GOLDI Axis Portal V2

Document version:	2
Hardware Version	V4.00.00
Release date:	30.09.2023

Overview:

The GOLDi Axis Portal V2 FPGA Control Unit is the main digital processing unit used in the GOLDi Axis Portal V2 to gather and control the sensor and actuator signals used in the physical model. The control unit uses a Lattice MachX02 FPGA to operate the system and provides a standardized SPI interface for the microcontroller unit to access the model's peripheral elements through a set of registers implemented in the FPGA.

Calibration procedure:

The GOLDi Axis Portal V2 does not require a specific calibration process to properly operate the motors or the internal protection system that limits the crane movement. The initial configuration of the stepper motor drivers of the X- and Y-axes is performed automatically and can be modified in the GOLDI_MODULE_CONFIG package or dynamically through the SPI interfaces in the corresponding sub-module. However, the incremental encoder DSP units are recommended to be zeroed at a known point every time the Axis Portal is initialized. This does not affect the operation of the system but ensures a stable reference point. A good approach is to drive the crane to the negative limits of each axis and reset the encoders once the physical limit switches have been pressed.

Operation:

The GOLDI Axis Portal V2 is operated by writing the corresponding values into the FPGA registers (See Register Table)

SPI Communication Protocol

The FPGA Control Unit is configured through an SPI interface. The SPI interface allows the independent reading and writing of values from and into the registers of the model.

SPI Signals
The GOLDI Axis Portal V2 has four implemented signals:

SPI0_SCLK	SPI serial clock
SPI0_MOSI	SPI serial data input
SPI0_MISO	SPI serial data output
SPI0_nCE0	SPI chip select input (active low)

The FPGA Control Unit is enabled for communication when a logic low is presented on the chip select input nCE0. The data transfered between a falling and rising edge of nCE0 is defined as a SPI transaction. The data bit transfer during a transaction is synchronous to the bus clock SCLK. The peripheral FPGA Control Unit registers data from MOSI on the rising edge of SCLK and drives data to MISO on the falling edge. The most significant bit is sent first. The SCLK signal is expected to remain high when the module is idle.

A valid SPI transaction consists of a 16-bit *configuration word* and one or more 8-bit *data words*, meaning a minimum of 24 SCLK clock cycles is required for a single valid transaction. If the nCE0 is driven high before the minimum 24 data bits have been registered or before an additional data word has been transfered the current data is discarded.

GOLDi SPI Protocol
The communication with the FPGA is controled by the *configuration word* at the begining of each SPI transaction. This input data configures the type of transaction in progress, the registers to be accessed during the data transfer and additional tag modifiers that change the behavior of the stored data.

The configuration word starts with the write enable "WE" tag which selects, if the operation is read-only (WE='0') or read-write (WE='1').
The next bit, the stream enable "SE" bit, selects the behavior of the SPI communication module when multiple *data words* are transfered in a single SPI transaction. When the multi-register mode is selected (SE='0'), the provided address acts as the inital register to be accessed. After the first operation has been performed, meaning the first data word has been transfered, the internal BUS address is drecreased by one and the next register is accessed. This data transfer mode simplifies the data transfer to a sub-module with multiple registers and data formats. In contrast, when the stream mode is selected (SE='1'), only the addressed register is accessed and the stored data is overwriten/read after every *data word* has been transfered. This data transfer mode is used primarly to transfer large data vectors to secondary communication sub-modules that configure ICs or other electronics. The sub-modules often have queue structures that prevent data loses.

The stream-enable bit is followed by a 4-bit tag word that is applied to all data words in the transaction and the 10-bit address word. Up to V4.00.00, the tag word remains unused by any sub-module in the GOLDi Axis Portal V2.

The GOLDi SPI Protocol is used by multiple GOLDi Model Units, therefore, the exact widths of the address, tag, and data words can be modified in the GOLDI_COMM_STANDARD package. A change to the "BUS_ADDRESS_WIDTH", "BUS_TAG_BITS", or "SYSTEM_DATA_WIDTH" automaticaly resizes the entire system.

