Converting between Analog and Digital Domains

Chapter 6

Renesas Electronics America Inc.
Advanced Embedded Systems using the RX63N

Rev. 0.1

Topics

- Need
- Reference voltage
- Resolution
- Sample and Hold circuit
- Successive approximation
- Transfer function
- Conversion speed
- 12-bit ADC registers
- Operating modes
- 10-bit ADC registers
- D/A converter
- D/A converter registers

Need

- The microcontroller can process only digital data

- Are the following commonly measured quantities analog or digital ?

1. Distance
2. Weight
3. Acceleration
4. Temperature

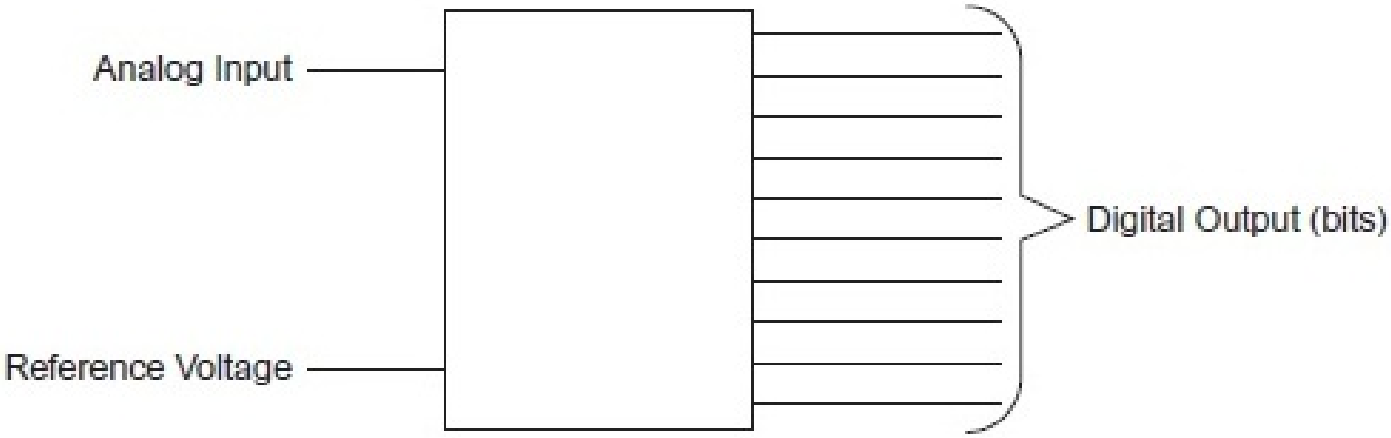
Need

- All physical quantities are analog. The world is analog !

- We need to convert these analog values to digital for the microcontroller to comprehend the value of the real analog physical quantity

Reference voltage

- The analog value is compared with a known reference voltage to obtain its digital form
- The measurement process is called *quantization*

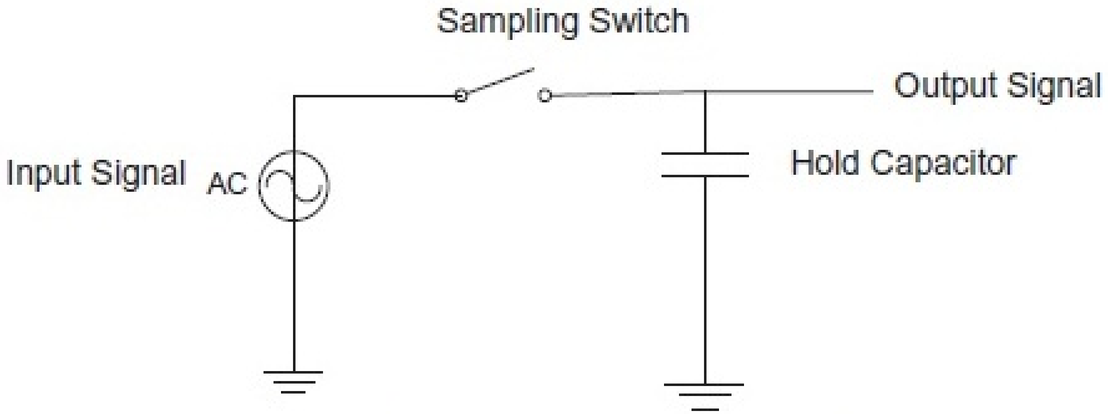


Resolution

- The number of bits in the digital output is called the resolution of the ADC
- A 10-bit A/D convertor can produce $2^{10} = 1024$ distinct digital outputs
- RX63N microcontroller has an 8 channel 10-bit and a 21 channel 12-bit A/D converter units

