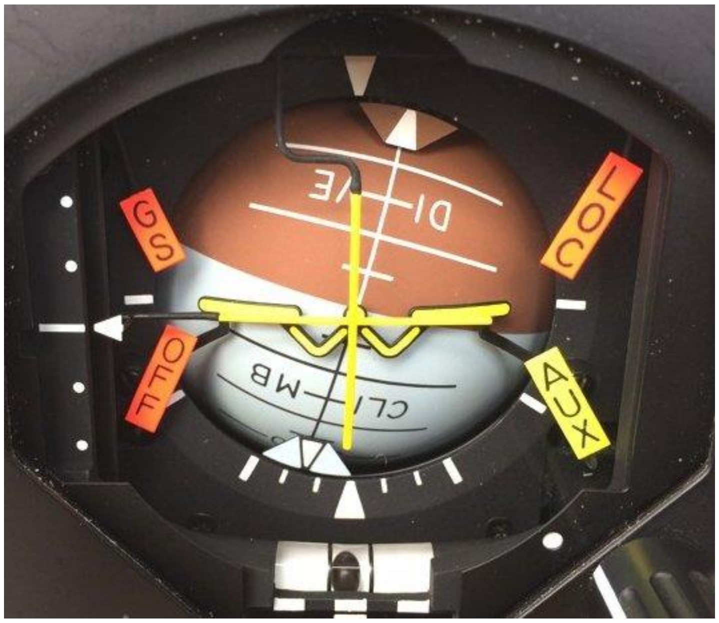
SYNCHRO DRIVE INTERFACE



Version: V1.2 March 2016

History

07-2014 : Draft version, v0.0 started
This version describes the hardware at a global perspective, and describes how the Synchro Drive Interface (SDI) will (probably) be implemented.

08-2014 : Major update, v0.1, still draft version
Based on an earlier design (HYDRAULIC PRESSURE A&B interface), a draft design of the SDI is made. This is a major update of this document with new information.

08-2014 : Version v0.2, still draft version
Based on the initial firmware code, the DIGITAL/PWM selectable output mask is no longer a message per output, but one message that specifies 7 "configuration bits".

09-2014 : Version v0.3, still draft version, the prototype of the PCB has arrived.
Added a few extra commands (see chapter 6).

10-2014 : Version v0.4. Minor update in chapter 3.4.
On last page updated remark #3 and added remark #4. Added an "Application example" chapter.

06-2015 : Version v0.5 after some distractions to other projects.
No longer a draft version! 'DEMO' mode command added, but also very useful for test purposes. Further, sub-address 0 is a special case. Phase-shift filters are not needed – hardware redesign. Added the IDENTIFY command (USB only). That command was available, but not documented (hidden feature, but might be useful).

07-2015 : Version v0.6. Description of URC, "update rate control", (chapter 6.3) added. Further, separate parts list are given for specific functional blocks. For example, if no local 400 Hz oscillator is needed, it is clear which parts on the PCB can be omitted.

08-2015 : Version v0.7. Corrections in several references.

11-2015 : Version v1.0 (!) Added commands to control watchdog functionality. The separate commands to set the stator polarity (directly or 'deferred') are combined to a single command that sets the polarity for all 3 stator signals. Further, the "POWER DOWN" feature is added (but proved to be not as useful as I expected). Command numbers 1 thru 13 have changed, command numbers 14 thru 31 remain the same.

12-2015 : Version v1.1 Errata chapter added ☹.

03-2016 : Version v1.2 USB debug enable/disable command added, and indicator minimum and maximum value limits. The "weird" 10-bit command is replaced by 4 commands, each defining a position (in 8-bit resolution) *in a quadrant.*
The SDI project (hardware/firmware/documentation) is released and distributed.

Contents

Errata

12-2015 : R18 and R20 must be 390kΩ for 400 Hz center frequency of the State Variable Filter.

