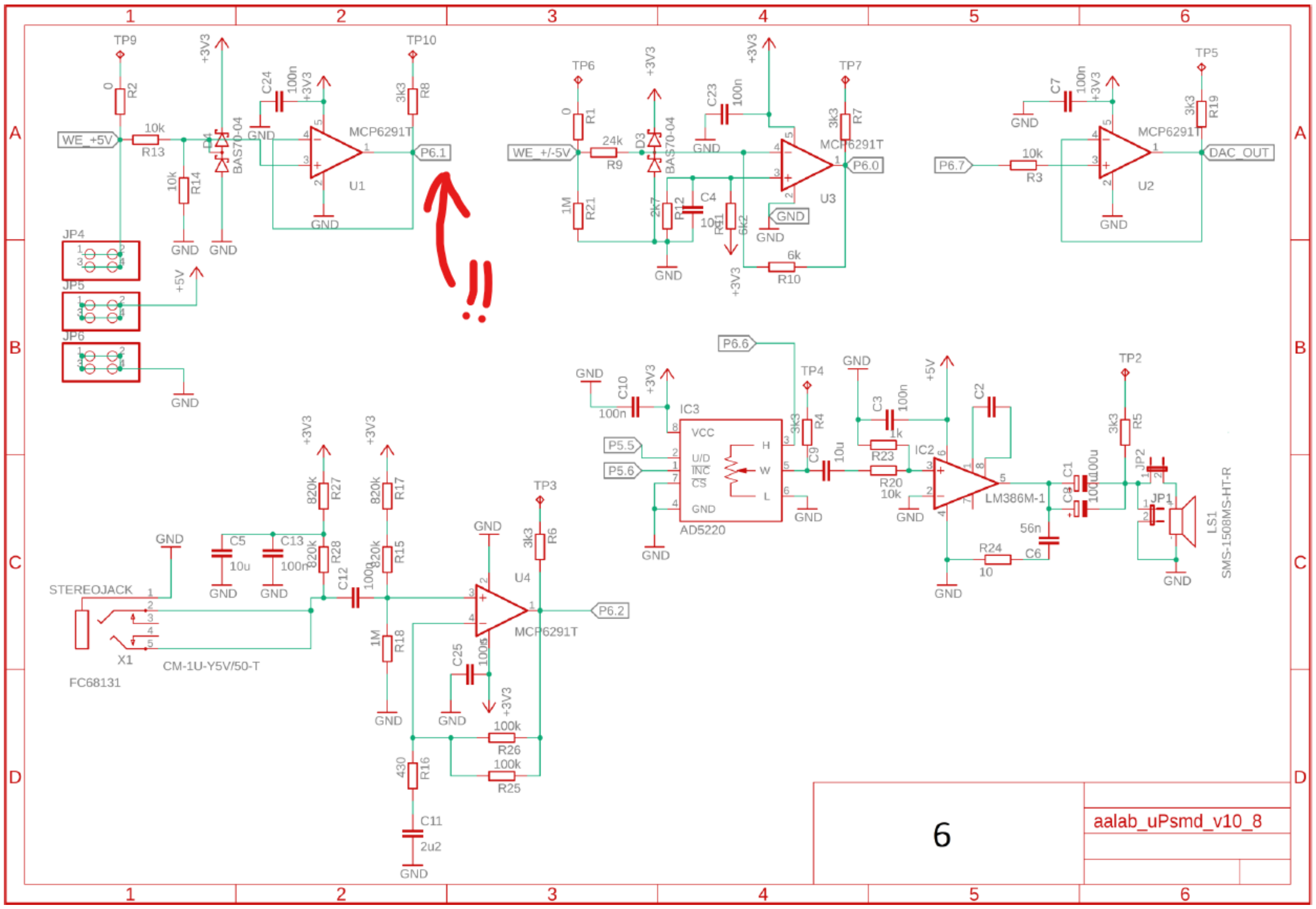
4. **Allowing conversion and starting it**

   We return to the *ADC12CTL0* configuration. This step must be done as the last element of the configuration, as it prevents the change of most of the other settings modified earlier. To do this, we change the values of the *ENC* and *ADC12SC* registers to **1**.