Default configuration for the GOLDI Axis Portal V2 model

BUS_ADDRESS_WIDTH	10
BUS_TAG_BITS	4
SYSTEM_DATA_WIDTH	8

Configuration Word [15:0]				Data Word[7:0]
Bit 15	**Bit [14]**	**Bit [13:10]**	**Bit [9:0]**	**Bit [7:0]**
WE	SE	TAG	ADDRESS	DATA[MSBF]
0	0	-	READ_START[9:0]	[MOSI: dc] | [MISO: Register data]
0	1	-	READ_ONLY[9:0]	[MOSI: dc] | [MISO: Register data]
1	0	-	WRITE_START[9:0]	[MOSI: New data] | [MISO: Register data]
1	1	-	WRITE_ONLY[9:0]	[MOSI: New data] | [MISO: Register data]

GOLDI Axis Portal V2 hardware pinout

Control Unit Pinout					
Hardware Pinout		FPGA System			
Signal Name	Schematic Name	Pin Type	Pin Number	Pin Mode	Entity name
External FPGA Clock	ClockFPGA	in	128	LVCMOS33	ClockFPGA
Reset	FPGA_nReset	in	126	LVCMOS33	FPGA_nReset
SCLK	SPI0_SCLK	in	138	LVCMOS33	SPI0_SCLK
MOSI	SPI0_MOSI	in	133	LVCMOS33	SPI0_MOSI
MISO	SPI0_MISO	out	139	LVCMOS33	SPI0_MISO
nCE	SPI0_nCE0	in	127	LVCMOS33	SPI0_nCE0
Multi-purpose GPIO0	GPIO0	inout	125	LVCMOS33	IO_DATA[0]
Multi-purpose GPIO1	GPIO1	inout	122	LVCMOS33	IO_DATA[1]
X-axis sensor left	XAxis_LSLeft	inout	84	LVCMOS33	IO_DATA[2]
X-axis sensor right	XAxis_LSRight	inout	85	LVCMOS33	IO_DATA[3]
Y-axis sensor back	YAxis_LSBack	inout	92	LVCMOS33	IO_DATA[4]
Y-axis sensor front	YAxis_LSFront	inout	91	LVCMOS33	IO_DATA[5]
Z-axis sensor bottom	ZAxis_LSDown	inout	114	LVCMOS33	IO_DATA[6]
Z-axis sensor top	ZAxis_LSUp	inout	113	LVCMOS33	IO_DATA[7]
Z-axis sensor proximity	Sensor_Proximity	inout	22	LVCMOS33	IO_DATA[8]
Encoder X channel A	XAxis_EncA	inout	87	LVCMOS33	IO_DATA[9]
Encoder X channel B	XAxis_EncB	inout	89	LVCMOS33	IO_DATA[10]
Encoder Y channel A	YAxis_EncA	inout	104	LVCMOS33	IO_DATA[11]
Encoder Y channel B	YAxis_EncB	inout	105	LVCMOS33	IO_DATA[12]
X TMC2660 Clock	XAxis_CLK	inout	78	LVCMOS33	IO_DATA[13]
X TMC2660 Enable	XAxis_ENN	inout	77	LVCMOS33	IO_DATA[14]
X TMC2660 StallGuard2	XAxis_SG	inout	83	LVCMOS33	IO_DATA[15]
X TMC2660 Step	XAxis_STEP	inout	81	LVCMOS33	IO_DATA[16]
X TMC2660 Direction	XAxis_DIR	inout	82	LVCMOS33	IO_DATA[17]
X TMC2660 Serial Clock	XAxis_SCK	inout	75	LVCMOS33	IO_DATA[18]
X TMC2660 chip select	XAxis_nCS	inout	76	LVCMOS33	IO_DATA[19]
X TMC2660 mosi	XAxis_MOSI	inout	74	LVCMOS33	IO_DATA[20]
X TMC2660 miso	XAxis_MISO	inout	73	LVCMOS33	IO_DATA[21]
Y TMC2660 Clock	YAxis_CLK	inout	103	LVCMOS33	IO_DATA[22]
Y TMC2660 Enable	YAxis_ENN	inout	100	LVCMOS33	IO_DATA[23]
Y TMC2660 StallGuard2	YAxis_SG	inout	95	LVCMOS33	IO_DATA[24]
Y TMC2660 Step	YAxis_STEP	inout	93	LVCMOS33	IO_DATA[25]
Y TMC2660 Direction	YAxis_DIR	inout	94	LVCMOS33	IO_DATA[26]
Y TMC2660 Serial Clock	YAxis_SCK	inout	98	LVCMOS33	IO_DATA[27]
Y TMC2660 chip select	YAxis_nCS	inout	99	LVCMOS33	IO_DATA[28]
Y TMC2660 mosi	YAxis_MOSI	inout	97	LVCMOS33	IO_DATA[29]
Y TMC2660 miso	YAxis_MISO	inout	96	LVCMOS33	IO_DATA[30]
Z DC Motor enable	ZAxis_DCEnable	inout	107	LVCMOS33	IO_DATA[31]
Z DC Motor out 1	ZAxis_DCAPWM	inout	109	LVCMOS33	IO_DATA[32]
Z DC Motor out 2	ZAxis_DCBPWM	inout	110	LVCMOS33	IO_DATA[33]
E-Magnet enable	ZAxis_MagnetEnable	inout	111	LVCMOS33	IO_DATA[34]