1. Introduction

This document describes the Synchro Drive Interface, abbreviated to SDI. This SDI design is an attempt to construct a generic interface which can drive ***one*** so-called synchro. The synchro is used in many (older) airplane instruments for some form of indication. The indication can be a simple needle pointer as in the HYDRAULIC PRESSURE A/B indicators, but can also be a (mechanically) complex object like the sphere of an Attitude Direction Indicator (ADI).

A synchro can be compared with a transformer which has a 3-phase stator and a one phase rotor. When an AC signal (often 400 Hz) is connected to the rotor coil and AC signals are applied to the 3 stator coils with the same phase relation to the rotor signal and certain amplitudes, the rotor is driven to a defined equilibrium position, due to the magnetic force fields.

The SDI has the following features.

- Small PCB (3.1"x3.9"), easy to build, no SMD components
- On-board 400 Hz sine wave generator
- Synchronization input to an external 400 Hz source
- Output signals for one synchro (rotor signal, stator signals S1, S2, S3)
- Output signal for one "center-zero" or "left-zero" coil-driven needle pointer
- 7 spare digital outputs, separately configurable as simple ON/OFF or PWM output
- Many commands to suit every application, also a 'DEMO' (test) mode command
- USB or PHCC DOA interface, jumper-selectable.

The SDI output signals for the synchro (rotor, stator S1, stator S2, stator S3) are the outputs of a TL084 OpAmp. These signals are probably not capable to drive a synchro directly. They can only drive the synchro of an instrument that incorporates internal amplifiers, or drive (trim) potentiometers that set the sensitivity of external (audio) amplifiers. These amplifiers connect to the synchro coils. The external amplifiers are described in a separate project.

Note that this document cannot give a detailed description how to connect instrument "this or that". The actual connection depends entirely on the specifications of the instrument which can vary a lot. If the instrument does not use synchros, the SDI is not what you need. Further, this SDI may not be sufficient to drive all functionality of an instrument. For example, the ADI has two synchros (one for ROLL and one for PITCH indication of the sphere), thus two SDI boards are required.
You need to know and understand the connections and operation of the instrument to decide whether the SDI is suitable to drive the instrument. The SDI is not a project suitable for electronic "noobs". ***However, if the connections and additional hardware (if required) for a specific instrument has been figured out, the SDI is easy to use.***

The SDI will always be a part of a specific instrument interface solution. At this moment, the ARU-50/A F-16 ADI is the only tested solution. This solution will also be supported in the PHCC Device Manager, SimLinkup Mapper and SimLinkup software (interface to BMS).

Take some time to read through the entire manual. The "Assembly" and "Parts list" chapters are at the end of the manual, because you will need that only in the beginning, and probably never later on. Have a look at the "Detailed descriptions", because it may help you understand what the function of the jumpers and connections is.

