Appendix 1: Microcontroller Peripheral Functions

```
// Peripheral Functions Definitions-----------------------------
```

```
//******** ADC_Init ***************

// Initialize ADC

void ADC_Init(void){

  ATDCTL2 = 0x80; // enable ADC

}
```

```
//******** ADC_In ***************

// perform 4 ADC samples on chan return average 4 results

unsigned char ADC_In(unsigned char chan){

  ATDCTL5 = chan;                 // start sequence

  while((ATDSTAT0&0x80)==0){};   // wait for SCF

  return (ATDDR0H+ATDDR1H+ATDDR2H+ATDDR3H)/4;

}
```

```
//******** SPI_Init ***************
// Initialize SPI interface make.
void SPI_Init(void){     // PS7=_CS=1
  DDRS = 0xE0;           // PS6=CLK=SPI clock out
  PTS = 0x50;          // PS5=SRI=SPI master data out
/* bit SP0CR1
 7 SPIE = 0   no interrupts
 6 SPE  = 1   enable SPI
 5 SPTIE = 0   disable transmit interruptregular outputs
 4 MSTR = 1   master
 3 CPOL = 0   output changes on +ve edge
 2 CPHA = 0   and input clocked in on activation of CS
 1 SSOE = 0   PS7 regular output
 0 LSBF = 0   most significant bit first */
  SPICR1 = 0x50;
/* bit SP0CR2
 3 PUPS = 0   no internal pullups
 2 RDS  = 0   regular drive
 4 MODFEN = 0 Don't use SS
 3 BIDROE = 0
 1 SPISWAI =0 operate normally in wait mode
 0 SPC0 = 0   normal mode */
  SPICR2 = 0x00;
  SPIBR = 0x02; // 3.125 MHz
  WOMS = 0x00;    // Enable Port S Pull-Up
}
//******** SPI_Out8 ***************
```

```
// Output 8-bit data using SPI port

void SPI_Out8(unsigned char code){

  unsigned char dummy;

  PTS_PTS7 = 0;             // PS7=_CS=0

  while((SPISR_SPTEF)==0);    // wait until write is permissible




  SPIDR = code;             // data

  while((SPISR_SPIF)==0);    // wait until write operation is
complete




  dummy = SPIDR;            // clear the SPIF flag

  PTS_PTS7 = 1;             // PS7=_CS=1


}
```

```
//******** SPI_Out16 ***************
```

```
// Output 2 8-bit data using SPI port

void SPI_Out16(unsigned char code1, unsigned char code2){

  unsigned char dummy;

  PTS_PTS7 = 0;               // PS7=_CS=0


  while((SPISR_SPTEF)==0);    // wait until write is permissible




  SPIDR = code1;             // 1st data

  while((SPISR_SPIF)==0);    // wait until write operation is
complete




  dummy = SPIDR;            // clear the SPIF flag


  while((SPISR_SPTEF)==0);    // wait until write is permissible


  SPIDR = code2;             // 2nd data

  while((SPISR_SPIF)==0);    // wait until write operation is
complete

   dummy = SPIDR;            // clear the SPIF flag

   PTS_PTS7 = 1;             // PS7=_CS=1


}
```

```
//******** MAX528_Init ***************
```

```
// Initialize MAX528 interface
void MAX528_Init(void){
  SPI_Out16(0x00,0x80);  // Set all channels to no buffer mode


}
```

```
//******** MAX528_Out ***************
// Set required analog voltage to required channel.
void MAX528_Out(unsigned char channelNo , unsigned char value){
  unsigned char code1, temp      = 0x01;
      code1      = temp<<channelNo;
  SPI_Out16(code1,value);
}
```

```
//********* MAX528_NOP ***************
// No Operation
void MAX528_NOP() {
  SPI_Out16(0x00,0x00);  // NOP
}
```

```
//********* MAX528_NOP ***************
// No Operation
void MAX528_NOP() {
  SPI_Out16(0x00,0x00);  // NOP
}
```