E-Magnet out 1	ZAxis_MagnetAPWM	inout	106	LVCMOS33	IO_DATA[35]
E-Magnet out 2	ZAxis_MagnetBPWM	inout	112	LVCMOS33	IO_DATA[36]
Power LED Red	LEDPowerR	inout	141	LVCMOS33	IO_DATA[37]
Power LED Green	LEDPowerG	inout	140	LVCMOS33	IO_DATA[38]
Environment LED Red	LightRed	inout	33	LVCMOS33	IO_DATA[39]
Environment LED White	LightWhite	inout	34	LVCMOS33	IO_DATA[40]
Environment LED Green	LightGreen	inout	35	LVCMOS33	IO_DATA[41]

Register Map

Document Version	2
Hardware Version	V4.00.00
Date	30.09.2023

All registers have a base address located on the package GOLDI_MODULE_CONFIG. This can be changed to move the modules in case the configuration word width is changed.

Register Name	Address (Dec)	Address (Hex)	Default	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
System Configuration	1	0x01	0x00	ID=0010 (rst)						enc_y_rst	enc_x_rst
Sensor IO	2	0x02	0x00		Proximity	YTop	YBottom	YFront	YBack	XRight	XLeft
Error list 1	3	0x03	0x00	error_7	error_6	error_5	error_4	error_3	error_2	error_1	error_0
Error list 2	4	0x04	0x00				error_12	error_11	error_10	error_9	error_8
GPIO0 Driver	5	0x05	0x00							Out_enb	Data
GPIO1 Driver	6	0x06	0x00							Out_enb	Data
X Encoder low	7	0x07	0x00	X_VAL[7:0]							
X Encoder high	8	0x08	0x00	X_VAL[15:8]							
Y Encoder low	9	0x09	0x00	Y_VAL[7:0]							
Y Encoder high	10	0x0A	0x00	Y_VAL[15:8]							
X Motor Control	11	0x0B	0x00	Pow_off					Stall	Dir1	Dir0
X Motor Speed low	12	0x0C	0x00	FRQ_VAL[7:0]							
X Motor Speed high	13	0x0D	0x00	FRQ_VAL[15:8]							
X Motor SPI 0	14	0x0E	0x00	CONFIG_WORD[7:0]							
X Motor SPI 1	15	0x0F	0x00	CONFIG_WORD[15:8]							
X Motor SPI 2	16	0x10	0x07	CONFIG_WORD[23:16]							
Y Motor Control	17	0x11	0x00	Pow_off					Stall	Dir1	Dir0
Y Motor Speed low	18	0x12	0x00	FRQ_VAL[7:0]							
Y Motor Speed high	19	0x13	0x00	FRQ_VAL[15:8]							
Y Motor SPI 0	20	0x14	0x00	CONFIG_WORD[7:0]							
Y Motor SPI 1	21	0x15	0x00	CONFIG_WORD[15:8]							
Y Motor SPI 2	22	0x16	0x00	CONFIG_WORD[23:16]							
Z Motor Control	23	0x17	0x00							Out1	Out2
Z Motor Speed	24	0x18	0x00	PWM[7:0]							
Electromagnet Power	25	0x19	0x00								mag_pow
Power LED Red	26	0x1A	0x00	on/off	Blink_enb	Delay_on				Delay_off	
Power LED Green	27	0x1B	0x00	on/off	Blink_enb	Delay_on				Delay_off	
Environment LED Red	28	0x1C	0x00	on/off	Blink_enb	Delay_on				Delay_off	
Environment LED White	29	0x1D	0x00	on/off	Blink_enb	Delay_on				Delay_off	
Environment LED Green	30	0x1E	0x00	on/off	Blink_enb	Delay_on				Delay_off	