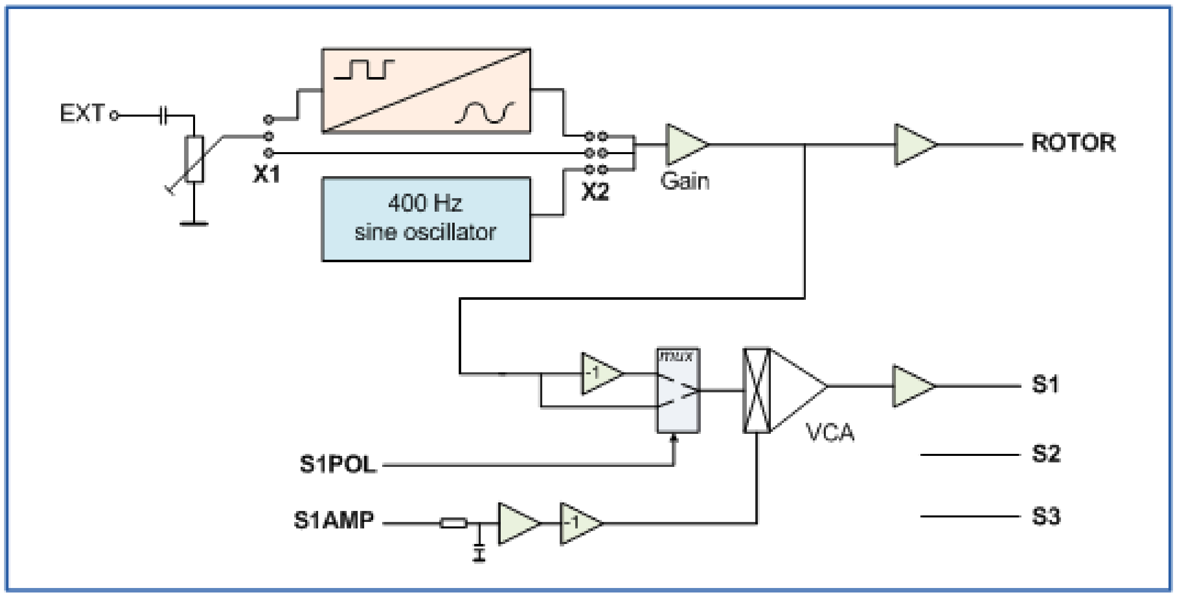
I spent quite some time searching the internet, studying documentation, developing and testing hardware and software for the SDI. Therefore, I put some "copyright" on my work. You are allowed to build and use the SDI for your own purpose, but you are not allowed to make commercial profit building and/or selling the SDI. This manual gives detailed descriptions of major parts of the hardware design, because I feel that this information is needed to successfully connect and control an instrument. However, it is not the intention to enable easy copying of my work and try to make some money. The SDI is developed for hobbyists who want to 'bring to life' a real instrument *without modifications* to the delicate mechanisms inside the instrument.
The source code of the PIC firmware is only available on request and will only be made available to persons who bought the SDI PCB. My email address is henk.gooijen@hotmail.com .

Disclaimer

All uses of the SDI are solely *your* responsibility. Any damage to the SDI hardware, other hardware or connected instrument(s) is ***not*** my responsibility.

2. Block diagram

The SDI has a local 400 Hz sine wave oscillator to generate the signal for the rotor and the 3 stator signals. The 3 generated stator signals and their polarity-inverted signals (180° phase shift) are connected to CMOS switches (small box called 'MUX'). A control signal SxPOL (*Stator signal "x" Polarity*) selects which signal will drive the next stage.



The output of the CMOS switch is connected to a VCA (Voltage Controlled Amplifier). A Pulse Width Modulated (PWM) control signal SxAMP (*Stator signal "x" Amplification*) is converted to an average DC voltage and these voltages set the amplification of the signals. These stator signals are connected via buffers to the output pins.

Sometimes an indication instrument is connected to a 400 Hz power supply. In the case that the 400 Hz power supply is used for the rotor signal of the synchro inside the instrument, the stator signals must have a correct phase relation with the rotor signal. For this purpose the SDI has an EXT input connection. The choice of an internal or external 400 Hz rotor signal source is jumper-selectable on the SDI board. If the external source is selected the SDI can handle a sine wave and a block wave source signal (via an on-board filter).

Not shown in the figure is the PIC processor. The PIC processor generates the polarity and amplification control signals for the 3 stator signals. The communication with the SDI can be either USB (to a PC) or PHCC DOA bus (to the PHCC Motherboard). This selection is set by a jumper on the SDI board.

