Chapter 3:
Microcontroller and Data Acquisition

3.1 What is a Local Controller?

Any process needed to be monitored and controlled should have a **local controller** connected to it. The local controller does two main jobs:

- **Data Acquisition**
- **Control**

The **Data Acquisition** module samples the process data every specified period of time and keeps it ready for any monitoring requests to the process, and the **Control** module takes benefit of the data collected by the Data Acquisition module and calculates the appropriate control action according to the user previously defined configuration of the controller.

In this project both the Data Acquisition and the Control functions were done at ONE module: **Freescale DEMO9S12NE64** which is based on the MC9S12NE64 microcontroller unit (MCU).

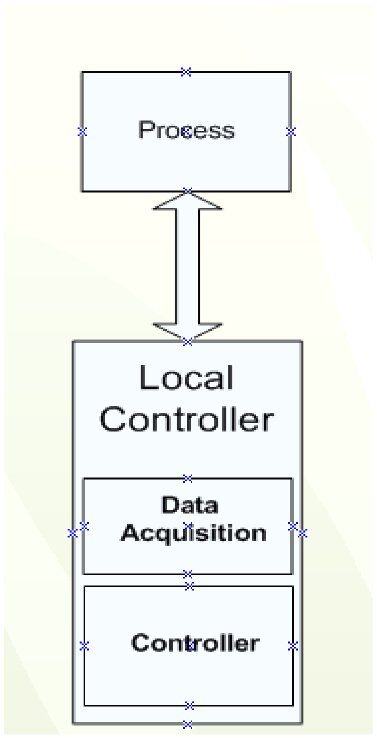


Figure 3.1 Local controller and a process.

3.2 Data Acquisition

DAQ is an abbreviation for **Data AcQuisition**. Though there have been a variety of different abbreviations used over the years (e.g. DAS, DAC, DataAcq, etc), the three letters D-A-Q truly have risen to the level of a universally accepted synonym for its much longer brother. Usage of the term DAQ has expanded to include much more than purely analog input. It is now used to describe an extremely wide variety of system tasks including analog input, analog output, digital input and output, counter/timer and motion monitoring.

Data acquisition is widely used in many areas of industry. Data acquisition is used to acquire data from sensors and other sources under computer control and bring the data together and store and manipulate it. In view of the wide variety of signals and parameters that can be sampled and stored, data acquisition involves many techniques and skills. There are many different components to a data acquisition system including sensors, communication links, signal processors, computers, databases, data acquisition software, etc. All these items have to operate together to make a successful data acquisition system.

Data acquisition systems can take many forms from very simple manual systems to high complicated computer controlled ones. The simplest form may be a technician manually logging information such as the temperature of an oven. However this form of data acquisition has its limitations. Not only is it expensive because of the fact that someone has to be available to take the measurements, but being manual it can be subject to errors. Readings may not be taken at the prescribed times, and also there can be errors resulting from the manual fashion in which the readings are taken. As can be imagined the problems become worse if a large number of readings need to be taken, as timing may become more of an issue, along with the volume of work required.

To overcome this, the simple answer is to use computer control to perform the data acquisition. As a result a definition of what is normally taken to be data acquisition is gathering information in an automated fashion from analogue and digital measurement sources, i.e. sensors and devices under test. The sensors may range from thermocouples and voltage and current sensors to strain gauges and displacement gauges and much more.

Data Acquisition Measurements

Data acquisition systems may make any number of a huge variety of measurements. These measurements typically measure analogue. Before they can be transferred into any computer system they need to be in a digital format.

Although a huge number of data acquisition measurements can be made, they basically boil down to a very few basic elements:

- Voltage
- Digital signals
- Frequency or time interval

The sensors that are used in data acquisition measurements often return values of voltage in particular that can then be converted to the values of displacement, temperature, or whatever is being measured.

Data Acquisition stage is essential as it prepare the data collected from the process and put them in a form that could be used in the controller stage, noticing that the collected data could have one of two usages:

- Sent to the Software application for monitoring
- Processed and used in taking the right control action

In this project, the process used is an R-C Circuit that is equivalent to some real processes (such as tank systems or heater systems); which have equivalent transfer functions.