Error List

Error code	Error definition
error_0	Sensors XLeft and XRight triggered
error_1	Sensors YBack and YFront triggered
error_2	Sensors ZBottom and ZTop triggered
error_3	X motor actuated to the left while crane not in top position
error_4	X motor actuated to the right while crane not in top position
error_5	Y motor actuated to the back while crane not in top position
error_6	Y motor actuated to the front while crane not in top position
error_7	XLeft triggered and X motor actuated to the left
error_8	XRight triggered and X motor actuated to the right
error_9	YBack triggered and Y motor actuated to the back
error_10	YFront triggered and Y motor actuated to the front
error_11	ZBottom triggered and Z motor actuated to the bottom
error_12	ZTop triggered and Z motor actuated to the top

TMC2660 Stepper Motor Driver Interface

General description
The stepper motors are driven by the TMC2660. The TMC2660 is a driver for two-phase stepper motors with multiple industrial features. The driver includes high-resolution microstepping, sensorless mechanical load measurement, load measurement, load-adaptive power optimization, and low-resonance chopper operation. It si operated through either a standard SPI interface of a STEP/DIRECTION interface.

Configuration process:
The TMC2660 requires setting configuration parameters and mode bits through the SPI interface before the motor can be driven. The SPI interface also allows reading back status values and bits.

The Axis Portal V2 hardware provides the TMC2660 with an initial configuration when the module is started. Five 24-bit SPI transactions are performed automatically after initialization to write the data to the 5 registers of the TMC2660 and activate the driver. The default values for the initial configuration are located in the GOLDI_MODULE_CONFIG package stored in an "array_16_bit" structure. This data structure stores 16-bit words in instantiated PLU ROM units. Once the hardware is started a FIFO structure loads the data into the SPI transmitter until the module has been programmed. After this initial configuration, the module can be operated through the STEP/DIRECTION interface.

After the initial configuration, the TMC2660 data can be modified by the user mid-operation using the three SPI registers in the TMC2660_SMODULE. Given that the TMC2600 returns the same data regardless of the rewritten register, three registers are enough to efficiently communicate with the chip. The data is passed to the FIFO structure that queues the SPI transmitter. The SPI stream interface reacts to the write strobe of the lower register, meaning the FIFO structure is loaded with the register's data when the register "SPI 0" data is modified. The FIFO structure has been placed between the registers and SPI transmitter to allow the user to program up to 5 configuration words with the faster FPGA SPI interface without data losses.

STEP/DIRECTION interface (normal actuation of motor):
To drive the stepper motor the STEP/DIRECTION interface of the TMC2660 is used. This interface is operated by the lower three registers in the TMC2660_DRIVER (the "motor speed" and "motor control") registers.

The motor speed registers contain the step signal frequency expressed in Hertz and the motor control register has two direction-dependent enable signals. (If both are active the Dir1 takes precedent) Additionally, the stall bit reads the StallGuard2 information of the TMC2660 and flags a stall of the stepper motor. The power-off bit temporarily disables the TMC2660 and allows the stepper motor to rotate freely in case it has to be moved manually and should be activated before the FPGA reprogramming. The STEP/DIRECTION interface takes the speed value at the beginning of the operation. To change the speed of the motor the driver must first be stopped.