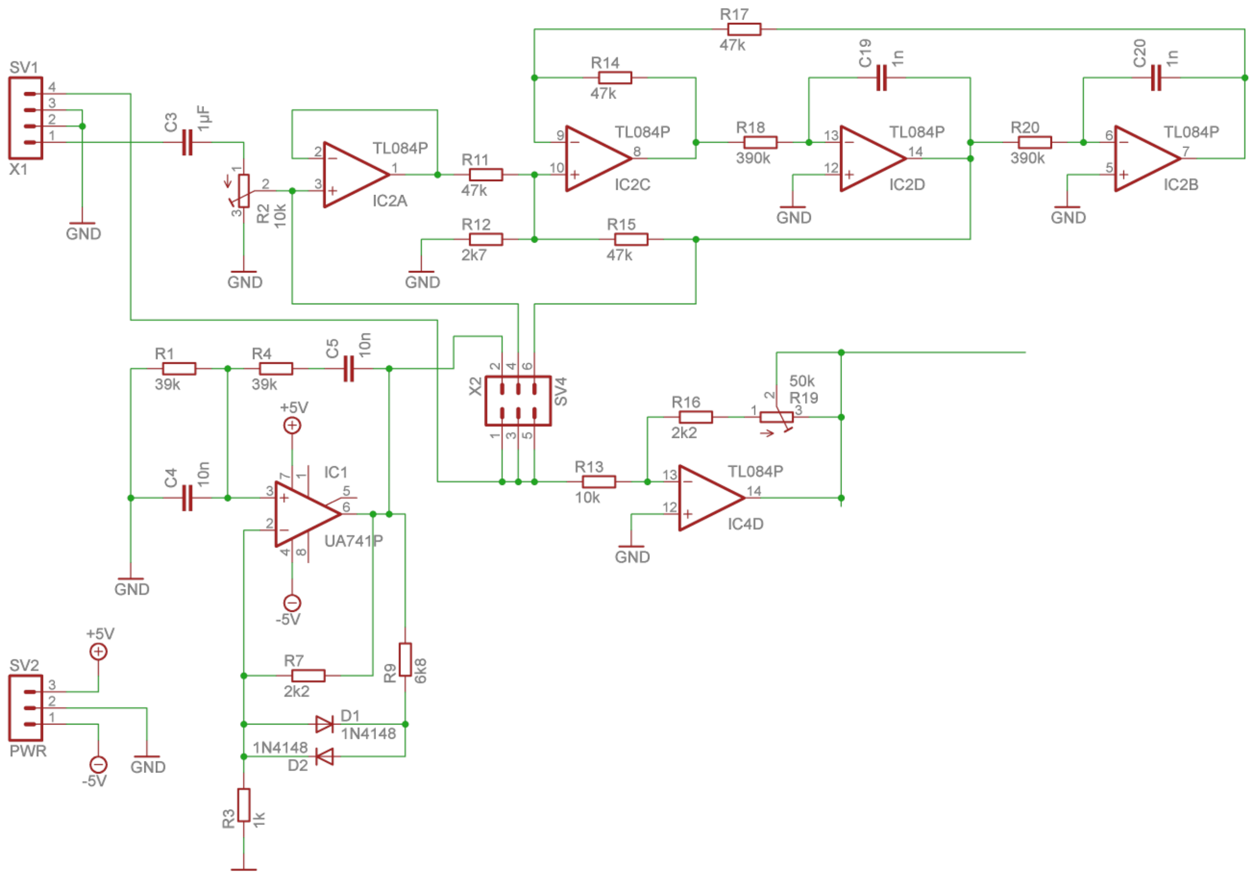
As the PIC processor has several pins that are not used for the synchro interface, some additional functionality is added. One pin is used to drive an on-board LED for diagnostic purposes. Other pins are defined as digital outputs. Via the communication with the SDI, these digital outputs each can separately be configured as a simple ON/OFF output (output is either logic "0" or logic "1", TTL levels), or as a 256 step duty-cycle PWM signal. One such output is defined as PWM in the firmware, and the output is on-board level-shifted and buffered to support a "center-zero" or "left-zero" coil driven indicator.

3. Detailed descriptions

This chapter describes the function blocks of the SDI in detail. The information is only supplied as background information for who is interested in the design details. The information in this chapter is not needed to connect and/or use the SDI, but still might be useful to know.

3.1 Internal / external 400 Hz (rotor) signal

A synchro requires an AC (often 400 Hz) rotor signal. This is always needed, regardless whether the synchro is used as a drive or as a resolver (position encoder). Therefore, the SDI includes a 400 Hz sine wave oscillator built using one common OpAmp type µA741.



