Trong các ứng dụng đo lường và điều khiển bằng vi điều khiển bộ chuyển đổi tương tự-số (ADC) là một thành phần rất quan trọng. Dữ liệu trong thế giới của chúng ta là các dữ liệu tương tự (analog). Ví dụ nhiệt độ không khí buổi sáng là 25oC và buổi trưa là 32oC, giữa hai mức giá trị này có vô số các giá trị liên tục mà nhiệt độ phải “đi qua” để có thể đạt mức 32oC từ 25oC, đại lượng nhiệt độ như thế gọi là một đại lượng analog.

Trong khi đó, rõ ràng vi điều khiển là một thiết bị số (digital), các giá trị mà một vi điều khiển có thể thao tác là các con số rời rạc vì thực chất chúng được tạo thành từ sự kết hợp của hai mức 0 và 1. Ví dụ chúng ta muốn dùng một thanh ghi 8 bit trong vi điều khiển để lưu lại các giá trị nhiệt độ từ 0oC đến 255 oC, như chúng ta đã biết, một thanh ghi 8 bit có thể chứa tối đa 256 (28) giá trị nguyên từ 0 đến 255, như thế các mức nhiệt độ không nguyên như 28.123 oC sẽ không được ghi lại. Nói cách khác, chúng ta đã “số hóa” (digitalize) một dữ liệu analog thành một dữ liệu digital. Quá trình “số hóa” này thường được thực hiện bởi một thiết bị gọi là “bộ chuyển đổi tương tự - số hay đơn giản là ADC (Analog to Digital Converter).



      Có rất nhiều phương pháp chuyển đổi ADC, tôi không có ý định giải thích cụ thể các nguyên lý chuyển đổi này trong bài học về AVR, tuy nhiên tôi sẽ giới thiệu một cách chuyển đổi rất cơ bản và phổ biến để các bạn phần nào nắm được cách mà một bộ ADC làm việc. Phương pháp chuyển đổi mà tôi nói là phương pháp chuyển đổi trực tiếp (direct converting)  hoặc flash ADC.  Các bộ chuyển đổi ADC theo phương pháp này được cấu thành từ một dãy các bộ so sánh (như  opamp), các bộ so sánh được mắc song song và được kết nối trực tiếp với tín hiệu analog cần chuyển đổi. Một điện áp tham chiếu (reference) và một mạch chia áp được sử dụng để tạo ra các mức điện áp so sánh khác nhau cho mỗi bộ so sánh. Hình 1 mô tả một bộ chuyển đổi flash ADC có 4 bộ so sánh, Vin là tín hiệu analog cần chuyển đổi và giá trị sau chuyển đổi là các con số tạo thành từ sự kết hợp các mức nhị phân trên các chân Vo. Trong hình 1, bạn thấy rằng do anh hưởng của mạch chia áp (các điện trở mắc nối tiếp từ điện áp +15V đến ground), điện áp trên chân âm (chân -) của các bộ so sánh sẽ khác nhau. Trong lúc chuyển đổi, giả sử điện áp Vin lớn hơn điện áp “V-“ của bộ so sánh 1 (opamp ở phía thấp nhất trong mạch) nhưng lại nhỏ hơn điện áp V- của các bộ so sánh khác, khi đó ngõ Vo1 ở mức 1 và các ngõ Vo khác ở mức 0, chúng ta thu được một kết quả số. Một cách tương tự, nếu tăng điện áp Vin ta thu được các tổ hợp số khác nhau. Với mạch điện có 4 bộ so sánh như trong hình 1, sẽ có tất cả 5 trường hợp có thể xảy ra, hay nói theo cách khác điện áp analog Vin được chia thành  5 mức số khác nhau. Tuy nhiên, bạn chú ý là các ngõ Vo không phải là các bit của tín hiệu số ngõ ra, chúng chỉ là đại diện để tổ hợp thành tín hiệu số ngõ ra, dễ hiểu hơn chúng ta không sử dụng được các bit Vo trực tiếp mà cần một bộ giải mã (decoder).  
Độ phân giải (resolution): như trong ví dụ trên, nếu mạch điện có 4 bộ so sánh, ngõ ra digital sẽ có 5 mức giá trị. Tương tự nếu mạch điện có 7 bộ so sánh thì sẽ có 8 mức giá trị có thể ở ngõ ra digital, khoảng cách giữa các mức tín hiệu  trong trường hợp 8 mức sẽ nhỏ hơn trường hợp 4 mức. Nói cách khác, mạch chuyển đổi với 7 bộ so sánh có giá trị digital ngõ ra “mịn” hơn khi chỉ có 4 bộ, độ “mịn” càng cao  tức độ phân giải (resolution) càng lớn. Khái niệm độ phân giải được dùng để chỉ số bit cần thiết để chứa hết các mức giá trị digital ngõ ra. Trong trường hợp có 8 mức giá trị ngõ ra, chúng ta cần 3 bit nhị phân để mã hóa hết các giá trị này, vì thế mạch chuyển đổi ADC với 7 bộ so sánh sẽ có độ phân giải là 3 bit. Một cách tổng quát, nếu một mạch chuyển đổi ADC có độ phân giải n bit thì sẽ có 2n mức giá trị có thể có ở ngõ ra digital. Để tạo ra một mạch chuyển đổi flash ADC có độ phân giải n bit, chúng ta cần đến 2n-1 bộ so sánh, giá trị này rất lớn khi thiết kế bộ chuyển đổi ADC có độ phân giải cao, vì thế các bộ chuyển đổi flash ADC thường có độ phân giải ít hơn 8 bit. Độ phân giải liên quan mật thiết đến chất lượng chuyển đổi ADC, việc lựa chọn độ phân giải phải phù hợp với độ chính xác yêu cầu và khả năng xử lý của bô điều khiển.  
  