For simplicity the sensor's role that transforms the real property into an equivalent electrical signal was canceled and therefore we could say that the input/output to/from the process is pure analog signal.

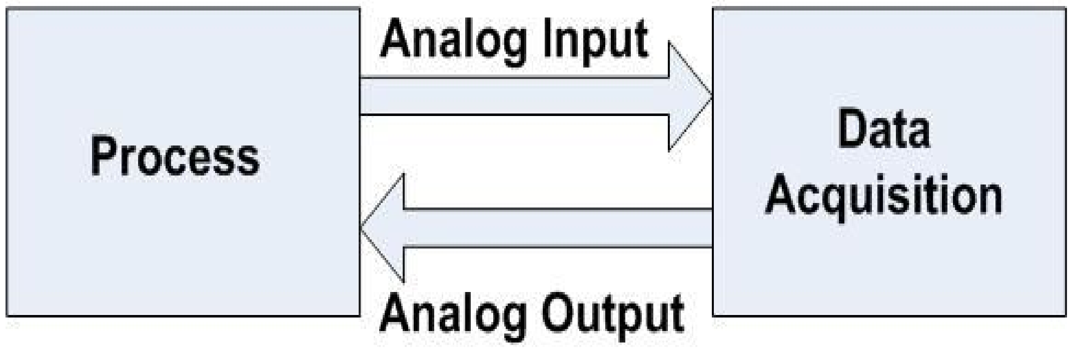


Figure 3.2 The interface between a process and a data acquisition card is mainly analog.

The **Analog Input** to the Data Acquisition is the data which needs to be digitized (sampled) first using the Analog-To-Digital Converter (ADC) and then enter the control stage which send these sampled digital data over the network for monitoring and/or processing to do the control action according to the commands specified.

The **Analog Output** from the Data Acquisition is the control action that first was in the form of digital signal and then passed over the Digital-To-Analog Converter (DAC) to convert it to analog form so as to be used to derive the process and regulate its behavior.

NOTE:
DAC is used in most types of control algorithms excluding On/Off control since its output is either high or low. Therefore, On/Off control uses digital output directly from the microcontroller.

Data Acquisition Functions:

1. Collect data of variables of interest by sampling them every determined period of time and keep them ready for any monitoring requests.

2. Perform any digital or analog conversions needed

3. Communicates with the controllers so they can benefit from the collected data in their control tasks, and configures the controllers to achieve the desired control also interfaces the controllers digital outputs to the processes analog input.

4. Interfaces and communicates somehow with the software application, giving it digital data for monitoring purposes.

For the **Data Acquisition** to perform its tasks it needs the following **sub-modules**:

1. ***Analog-to-Digital*** Conversion sub-module to collect the analog data and convert them to digital form so as to be sent to the monitoring software and also to be used in digital controllers' algorithms applied on the microcontroller.

2. ***Digital-to-Analog*** Conversion sub-module to interface the controllers digital output to the processed analog input (control action).

3.3 HCS12NE64 Microcontroller

As mentioned before, that **"Freescale DEMO9S12NE64"** which is based on the **MC9S12NE64 microcontroller unit** (MCU) was used to do the Data Acquisition and the Control tasks. Let us investigate it in more details.