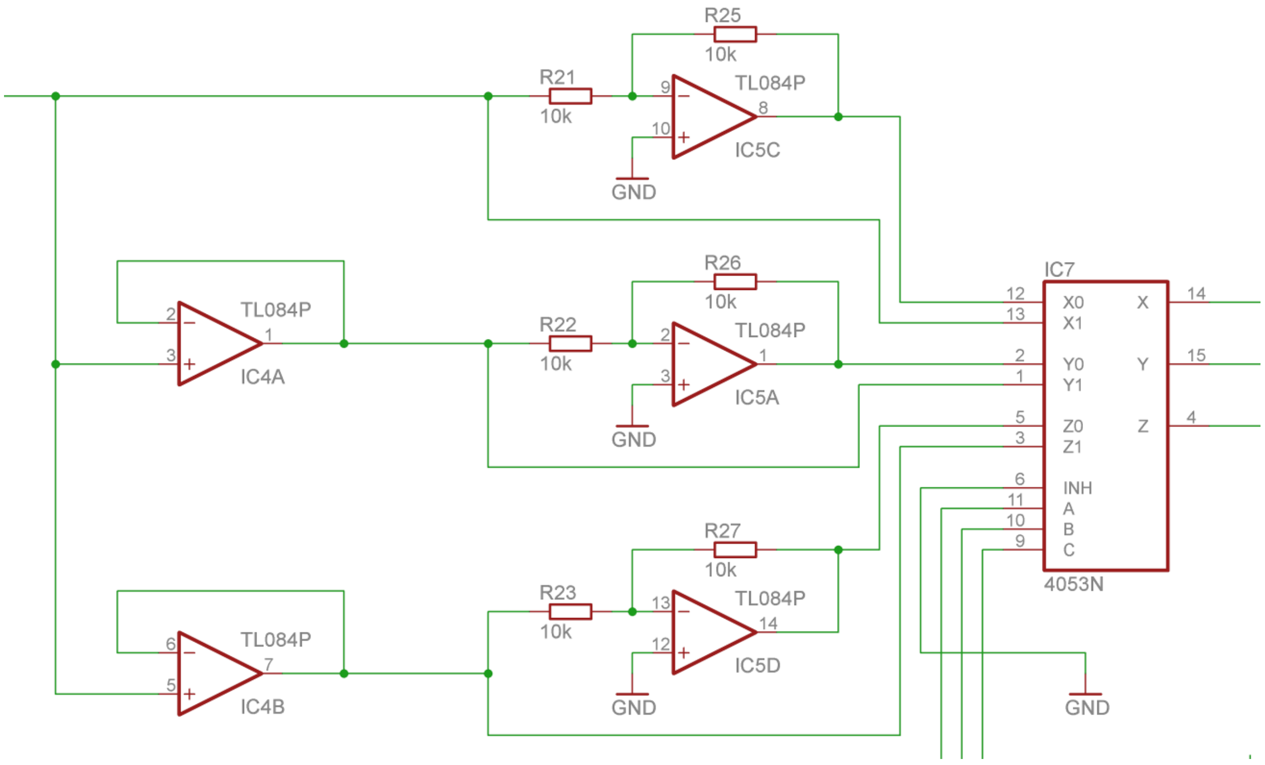
Large indicators (such as the ADI) have connections for a 400 Hz AC power supply. This power supply is internally used to generate the required rotor signal for the synchro(s), and possibly feed internal amplifiers. As the correct stator signals must have the same phase relation with the rotor signal, the SDI has an input to which the (400 Hz) AC power supply voltage can be connected. This external signal is connected to connector X1. Note that if the power supply voltage is 115V, an external voltage divider must be used! If you use a (home-made) DC to 400 Hz AC inverter, you might use the low-voltage block wave from the oscillator at the "DC side" of the inverter. In that case the SDI provides a 2nd order band pass filter to convert the block wave to a sine wave. However, the voltage divider using the 115VAC is better, as it is already a 'clean' sine wave. *Note that the 2nd order band pass filter introduces a phase shift which must be corrected by setting the base angle to compensate for the phase shift.*

The selection whether the external source signal is fed via the 2nd order band pass filter is done with jumper X2. The adjustment potentiometer R2 “SENS” adjusts the input level to max 4 Vtt amplitude. You need an oscilloscope to check that the output is not clipped, or use an AC voltmeter and set the output level “conservative” (max 3 Vtt), because using an AC meter does not tell you that the amplifier output is clipping.