Sample and Hold circuit

- This circuit catches hold of the voltage to be converted to digital form
- It is helpful particularly when the input analog voltage varies very fast
- When the switch is closed, the capacitor charges to the value of analog voltage and that value is fed to the A/D converter

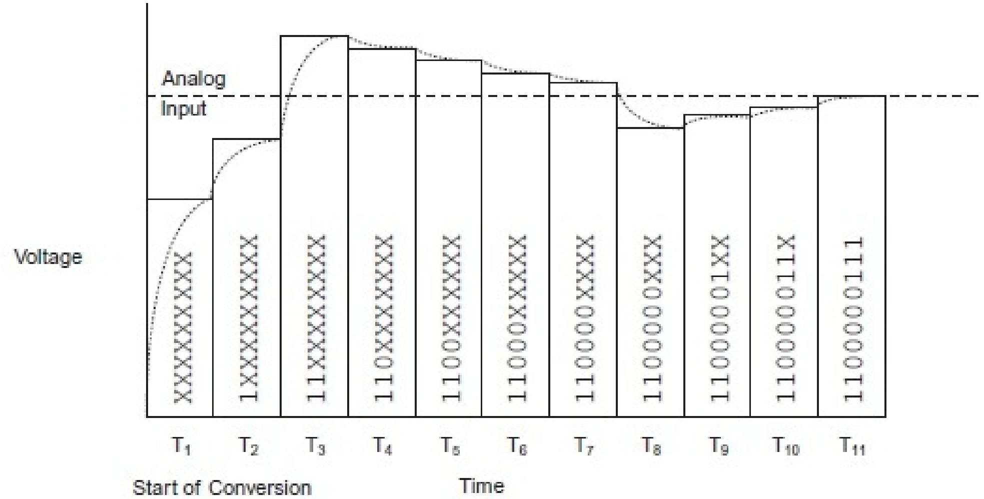


Successive Approximation

- RX63N microcontroller employs this method of conversion
- In this method, initially the microcontroller compares the analog voltage with half the reference voltage
- In each approximation step, the microcontroller halves the possible range between which the digital value lies
- In this way the microcontroller closes in on the analog value, setting 1 or 0 to the bit position starting from MSb
- Set 1 if the analog value is greater than the reference value of that step, else set to 0

Successive Approximation

- Consider 2.5V to be measured with Vref= 3.3V using 10-bit A/D converter
- First 2.5 is compared with 1.65 (mean of 0 & 3.3) . Since 2.5>1.65, our digital value is 1xxxxxxxxx
- Next compare 2.5 with 2.47 (mean of 1.65 & 3.3) . Since 2.5>2.47, our digital value is 11xxxxxxxx
- We proceed in similar way till we get the lsb of the digital form
- We compare ‘n’ times, where ‘n’ is the resolution of the A/D converter