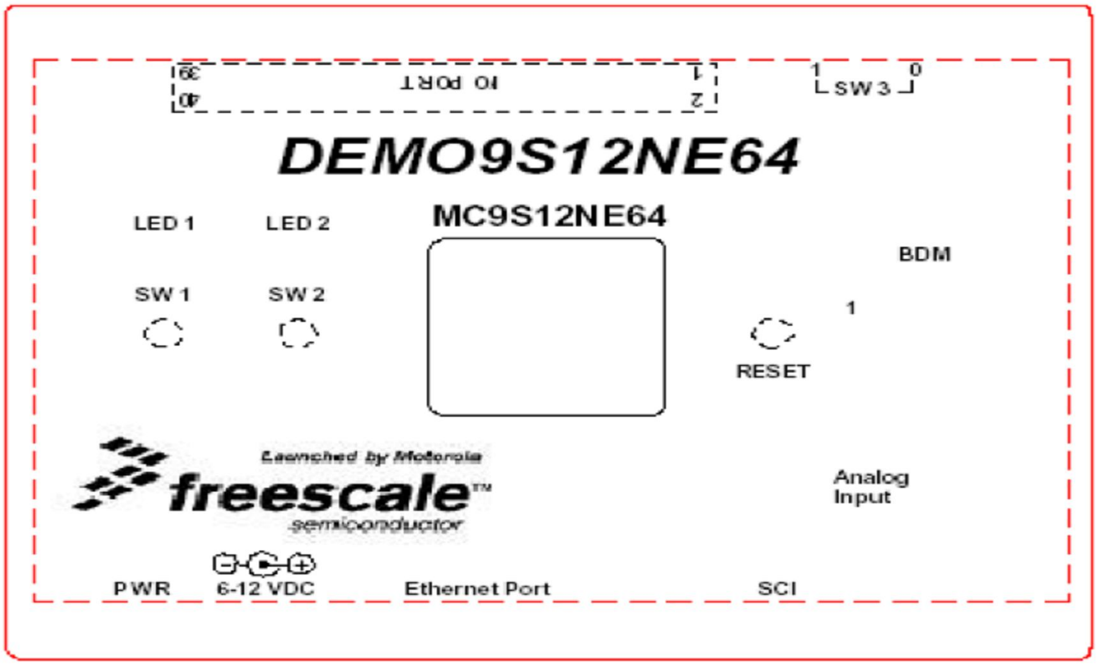


Figure 3.3 The enclosing case of DEMO9S12NE64.

Characteristic	Specification
Maximum Clock Speed	25 MHz at 3.3V (12.5 MHz bus)
Temperature Operating Storage	 -10°C to +50°C -40°C to +85°C
MCU Extension I/O	HCMOS Compatible at 3.3V
Relative Humidity	0 to 90% (non-condensing)
Power Requirements	6 to 12VDC 0.75 Amp supplied externally. A barrel type power plug is required with an outside diameter of 5.5 mm and inside diameter of 2.1 mm.
Dimensions	3.0 x 4.5 x 1.25 inches

Table 3.1 Details of the Demo Board. DEMO9S12NE64.

The Microcontroller's Main Features

- ✓ 16-bit HCS12 core
  - HCS12 CPU
    - Upward compatible with M68HC11 instruction set
    - Interrupt stacking and programmer's model identical to M68HC11
  - Memory map and interface (MMC)
  - Background debug mode (BDM)
- ✓ Memory
  - 64K bytes of FLASH EEPROM
  - 8K bytes of RAM
- ✓ ***Analog-to-Digital converter (ATD)***
  - One 8-channel module with 10-bit resolution
  - External conversion trigger capability
- ✓ ***Timer module (TIM)***
  - 4-channel timer
  - Each channel configurable as either input capture or output compare
  - Simple PWM mode
  - 16-bit pulse accumulator
- ✓ ***Serial interfaces***
  - Two asynchronous serial communications interface (SCI)
  - ***One synchronous serial peripheral interface (SPI)***
  - One inter-IC bus (IIC)
- ✓ Operating frequency
  - 50 MHZ equivalent to 25 MHZ bus speed for single chip
  - 32 MHZ equivalent to 16 MHZ bus speed in expanded bus modes
- ✓ Internal 2.5-V regulator
  - Supports an input voltage range from 3.3 V (+/-)5%
  - Low-power mode capability
  - Includes low-voltage reset (LVR) circuitry
- ✓ Development support
  - Single-wire background debug™ mode (BDM)

- On-chip hardware breakpoints
- Enhanced DBG debug features

In this project, Freescale's CodeWarrior IDE was used to compile, debug and burn code in to the microcontroller.

Analog-to-Digital converter (ATD)

The ATD that is part of the Data Acquisition is a built-in module in the MCU. The A/D input is programmable and up to 8 analog channels for the digitizing process could be handled.

