B ADC

This appendix documents the error in the ADC register setting regarding channel number greater than 15. The adc handle contains a structure containing, amongst other, hardware attributes for the channel. The structure member “adcPin” is the subject of the focus.

ADC-pin PK0

Expression	Type	Value
hwAttrs	struct <unnamed> *	0x0000692C
*(hwAttrs)	struct <unnamed>	{...}
adcPin	unsigned int	1073409
refVoltage	unsigned int	0
adcModule	enum <unnamed>	ADCMSP432E4_MOD0
adcSeq	enum <unnamed>	ADCMSP432E4_SEQ3

```
Name : adcPin
    Default:1073409
    Hex:0x00106101
    Decimal:1073409
    Octal:04060401
    Binary:00000000000100000110000100000001b
```

ADC-pin PK1

Expression	Type	Value
hwAttrs	struct <unnamed> *	0x0000693C
*(hwAttrs)	struct <unnamed>	{...}
adcPin	unsigned int	1138946
refVoltage	unsigned int	0
adcModule	enum <unnamed>	ADCMSP432E4_MOD0
adcSeq	enum <unnamed>	ADCMSP432E4_SEQ3

```
Name : adcPin
    Default:1138946
    Hex:0x00116102
    Decimal:1138946
    Octal:04260402
    Binary:00000000000100010110000100000010b
```

The file ADCMSP432E4.C reveals that the adc pin information holds three informations, Channel, Port and Pin for setting up the ADC registers.

```
49 #define PinConfigChannel(config) (((config) >> 16) & 0x1F)
50 #define PinConfigPort(config) (((config << 4) & 0x000FF000) | 0x40000000)
51 #define PinConfigPin(config) ((config) & 0xFF)
```

The pin PK0 is connected to the ADC channel 16.
PK0: adcPin = 0x00106101
Channel.....: 0x10 (16_d)
Port .....: 0x61
Pin.....: 0x10 (16_d)
(while PK1: adcPin = 0x00116102)

The “Channel” is used to set up the ADC MUX registers ADCSSMUX<n> and ADCSSEMUX<n> (n is the chosen sequencer number).
As can be seen in the code of ADCMSP432E4, the channel value (bit 20:16) is extracted from the adcPin. Two bits (0x40 and 0x20) is added to flag interrupt enable and end-sequence. Hence a 7-bit value is passed to the “MAP_ADCSequenceStepConfigure() which for the ADC channel 16 becomes 0x70 (with flags).

```
/* Configure ADC sequencer to take one sample at given channel*/
MAP_ADCSequenceStepConfigure(
        hwAttrs->adcModule, hwAttrs->adcSeq, 0,
        PinConfigChannel(hwAttrs->adcPin) | ADC_CTL_IE | ADC_CTL_END);
```

In the MAP_ADCSequenceStepConfigure() the parameters becomes:
ui32Base = hwAttrs->adcModule (i.e., 0x40038000, the base address of the ADC registers)
ui32SequenceNum = 3 (Chosen sequencer 3 offering only one ADC sequence)
ui32Step = 0 (Only one sample from ADC)
ui32Config = adcPin(bit 20:16) | 0x40 | 0x20 = 0x70 (where 0x10 is the channel)