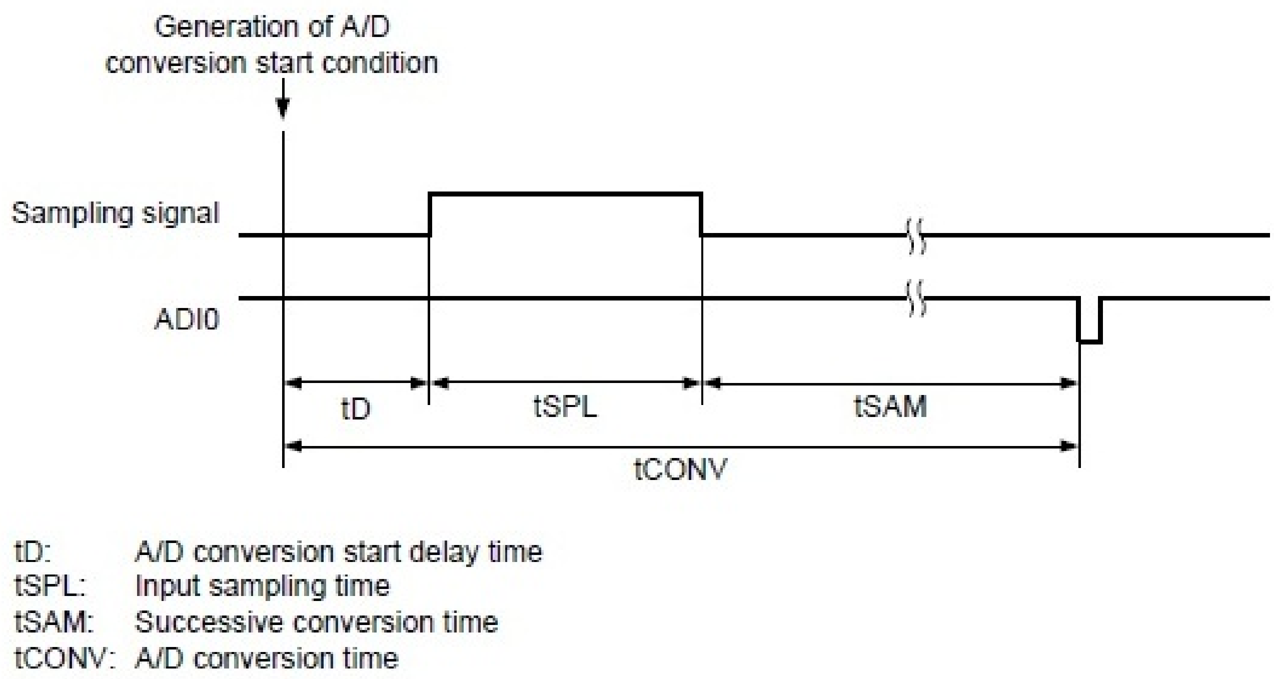
Transfer function

- n= digital output
- Vin = input analog voltage
- V+ref = upper reference voltage
- V-ref = lower reference voltage, generally zero
- N = resolution of A/D converter

$$n = \left[\frac{(V_{\text{in}} - V_{-\text{ref}})(2^N - 1)}{V_{+\text{ref}} - V_{-\text{ref}}} + \frac{1}{2} \right] \text{int}$$

Conversion speed

- Conversion speed = Start delay(tD) + input sampling time(tSPL) + conversion time (tSAM)



12-bit ADC registers

Some of the important registers are:

- A/D Data Registers (ADDRn) (n = 0 to 20)

- 16-bit register
- Holds the digital value

To use a particular channel, the respective port has to be set up as input. For example, to use AN0, port 4 pin 0 use:

```
PORT4.PDR.BIT.B0 = 0;
```

For inputs, the Port Mode Register (PMR) also has to set up. This can be done using:

```
PORT4.PMR.BIT.B0 = 1;
```

12-bit ADC registers

- A/D Control Register (ADCSR)

- Start conversion control
- Mode select
- Interrupt enable
- A/D clock speed

Address: 0008 9000h

	b7	b6	b5	b4	b3	b2	b1	b0
	ADST	ADCS	—	ADIE	CKS[1:0]		TRGE	EXTRG
Value after reset:	0	0	0	0	0	0	0	0

12-bit ADC registers

- A/D Channel Select Register (ADANSx (x=0 or 1))

  - 2 registers to select 20 channels

	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
	ANS0[15:0]															
Value after reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
	—	—	—	—	—	—	—	—	—	—	—	ANS1[4:0]				
Value after reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Operating modes

- Single cycle scan

  Performs conversion on single or multiple channels once

- Continuous scan mode

  Performs continuous conversion on single or multiple channels

ADC Initialization

```
1. void ADC_Init(){
2.     SYSTEM.MSTPCRA.BIT.MSTPA17 = 0;
3.     S12AD.ADCSR.BYTE = 0x0C;
4.     S12AD.ADANS0.WORD = 0x01;
5.     S12AD.ADCER.BIT.ACE = 1;
6.     S12AD.ADCER.BIT.ADRFMT = 0;
7. }
```

Line 2: 12-bit ADC has been selected using the Module Stop Control Register A.