   The code performing this step is below:

   ```
   BIS.W #11b, &ADC12CTL0 ; has to be at the end
   ; ENC = 1 | ADC12SC = 1 -> enables and starts conversion
   ```

6 Setting switch A1 as an input channel



Source: Blocks_scheme.pdf, page: 8 We have already partially done this task in the previous steps, but just like in the case of the converters above, it is necessary to set certain bits to specific values. In this case, it is necessary to set the *P6.1* port as an **analog input** with the *P6SEL* register using the following code:

```
BIS.B #10b, &P6SEL ; set P6.1 as analog input
```

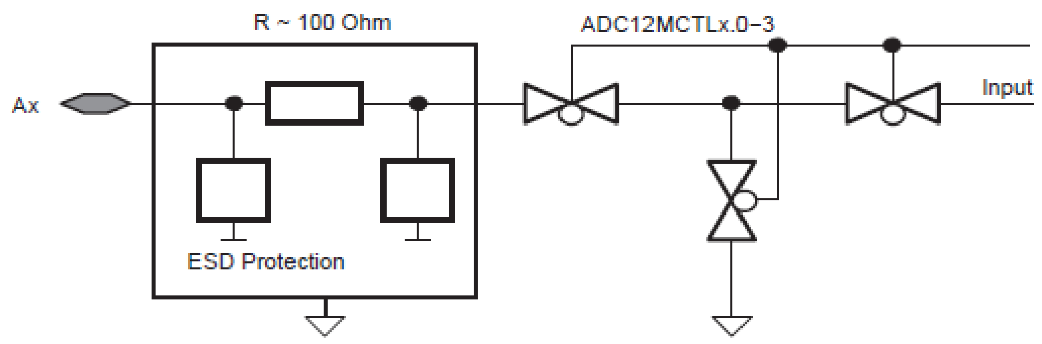
This fact can be supported by a fragment of the documentation below:

17.2.2 ADC12 Inputs and Multiplexer

The eight external and four internal analog signals are selected as the channel for conversion by the analog input multiplexer. The input multiplexer is a break-before-make type to reduce input-to-input noise injection resulting from channel switching as shown in Figure 17−2. The input multiplexer is also a T-switch to minimize the coupling between channels. Channels that are not selected are isolated from the A/D and the intermediate node is connected to analog ground (AV_{SS}) so that the stray capacitance is grounded to help eliminate crosstalk.

The ADC12 uses the charge redistribution method. When the inputs are internally switched, the switching action may cause transients on the input signal. These transients decay and settle before causing errant conversion.

Figure 17−2. Analog Multiplexer



Analog Port Selection

!!

The ADC12 inputs are multiplexed with the port P6 pins, which are digital CMOS gates. When analog signals are applied to digital CMOS gates, parasitic current can flow from V_{CC} to GND. This parasitic current occurs if the input voltage is near the transition level of the gate. Disabling the port pin buffer eliminates the parasitic current flow and therefore reduces overall current consumption. The P6SELx bits provide the ability to disable the port pin input and output buffers.

```
; P6.0 and P6.1 configured for analog input
      BIS.B #3h,&P6SEL   ; P6.1 and P6.0 ADC12 function
```