Jumper X2 selects the rotor signal source for the generation of the stator signals. Three possible sources are available, on-board 400 Hz oscillator (jumper on pins 1–2), external connection direct (no jumper, X1 pin 4 is an input), and external connection with amplitude adjustment (via R2) either directly (jumper on pins 3–4) or via band pass filter (jumper on pins 5–6). The adjustment potentiometer R19 “GAIN” adjusts the input level for the stator signal generation circuits to max 4 Vtt amplitude. Again, check that the amplifier output is not clipping. If X1 pin 4 is not used as input, the selected source signal is available on this pin.

3.2 Stator signal generation

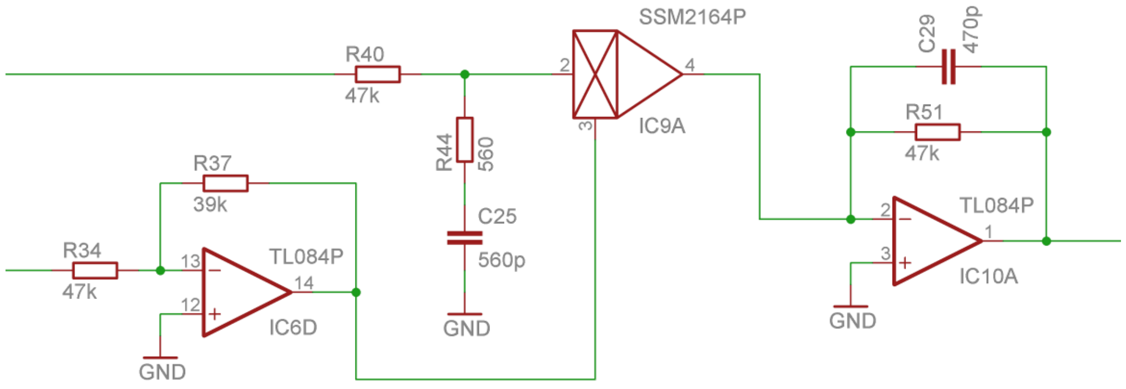
The stator signals are generated using the source selected sine wave. For full control over 360° rotation, it must be possible that the stator signal has 180° phase shift related to the selected source signal. 180° phase shift is simply a signal polarity inversion, which is realized with the OpAmp IC5.



The stator signals and inverted stator signals are connected to a 3-channel CMOS switch CD4053. With the selection lines A, B, C of the CMOS switch the outputs of the switch are either the stator signals or the inverted stator signals.