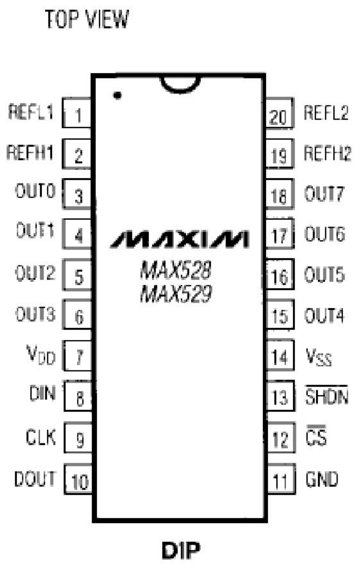
The MCU supports an internal 10 Bit Resolution but it was adjusted so as to be only 8 bits Resolution (1 byte), to ease the calculations, compromise data storage.

3.4 Digital-to-Analog converter (DAC)

This sub-module is needed in the Data Acquisition to convert from Digital to Analog, but it is not included in the MCU.

After a little search, a D/A was chosen that supports many great features: the MAX528 from MAXIM INC.

DAC Pin Configuration



Pin	Function
CS Active-Low Chip Select	When asserted low, this input pin initializes the 528 to start a new frame of serial data. When asserted high, the 16-bit data is latched and the internal shift register is turned off. The DAC registers also are updated with the new data.
DOUT Serial Data Out	This open drain pin serves as the serial output data from the DIN pin.
DIN Serial Data In	This pin serves as the input data line that receives the 16-bit serial data stream.
CLK Serial Data Clock	This pin is an input that drives the serial transmission lines.
SHDN Shutdown	Connect this input pin high for normal operation. Connect it low to conserve power.

Table 3.2 Pin functions of MAX528.

General Description

The MAX528 are monolithic devices combining an octal 8-bit, digital-to-analog converter (DAC), 8 output buffers, and serial-interface logic in space-saving shrink small outline package (SSOP).