Source: slau049f.pdf, page: 348

7 Final code with the entire configuration

Complete code of the program with the configuration is visible below.

```
#include "msp430.h"                 ; #define controlled include file

     NAME    main                   ; module name

     PUBLIC  main                   ; make the main label vissible
                                    ; outside this module
     ORG     0FFECh
     DC16    TIMER_A0_Interrupt
     ORG     0FFFEh
     DC16    init                   ; set reset vector to 'init' label

     RSEG    CSTACK                 ; pre-declaration of segment
     RSEG    CODE                   ; place program in 'CODE' segment

init:   MOV     #SFE(CSTACK), SP        ; set up stack
     ; DAC_2 (P2) config
     BIS.B #00010000b, &P1DIR ; set P1.4 as out
     BIS.B #10000000b, &P5DIR ; set P5.7 as out
     BIC.B #00010000b, &P1OUT ; clear bit P1.4
     BIC.B #10000000b, &P5OUT ; clear bit P5.7
     MOV.B #255, P2DIR        ; set all pins from port 2 as outputs
     MOV.B #0, P2OUT          ; set port 2 to low

     ; ADC config (based on documentation)
     BIS.W #0000100011110000b, &ADC12CTL0
     ; SHT0 = 1000b (256 cycles) | MSC = 1 | REF2_5V = 1 | REFON = 1 | ADC12ON = 1
     BIS.W #0000001000000010b, &ADC12CTL1
     ; SHP = 1 | CONSEQ2 = 1 -> A1 forwarded to MEM0
     BIS.B #00000001b, &ADC12MCTL0 ; set input channel as A1
     BIS.B #10b, &P6SEL ; set P6.1 as analog input
     BIS.W #11b, &ADC12CTL0 ; has to be at the end
     ; ENC = 1 | ADC12SC = 1 -> enables and starts conversion

     ;---------- Basic Clock Module Initialisation --------------------------------------
     ; - switch from DCO to XT2
     ; - MCLK & SMCLK supplied from XT2, ACLK = n/a
     ; - the DCO is left runing
     bis.b #OSCOFF,SR ;turn OFF osc.1
     bic.b #XT2OFF,BCSCTL1 ;turn ON osc.2
BCM0 bic.b #OFIFG,&IFG1 ;clear OFIFG
     mov #0FFFFh,R15 ;delay (waiting for oscilator start)
BCM1 dec R15 ;delay
     ;jnz BCM1 ;delay -> commented out as it was ocasionally leading to infinite loop
     bit.b #OFIFG,&IFG1 ;test OFIFG
     jnz BCM0 ;repeat test if needed
     ;MCLK
     bic.b #040h,&BCSCTL2 ;slelect XT2CLK as source
     bis.b #080h,&BCSCTL2 ;
     bic.b #030h,&BCSCTL2 ;MCLK=source/1 (8MHz)
     ;SMCLK
     bis.b #SELS,&BCSCTL2 ;slelect XT2CLK as source
     bic.b #006h,&BCSCTL2 ;SMCLK=source/1 (8MHz)
     ;---

     ;... ;DAC_0 initialisation
     bis.w #REFON+REF2_5V,&ADC12CTL0 ;Reference generator ON, VRef+=2.5V
     bic #DAC12SREF0,&DAC12_0CTL ;set Vref=VREF+
     bic #DAC12SREF1,&DAC12_0CTL ;
     bic #DAC12RES,&DAC12_0CTL ;12-bit resolution
     bic #DAC12LSEL0,&DAC12_0CTL ;Load mode 0
     bic #DAC12LSEL1,&DAC12_0CTL ;
     bis #DAC12IR,&DAC12_0CTL ;Full-Scale=1xVref
     bis #DAC12AMP0,&DAC12_0CTL ;High speed amplifier output
     bis #DAC12AMP1,&DAC12_0CTL ;
     bis #DAC12AMP2,&DAC12_0CTL ;
     bic #DAC12DF,&DAC12_0CTL ;Data format - straight binary
     bic #DAC12IE,&DAC12_0CTL ;Interrupt disabled
     bis #DAC12ENC,&DAC12_0CTL ;DAC_0 conversion enabled
     ;...

     ;... ;DAC_1 initialisation
     bis.w #REFON+REF2_5V,&ADC12CTL0 ;Reference generator ON, VRef+=2.5V
     bic #DAC12SREF0,&DAC12_1CTL ;set Vref=VREF+
     bic #DAC12SREF1,&DAC12_1CTL ;
     bic #DAC12RES,&DAC12_1CTL ;12-bit resolution
     bic #DAC12LSEL0,&DAC12_1CTL ;Load mode 0
     bic #DAC12LSEL1,&DAC12_1CTL ;
     bis #DAC12IR,&DAC12_1CTL ;Full-Scale=1xVref
     bis #DAC12AMP0,&DAC12_1CTL ;High speed amplifier output
     bis #DAC12AMP1,&DAC12_1CTL ;
     bis #DAC12AMP2,&DAC12_1CTL ;
     bic #DAC12DF,&DAC12_1CTL ;Data format - straight binary
     bic #DAC12IE,&DAC12_1CTL ;Interrupt disabled
     bis #DAC12ENC,&DAC12_1CTL ;DAC_1 conversion enabled
     ;...

main:   NOP                            ; main program
     MOV.W   #WDTPW+WDTHOLD,&WDTCTL  ; Stop watchdog timer
     mov.w   #0x5,&TACCR0            ; Period for up mode
     mov.w   #CCIE,&TACCTL0          ; Enable interrupts on Compare 0
     mov.w   #MC_1|ID_3|TASSEL_2|TACLR,&TACTL
     bis.w   #GIE,SR                 ; Enable interrupts (just TACCR0)

Mainloop:
     nop                             ; Required only for debugger
     JMP $                           ; jump to current location '$'
                                     ; (endless loop)

TIMER_A0_Interrupt:
     MOV.W  &ADC12MEM0, R5           ; moving the value from ADC to R5
     MOV    R5, &DAC12_1DAT          ; moving that value to converter DAC_1
     MOV.B  R5, &P2OUT               ; moving that value to converter DAC_2
     RETI

     END
```