Line 3: the Control Register is set: software trigger has been enabled (b1=0, b0=0), the PCLK (b3=1, b2=1) has been selected, A/D Interrupt Enable has not been enabled (b4=0) and Single-Cycle Scan mode has been selected (b6=0).

Line 4: Channel 0 (AN000) has been selected.

Line 5: automatic clearing of ADDRn

Line 6: right alignment of ADDRn is done.

Example of a ADC Initialization

```
void ADC_Init() {
   PORT4.PDR.BIT.B0 = 0;
   PORT4.PMR.BIT.B0 = 1;
   SYSTEM.MSTPCRA.BIT.MSTPA17 = 0;
   S12AD.ADCSR.BYTE = 0x0C;
   S12AD.ADANS0.WORD = 0x01;
   S12AD.ADCER.BIT.ACE = 1;
   S12AD.ADCER.BIT.ADRFMT = 0;
   S12AD.ADSTRGR.BIT.ADSTRS = 0x0;
   S12AD.ADCSR.BIT.ADST = 1;
}
```

What does each line do?

Using ADC data

```
while(1){
    if(S12AD.ADCSR.BIT.ADST == 0 && i == 0){
        ADC_out = S12AD.ADDR0 & 0X0FFF;
        sprintf(ADC_OUT,"%d",ADC_out);
        lcd_display(LCD_LINE2,ADC_OUT );
        i++;
    }
}
```

What will this code do?

In Class Exercise

How would you initialize the ADC and read the internal temperature sensor?

```
 1. void ADC_Init() {
 2.
 3.
 4.
 5.
 6.
 7.
 8.
 9.
10.
11. }
```


10-bit ADC registers

Some of the important registers are:

- A/D Data Register (ADDRn) (n = A to H)

- 16-bit register
- Holds the digital data

10-bit ADC registers

- A/D Control/Status Register (ADCSR)

  - Select the input channels
  - Start or stop A/D conversion
  - Enable or disable ADI interrupt

10-bit ADC registers

- A/D Control Register (ADCR)

- Type of A/D conversion mode
- Clock select
- Trigger select

	b7	b6	b5	b4	b3	b2	b1	b0
	TRGS[2:0]			—	CKS[1:0]		MODE[1:0]	
Value after reset:	0	0	0	0	0	0	0	0

D / A converter

- It converts a digital value stated by programmer to corresponding analog voltage on a microcontroller pin
- It may be needed for controlling other devices like motor
- RX63N has a 10-bit D/A converter which has 2 channels
- Analog value = (D/A data register value / 1024) * Vref

D/A converter registers

Some of the important registers are:

- D/A Data Register (DADRm) (m = 0, 1)

- 16-bit registers
- Holds the digital value to be converted to analog voltage

D/A converter registers

- D/A Control Register (DACR)

- Channel select
- Enable or disable D/A converter unit

	b7	b6	b5	b4	b3	b2	b1	b0
	DAOE1	DAOE0	DAE	—	—	—	—	—
Value after reset:	0	0	0	0	0	0	0	0

Example of using the DAC

```c
#include "iodefine.h"
void DAC_Init();
void main(void){
    PORT0.PDR.BIT.B5 = 1;
    PORT0.PMR.BIT.B5 = 0;
    DAC_Init();
    while(1){}
}

void DAC_Init(){
    SYSTEM.MSTPCRA.BIT.MSTPA19 = 0;
    DA.DADR1 = 102;
    DA.DACR.BYTE = 0x9F;
}
```

Conclusion

- We covered the A/D conversion concepts like transfer function, resolution, successive approximation technique
- The important control registers were also discussed
- You can now set A/D converter and D/A converter of RX63N to be used in your program

References

All images taken from:

[1] Renesas Electronics, Inc., *RX63N Group, RX631 Group User's Manual: Hardware, Rev 1.60, February* 2013

Renesas Electronics America Inc.