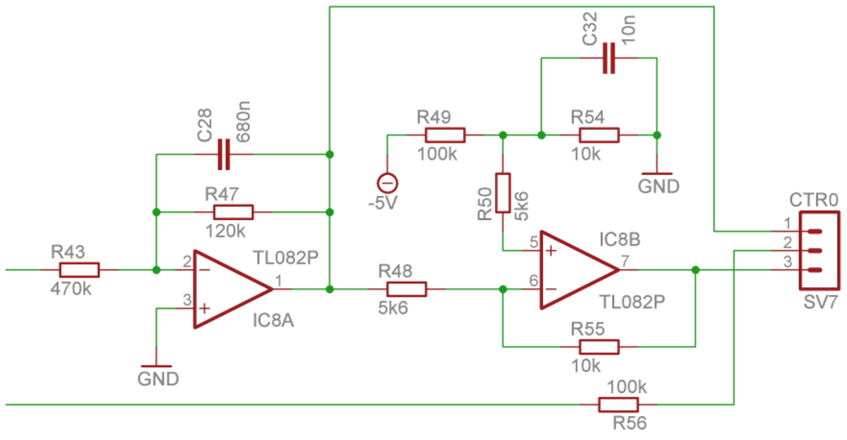
3.3 Stator signal amplitude control

The output signal for each stator coil must be driven from 0V to the maximum amplitude. This amplitude variation is realized using a so-called VCA, Voltage Controlled Amplifier. The PIC generates a PWM signal which is fed to an integrator. At the output is a DC voltage level proportional with the duty-cycle of the PWM signal. This signal controls the attenuation of the VCA which is fed with the signal from the CMOS switch. The output of the VCA is a current which is converted to a voltage level with an additional OpAmp.



3.4 PWM output

The PIC on the SDI board has several not used pins. These pins are available via two headers, SV1 and SV2. As there was some unused space on the circuit board, one output is used to generate a voltage level to drive an analog meter, either “center-zero” or “left-zero”. The PIC generates a TTL-level PWM signal which is converted to a DC voltage level. The output of IC8A is approximately 0V at a duty-cycle of 0%, and is -1.25V at a duty-cycle of 100%. IC8B takes care of level shifting, so that duty-cycle 0% results in -1.25V and duty-cycle 100% results in +1V. A duty-cycle of 50% should result in 0V output, but that depends on R49. With the given standard values, the output will be a few tenth Volt negative. On-board is a current limiting series resistor (R56). Its value depends on the sensitivity of the connected analog “center-zero” indicator. In combination with the ADI Support board (for the Rate of Turn indicator) replace R56 with a wire or 0Ω resistor. If the indicator is “left-zero” put the jumper on pins 1–2, if the indicator is “center-zero” put the jumper on pins 2–3.



4. Connections and jumpers

The SDI has 7 connectors and 4 jumpers. Pin #1 is indicated on the PCB.

4.1 POWER connector

The POWER connector connects a +5 Volt and -5 Volt power supply to the SDI. Note that the SDI needs a symmetrical power supply. Make sure that you connect the power supply correctly.
→ ***The SDI board has no provisions against wrong polarity connection.***

pin number	Signal	Usage
1	–5V	–5V power supply for the SDI
2	GND	Ground (0V)
3	+5V	+5V power supply for the SDI

4.2 X1 connector

The X1 connector has two different purposes, depending on whether you use the internal 400 Hz sine wave oscillator or need to synchronize the stator signals generated by the SDI to an external (AC power) source used by the instrument. See chapter 3.1 for details.

pin number	Signal	Usage
1	EXT_IN	400 Hz input for synchronization to external source (max 4Vtt)
2	GND	Ground (0V)
3	GND	Ground (0V)
4	SRC IN/OUT	Output used (rotor) source signal if a jumper in X2 Input (direct) for synchronization to external source (no jumper in X2)

4.3 X2 jumper

The X2 jumper selects what the source is for the generation is the sine wave signals. See chapter 3.1 for details.

pin number	Signal	Usage
1 – 2	SRC_INT	Internal 400 Hz sine wave oscillator
3 – 4	SRC_EXT	External 400 Hz sine synchronization
5 – 6	SRC_EXT_CDX	External 400 Hz sine synchronization (conditioned)
no jumper	SRC_DIRECT	External 400 Hz directly from X1 pin 1

4.4 SYNCHRO connector

The SYNCHRO connector has the outputs pins that drive the synchro, probably using external (trim) potentiometers and amplifiers. The signals are outputs of TL084 OpAmps.

pin number	Signal	Usage
1	GND	Ground (0V)
2	R	Synchro rotor signal
3	S3	Synchro stator S3 signal
4	S2	Synchro stator S2 signal
5	S1	Synchro stator S1 signal

4.5 RST jumper

The RST jumper selects what the function is of pin #1 of the DOA connector.

pin number	Signal	Usage
1 - 2	DOA1_GND	Pin 1 of DOA connector tied to GND (original DOA specification)
2 - 3	DOA1_RST	Pin 1 of DOA connector tied to RST* of PIC Pin 1 of the DOA connector can be used as external reset signal for the PIC. Note that the original ribbon cable from the PHCC Motherboard for DOA **cannot** be used, because it will keep the PIC in reset continuously.