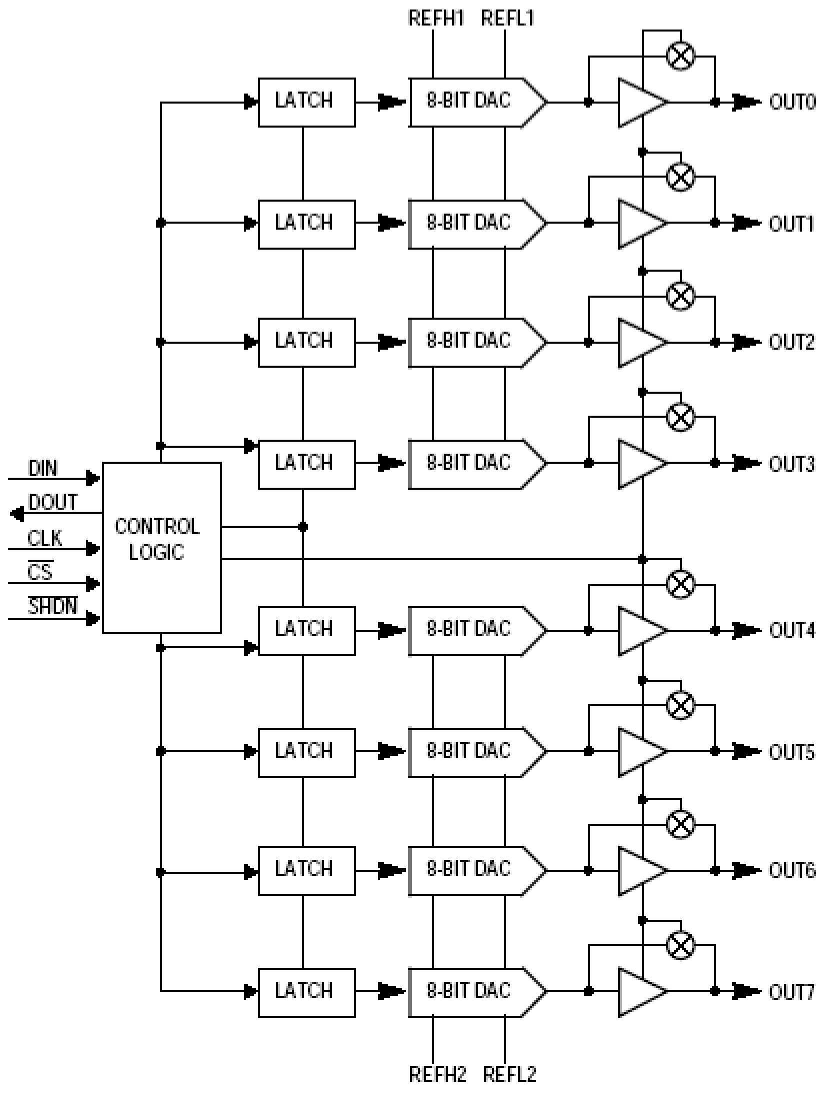


Figure 3.4 Block diagram for MAX528.

Three output modes are serially programmable for each pair of 8 analog outputs:

a) Unbuffered Mode: Connects the internal R-2R DAC network directly to the output pin, reducing power consumption and avoiding the buffer's DC errors.

b) Full-Buffered Mode: Inserts a buffer between the R-2R network and the output, providing +5mA/-2mA output drive.

c) Half-Buffered Mode: It is similar to the Full-Buffered Mode, but uses less power while still providing up to 15mA of output drive in a unipolar output configuration.

Serial data can be "daisy-chained" from one device to another. On power-up, all data bits are reset to 0, and analog outputs enter the unbuffered mode.

DAC Output Range

The MAX528 provides 8 voltage outputs (OUT0 – OUT7) from 2 reference inputs. Each reference voltage has 2 pins; REFH and REFL . The OUT0 – OUT3 output voltages are derived from REFH1 and REFL1, while OUT4 – OUT7 are derived from REFH2 and REFL2.

For each reference, REFH must be more positive than REFL . A DAC output voltage is the product of its programmed 8-bit code and its reference input voltage. For example, the output (analog) voltage of OUT5 is:

OUT5 = (REFH2 – REFL2) (nn/256 + REFL2)

Where nn = 8-bit code for OUT5, with a range of 0 – 255(00 to FF hex)

DAC Programming

The MAX528 are programmed by 16 data bits in two 8-bit bytes, the address pointer (A7 – A0) followed by the data byte (D7 – D0). These bits enter a shift register serially through DIN (pin8) A7 first, and D0 last. The data exits the DOUT (pin10) 16 clocks later in the same order.

Address pointer bits								*Data byte*							
A7	A6	A5	A4	A3	A2	A1	A0	D7	D6	D5	D4	D3	D2	D1	D0

Data at DIN is shifted into the first register (while all 16 register bits shift forward one stage) on the rising CLK edge, while holding _CS low and _SHDN high. This must occur 16 times to load all the data bits into the shift registers. On the rising edge of _CS, data in the 16 shift registers is transferred as addressed and CLK disabled.

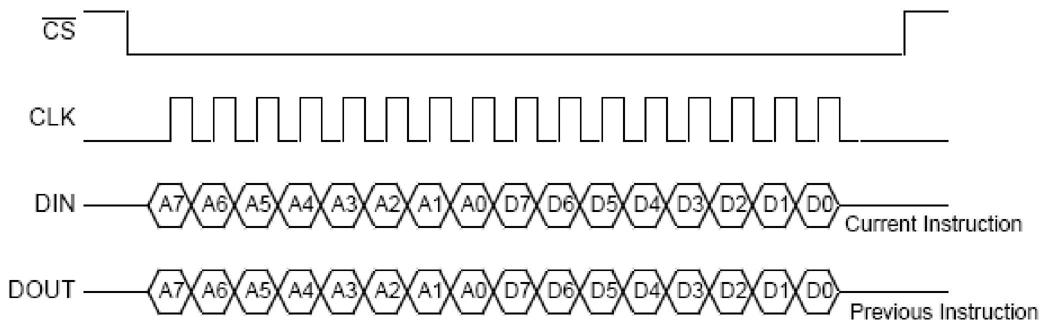


Figure 3.5 Timing diagram for MAX528.

