SERIAL PERIPHERAL INTERFACE (SPI)

The SPI bus provides the mechanism for all digital control of the GC0801. Each SPI register is 8-bit wide, and each register contains control bits, status monitors, or other settings that control all functions of the device. The following sections explain the specifics of this interface.

SPI FUNCTIONAL LAYER

The SPI bus can be configured by setting the bit values in SPI register 0x000. Register 0x000 is symmetrical; that is, D7 is equivalent to D0, D6 is equivalent to D1, and D5 is equivalent to D2 (D4 and D3 are unused). The device powers up in its default mode (MSB-first addressing), but can accept an LSB-first write to 0x000 because of this symmetry. The symmetrical bits are ORed together, so setting one bit sets both in the pair. The bit order is MSB-first when D5 and D2 are left clear, while bit order is swapped to LSB-first when these bits are set. Once properly configured, all subsequent register writes must follow the selected format.

The bus is configured as a 4-wire interface by default.

Each SPI bus signal is described in the following sections.

SPI_ENB

SPI_ENB is the bus enable signal driven from the BBP to the GC0801. SPI_ENB is driven low before the first SPI_CLK rising edge and is normally driven high again after the last SPI_CLK falling edge. The GC0801 ignores the clock and data signals while SPI_ENB is high.

The SPI_DO and SPI_DI pins transition to a high impedance state when this input is high. If driven high during any communications cycle, that cycle is suspended until SPI_ENB is reactivated low.

SPI_CLK

SPI_CLK is the interface reference clock driven by the BBP to the GC0801. It is only active while SPI_ENB is low. The maximum SPI_CLK frequency is 50 MHz. But we recommend 20MHz SPI_CLK frequency for normal operation.

SPI_DI, SPI_DO and SPI_DIO

When configured as a 4-wire bus, the SPI utilizes two data signals – SPI_DI and SPI_DO. SPI_DI is the data input line driven from the BBP to the GC0801 and SPI_DO is the data output from the GC0801 to the BBP in this configuration.

The data signals are launched on the rising edge of SPI_CLK and sampled on the falling edge of SPI_CLK by both the BBP and the GC0801. SPI_DI (or SPI_DIO) carries the control field from BBP to the GC0801 during all transactions and the write data fields during a write transaction. SPI_DO (or SPI_DIO) carries the returning read data fields from the GC0801 to BBP during a read transaction.

The GC0801 does not provide any weak pull-ups or pull-downs on these pins. When SPI_DO is inactive, it is floated in a high impedance state. If a valid logic state on SPI_DO is required at all times, an external weak pull-up/down should be added on the PCB.

SPI DATA TRANSFER PROTOCOL

The GC0801 SPI is a flexible, synchronous serial communications bus allowing seamless interfacing to many industry standard microcontrollers and microprocessors. The GC0801 cannot be used to control other devices on the bus – it only operates as a slave.

There are two phases to a communication cycle. Phase 1 is the control cycle, which is the writing of a control word into the GC0801. The control word provides the GC0801 serial port controller with information regarding the data field transfer cycle, which is Phase 2 of the communication cycle. The Phase 1 control field defines whether the upcoming data transfer is read or write. It also defines the register address being accessed.

Phase 1 Instruction Format

The 16-bit control field contains the following information:

MSB	D14	D13	D12	D11:D0
W/Rb	NB2	NB1	NB0	A<11:0>

W/Rb — Bit 15 of the instruction word determines whether a read or write data transfer occurs after the instruction byte write. Logic high indicates a write operation; logic zero indicates a read operation.

D11:D0 – Bits <11:0> specify the starting byte address for the data transfer during Phase 2 of the IO operation.

All byte addresses, both starting and internally generated addresses, are assumed to be valid. That is, if an invalid address (undefined register) is accessed, the IO operation continues as if the address space were valid. For write operations, the written bits are discarded, and read operations result in logic zeros at the output.

Single-Byte Data Transfer

When NB2, NB1, and NB0 are all zero, a single-byte data transfer is selected. In this scenario, the next eight bits to follow the address bits contain the data being written to or read from the GC0801 register. Once the final bit is transferred, the data signals return to their idle states and the SPI_ENB signal goes high to end the communication session.

Example: LSB-first Multi-byte Transfer

To complete a 4 byte write, starting at register address 0x02A in LSB first format, apply an instruction word of 010101000000_110_1 (binary). This instruction directs the GC0801 SPI controller to perform a write transfer of four bytes with the starting byte address of 0x02A. After the first data byte is written, the internal byte address generation logic increments to 0x02B, which is the destination of the second byte. After the second byte is written, the internal byte address generation logic increments to 0x02C, which is the destination of the third byte. After the third byte is written, the internal byte address generation logic increments to 0x02D, which is the destination of the last byte. After the fourth byte is written, the IO communication cycle is complete and the next 16 falling clock cycles on SPI_CLK are utilized to clock in the next instruction word. If no further communication is needed, the data signals return to their idle states, SPI_CLK goes low, and the SPI_ENB signal goes high to end the communication session.