As can be seen from the ADCMSP432E4 code below, the setting of ADC_SSEMUX is given bit 11:8 of the io32Config value `((ui32Config & 0xf00) >> 8)` which always will give a value of 0x0 for the ADCSSEMUX register.

```
584     //
585     // Compute the shift for the bits that control this step.
586     //
587     ui32Step *= 4;
588
589     //
590     // Set the analog mux value for this step.
591     //
592     HWREG(ui32Base + ADC_SSMUX) = ((HWREG(ui32Base + ADC_SSMUX) &
593                                     ~(0x0000000f << ui32Step)) |
594                                    ((ui32Config & 0x0f) << ui32Step));
595
596     //
597     // Set the upper bits of the analog mux value for this step.
598     //
599     HWREG(ui32Base + ADC_SSEMUX) = ((HWREG(ui32Base + ADC_SSEMUX) &
600                                      ~(0x0000000f << ui32Step)) |
601                                    (((ui32Config & 0xf00) >> 8) << ui32Step));
602
603     //
604     // Set the control value for this step.
605     //
606     HWREG(ui32Base + ADC_SSCTL) = ((HWREG(ui32Base + ADC_SSCTL) &
607                                     ~(0x0000000f << ui32Step)) |
608                                    (((ui32Config & 0xf0) >> 4) << ui32Step));
609
610     //
611     // Set the sample and hold time for this step.
612     //
613     HWREG(ui32Base + ADC_SSTSH) = ((HWREG(ui32Base + ADC_SSTSH) &
614                                     ~(0x0000000f << ui32Step)) |
615                                (((ui32Config & 0xf00000) >> 20) << ui32Step));
616
```

```
00001bf6:   9002            str     r0, [r13, #8]
 592          HWREG(ui32Base + ADC_SSMUX) = ((HWREG(ui32Base + ADC_SSMUX) &
00001bf8:   9A03            ldr     r2, [r13, #0xc]
00001bfa:   9800            ldr     r0, [r13]
00001bfc:   9C02            ldr     r4, [r13, #8]
00001bfe:   9D02            ldr     r5, [r13, #8]
00001c00:   9B00            ldr     r3, [r13]
00001c02:   6801            ldr     r1, [r0]
00001c04:   F002020F        and     r2, r2, #0xf
00001c08:   40A2            lsls    r2, r4
00001c0a:   200F            movs    r0, #0xf
00001c0c:   40A8            lsls    r0, r5
00001c0e:   4381            bics    r1, r0
00001c10:   430A            orrs    r2, r1
00001c12:   601A            str     r2, [r3]
 599          HWREG(ui32Base + ADC_SSEMUX) = ((HWREG(ui32Base + ADC_SSEMUX) &
00001c14:   9800            ldr     r0, [r13]
00001c16:   9A03            ldr     r2, [r13, #0xc]
00001c18:   9C02            ldr     r4, [r13, #8]
00001c1a:   9D02            ldr     r5, [r13, #8]
00001c1c:   9B00            ldr     r3, [r13]
00001c1e:   6981            ldr     r1, [r0, #0x18]
00001c20:   F4026270        and     r2, r2, #0xf00
00001c24:   0A12            lsrs    r2, r2, #8
00001c26:   40A2            lsls    r2, r4
00001c28:   200F            movs    r0, #0xf
00001c2a:   40A8            lsls    r0, r5
00001c2c:   4381            bics    r1, r0
00001c2e:   430A            orrs    r2, r1
00001c30:   619A            str     r2, [r3, #0x18]
 606          HWREG(ui32Base + ADC_SSCTL) = ((HWREG(ui32Base + ADC_SSCTL) &
00001c32:   9800            ldr     r0, [r13]
00001c34:   9A03            ldr     r2, [r13, #0xc]
```

```
        HWREG(ui32Base + ADC_SSEMUX) = ((HWREG(ui32Base + ADC_SSEMUX) &
00001c14:   9800      ldr   r0, [r13]         # r0 = 0x400380A0, Address of ADC_SSMUX
00001c16:   9A03      ldr   r2, [r13, #0xc]   # r2 = 0x70 (channel)
00001c18:   9C02      ldr   r4, [r13, #8]     # r4 = 0 (sequencer no. 0)
00001c1a:   9D02      ldr   r5, [r13, #8]     # r5 = 0 (sequence no. 0)
00001c1c:   9B00      ldr   r3, [r13]         # r3 = 0x400380A0, Address of ADC_SSMUX
00001c1e:   6981      ldr   r1, [r0, #0x18]   # r1 = Content of 0x400380B8 (ADC_SSEMUIX)
00001c20:   F4026270  and   r2, r2, #0xf00    # r2 = 0x70 & 0xf00 = 0(!)
00001c24:   0A12      lsrs  r2, r2, #8        # r2 >> 8
00001c26:   40A2      lsls  r2, r4            # r2<<0 (sequence number)
00001c28:   200F      movs  r0, #0xf          # r0 = 0xf
00001c2a:   40A8      lsls  r0, r5            # r0 << 0
00001c2c:   4381      bics  r1, r0            # r1 &= r0 (0xf)
00001c2e:   430A      orrs  r2, r1            # r2 |= r1 (insert r1 into r2)
00001c30:   619A      str   r2, [r3, #0x18]   # Write r2 to ADC_SSEMUX register)
```