Set Buffer Modes

Set buffer modes is implemented when all 8 address pointer bits (A7 – A0) are logic 0 and data bit D7 is 1. When this instruction is received, data bits D5 – D0 (ignoring D6) are transferred to the mode registers only; and the DAC registers are unchanged.

Enabling and disabling the 8 buffers is done in four pairs by data bits {D1, D2, D4, D5}. D1 controls buffers 6 and 7, D2 controls buffered 4 and 5, D4 controls buffers 2 and 3 and D5 controls buffers 0 and 1. A logic 1 enables a buffer pair (full-buffered or half buffered mode)l a logic 9 disables a buffer pair (unbuffered mode).

Full-buffered and half-buffered modes are set by two data bits, D0 and D3. D0 controls OUT4 through OUT7; D3 controls OUT0 through OUT3. A logic 1 enables full-buffered mode; a logic 0 enables half-buffered mode. These data bits apply only when buffer output pairs are enabled by a 1 in D1, D2, D4, or D5.

Mode	OUT 0,1	OUT 2,3	OUT 4,5	OUT 6,7
Unbuffered (D0 = D3 = X)	D5 = 0	D4 = 0	D2 = 0	D1 = 0
Half-buffered	D5 = 1	D4 = 1	D2 = 1	D1 = 1
	D3 = 0		D0 = 0	
Full-buffered	D5 = 1	D4 = 1	D2 = 1	D1 = 1
	D3 = 1		D0 = 1	

Table 3.3 Buffer mode programming.

3.5 SPI Module

The **Serial Peripheral Interface** Bus or **SPI** bus is a synchronous serial data link standard named by Motorola that operates in full duplex mode. Devices communicate in master/slave mode where the master device initiates the data frame. Multiple slave devices are allowed with individual slave select (chip select) lines. Sometimes SPI is called a "four wire" serial bus, contrasting with three, two, and one wire serial busses. It has the advantage over SCI (Serial Communication Interface) or UART for its very bit rate since its clock may be as high as 70 MHz while the maximum data rate of UART is about 100 kbps only.

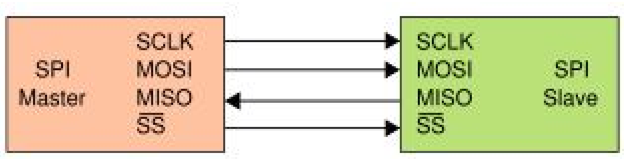


Figure 3.4 SPI bus: single-master single-slave.

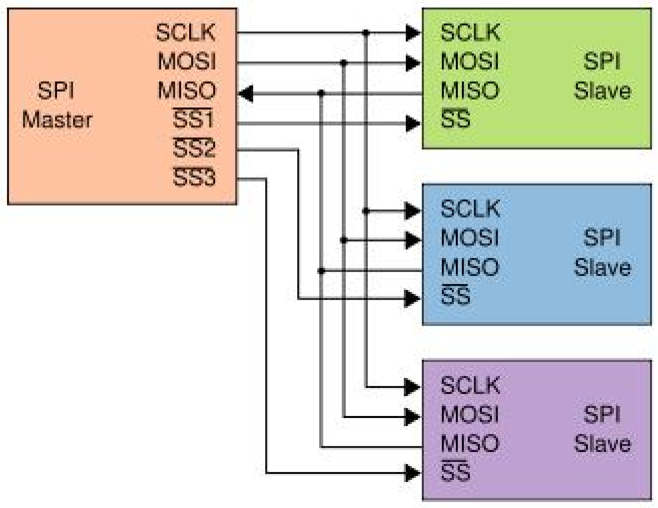


Figure 3.6 SPI bus: single-master multiple-slave.