Chuyển đổi ADC trên STM8  
Với STM8s103k3, ADC gồm những chân AIN0, AIN1, AIN2, AIN3  
ADC trong STM8 chỉ cần quan tâm tới 3 thanh ghi

1. ADC\_CSR
2. ADC\_CR1
3. ADC\_CR2

Thanh ghi ADC\_CSR là thanh ghi chứa các cờ và 1 vài bit để điều khiển ADC. Bao gồm: chọn kênh ADC, Cờ báo đã convert xong  
Thanh ghi ADC\_CR2 thì cần quan tâm tới bit 3 là bit ALIGN, Bít này dùng để canh chỉnh kết quả ra cho ADC  
Vì ADC của nó là 10 bit nên cần tới 2 thanh ghi 8 bit để lưu trữ kết quả, mà 2 thanh ghi 8 bit nó là 16 bit  
Và nó có thể cấu hình được 10 bit kết quả sẽ là 10 bit cao hay 10 bit thấp của 16 bit kết quả  
Nó gọi là canh chỉnh trái với canh chỉnh phải  
Nếu bit này là

* 0 là canh chỉnh trái
* 1 là canh chỉnh phải

Thanh ghi ADC\_CR1  cũng chỉ cần quan tâm tới bit 0 - bit Enable ADC và Start Convert ADC  
Khi set bit này lần đầu tiên từ khi cấp nguồn thì nó là cho phép bộ ADC hoạt động, còn sau đó nếu set tiếp nó bằng 1 thì nó lại là bắt đầu convert  
Để đọc về kết quả ADC thì làm theo trình tự như sau:

1. Start Convert
2. Chờ cờ
3. Lấy kết quả
4. Xóa cờ.

2 cái cuối có thể đổi chỗ cho nhau  
Cờ kết thúc quá trình convert nó là bit 7 trong thanh ghi ADC\_CSR - EOC: End of conversion  
  
Code ví dụ:  
*Khởi tạo ADC:*

void ADC\_Init(){

  ADC\_CSR = 2;     *// Channel AIN2*

  ADC\_CR2 = 0x08;     *// Canh chỉnh phải*

  ADC\_CR1 = 0x01;     *// Cho phép ADC*

}

*Đọc kết quả:*

unsigned int Get\_ADC(){

  unsigned int result;

  ADC\_CR1 = 0x01;  *// Bắt đầu Convert*

  while(!(ADC\_CSR & 0x80));   *// chờ cở EOC*

  result = ADC\_DRL;      *// Lấy 8 bit thấp*

  result |= ADC\_DRH<<8;  lấy 8 bit cao

  ADC\_CSR &= 0x7F;  *// Xóa cờ*

  return result;

}

Để giảm thời gian phải chờ cờ thì quy trình đọc ADC có thể làm ngược lại như sau:

1. Chờ cờ
2. Xóa cờ
3. Lấy kết quả.
4. Bắt đầu đọc ADC

Cái này yêu cầu ADC phải đc Bắt đầu đọc từ trước đó, Có thể đặt trong hàm khởi tạo  
Kết quả nó trả về thực chất là kết quả của lần đọc trước đó  
Nếu khoảng thời gian giữa 2 lần gọi hàm đủ dài để ADC đọc kết quả xong thì có thể bỏ hẳn đoạn chờ cờ đi được  
Mục định của kiểu đọc ngược này là để chip thực hiện các lệnh khác thay vì chả làm gì ngoài việc phải chờ cho ADC lấy kết quả xong