TIMING DIAGRAMS

The following diagrams in Figure 1 and Figure 2 detail the SPI bus waveforms for a single-register write operation and a single-register read operation, respectively. In the first figure, the value 0x55 is written to register 0x15A. In the second value, register 0x15A is read and the value returned by the device is 0x55. If the same operations were performed with a 3-wire bus, the SPI_DO line in Figure 1 would be eliminated, and the SPI_DI and SPI_DO lines in Figure 2 would be combined on the SPI_DI line.

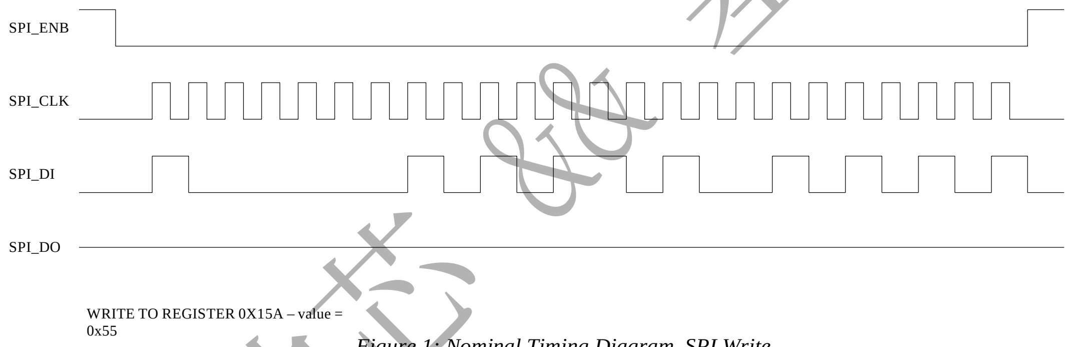

Figure 1: Nominal Timing Diagram, SPI Write

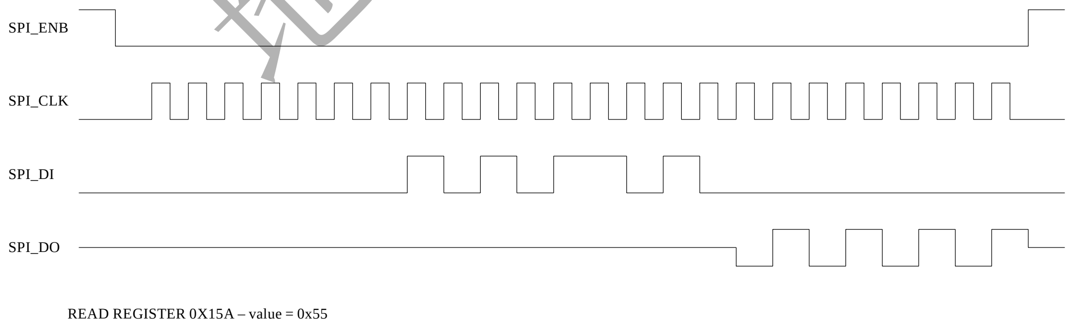

Figure 2: Nominal Timing Diagram, SPI Read

Table 1 lists the timing specifications for the SPI bus. The relationship between these parameters is shown in Figure 3.

Table 1: SPI Bus Timing Constraint Values

Parameter	Min	Typical	Max	Description
t_{CP}		48ns		SPI_CLK cycle time (clock period)
t_{MP}		25ns		SPI_CLK pulse width
t_{SC}		25ns		SPI_ENB setup time to first SPI_CLK rising edge
t_{HC}		25ns		Last SPI_CLK falling edge to SPI_ENB hold
t_{S}		23ns		SPI_DI data input setup time to SPI_CLK
t_{H}		25ns		SPI_DI data input hold time to SPI_CLK
t_{CO}		3ns		SPI_CLK rising edge to output data delay (4-wire mode)
t_{HZM}		46ns		Bus turnaround time after BBP drives the last address bit
t_{HZS}		3ns		Bus turnaround time after GC0801 drives the last data bit

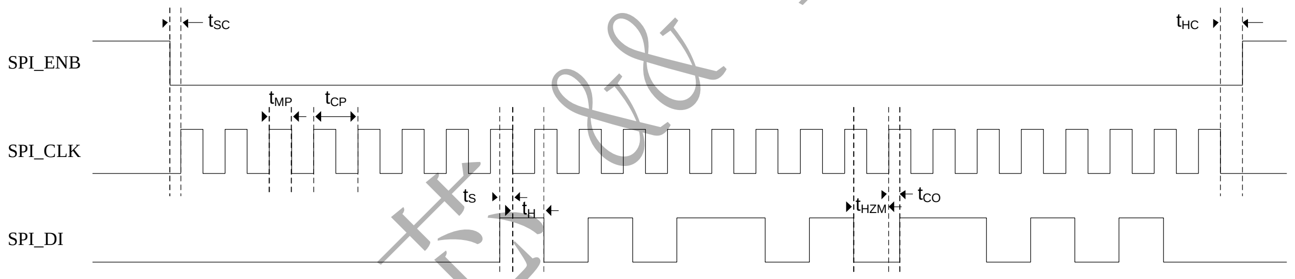

Figure 3: SPI Timing with Parameter Labels