Advantages	Disadvantages
Full duplex communication	Requires more pins on IC packages than I²C: -Half duplex "3-wire" mode uses one less pin per slave (possible in newer controllers with bidirectional mode; some slaves, like EEPROMs, tristate their outputs when receiving data from the master and don't care about the input when sending data back) -No in-band addressing protocol, so out-of-band chip select signals are required
Higher throughput than I²C	No hardware flow control
Complete protocol flexibility for the bits transferred	No slave acknowledgment (the master could be "talking" to nothing and not know it)
Extremely simple hardware interfacing	Multi-master busses are rare and awkward, and are usually limited to a single slave
Uses many fewer pins on IC packages, and wires in board layouts or connectors, than parallel interfaces	

Table 3.4 Advantages and disadvantages of SPI.

As was mentioned before that there exist a built-in SPI module in the MCU, and this will be used to interface with the DAC (MAX528) module so as to input the digital data through pin-8 DIN in the DAC.

- **The SPI supports:**

  - Full duplex
  - Synchronous serial transmission
  - Serial Communication with peripheral devices

- **Registers used:**
  - SP0CR1: (Control Register 1)
  - SP0CR2: (Control Register 2)
  - SP0BR: (Baud Rate)
  - SP0SR: (Status Register)
  - SP0DR: (SPI data Register)
  - PORTS: (port s)
  - DDRS: (Data Direction register)

Pin	Function
SCK Serial Data Clock	The SCK signal is used to synchronize the movement of data in and out of the SPI module. This pin is an output or an input depending on whether the SPI is configured as a master or a slave. Data is shifted on one side of the clock edge and sampled on the other. The SCK signal can be configured to accommodate different serial peripheral bus structures.
MOSI Master Output, Slave Input	When the SPI is configured as a master, this pin is used as an output to shift the 8-bit serial data out with the most significant bit first. The pin is used as a slave data input when the SPI is configured as a slave.
MISO Master Input, Slave Output	If the SPI is configured as a master, this pin is utilized as an input. When the SPI is in slave mode, the pin is used as an output.
SS Slave Select	When the SPI is a slave, this pin enables the SPI for an incoming transfer. As a master, this pin should be tied high.

Table 3.5 Pin functions of SPI module of HCS12.

Functional Description:

The SPI has two modes of operations:

1 – Master Mode: in which the SPI generates the synchronizing clock and initiates the transmissions. In our project, we use this mode to communicate with the MAX528 DAC.

2- Slave Mode: in which the SPI depends on the Master peripheral to generate the synchronizing clock and initiates transmissions
The SPI operates in Master Mode by setting the MSTR bit in SPICR1.

Baud Rate Generation

The P-clock divisor is selected by the SPIBR[2:1:0] between 2,4,8,16,32,64,128,256 it controls the rate of the shift register. Through the SCK pin PORTS(6), the selected clock signal also controls the rate of the shift register of the slave SPI or other slave peripheral.

Operation Brief Description

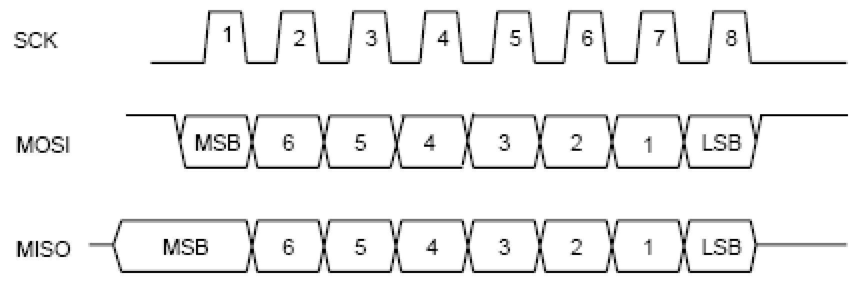


Figure 3.7 SPI timing diagram.

The SPI operates in master mode when the MSTR bit is set. Only a master SPI module can initiate transmissions. A transmission begins by writing to the master SPI Data Register. If the shift register is empty, the byte immediately transfers to the shift register. The byte begins shifting out on the MOSI pin under the control of the serial clock.

When a write to the SPI Data Register in the master occurs, there is a half SCK-cycle delay. After the delay, SCK is started within the master. The rest of the transfer operation differs slightly, depending on the clock format specified by the SPI clock phase bit, CPHA, in SPI Control Register 1