4.6 USB/DOA jumper

The USB/DOA jumper selects which the communication interface to the SDI is used.

pin number	Signal	Usage
1 - 2	USE_USB	The SDI uses the USB connector for communication
no jumper	USE_DOA	The SDI uses the DOA connector for communication

4.7 USB connector

Standard Type B USB connector to PC.

4.8 DOA connector

Standard PHCC DOA bus connector.
Note the use of jumper RST.

4.9 SV1 connector

The SV1 connector has the connection pins to the first 5 digital outputs. These outputs are available for any use. Each of these 5 outputs can be configured via the active communication connection. The output can be a simple digital output (logic “0” / logic “1”), or a PWM output (0V or +5V level) with (programmable) adjustable duty-cycle (0 – 100%).

pin number	signal	Usage
1	DIG_PWM_1	Digital output DIG1 or PWM1
2	DIG_PWM_2	Digital output DIG2 or PWM2
3	DIG_PWM_3	Digital output DIG3 or PWM3
4	DIG_PWM_4	Digital output DIG4 or PWM4
5	DIG_PWM_5	Digital output DIG5 or PWM5
6	GND	Ground (0V)

4.10 SV2 connector

The SV2 connector has the connection pins to the last 2 digital outputs as describe for the SV1 connector, and the PWM output with a “conditioned” level suitable to drive an analog needle pointer indicator directly.

pin number	signal	Usage
1	DIG_PWM_6	Digital output DIG6 or PWM6 (see SV1 note)
2	DIG_PWM_7	Digital output DIG7 or PWM7 (see SV1 note)
3	GND	Ground (0V)
4	PWM_OUT	PWM output for analog indicator (needle pointer) –1V ... +1V or 0V ... +2V (see chapter 3.4)

4.11 CTR0 jumper

The CTR0 jumper selects the signal output range of the on-board PWM output.

pin number	Signal	Usage
1 - 2	ZERO	PWM_OUT output range is 0 ... –2V. Note the polarity with respect to GND! If the indicator is “left-zero” and one connection of the indicator is “hard-wired” connected to GND, you **cannot** use this output to control the indicator.
2 - 3	CENTER	PWM_OUT output range is -1V ... +1V.

5. Setting up and first use

Although the SDI is in principle a simple interface with relative few connections, there are a few attention points. Therefore, I propose a step-by-step approach to use the SDI.

5.1 Initial jumper configuration

The SDI has 3 jumpers, USB/DOA, RST, and X2.

- USB/DOA

Jumper USB/DOA defines which communication port you use. If the jumper is installed the SDI uses the USB connection. If the jumper is not installed the SDI uses the DOA connection.

- RST

Jumper RST (re)defines the use of pin 1 of the DOA connector. A jumper installed on pin 1 – 2 of the RST header makes the DOA connector 100% compatible with the standard DOA bus ribbon cable connection to the PHCC Motherboard. If the jumper is installed on pin 2 – 3 of the RST header, pin 1 of the DOA connector is redefined as a RESET input for the PIC processor. Note that this is *not* compatible with the standard DOA ribbon cable! If you want an external reset possibility for the PIC, you have to cut wire #1 of the DOA connection cable. If you use the USB connection (jumper USB/DOA installed), you can still use the reset option via the DOA connector if the jumper is installed on pin 2 – 3 of the RST header.

- X2

Jumper X2 defines the source of the rotor signal to be used for the synchro. The signal source can be the on-board 400 Hz sine wave oscillator, or an external signal source. If you want to drive a simple (needle pointer) indicator you can use the internal signal source, because these simple instruments do not have/need some form of power supply connection. The on-board signal source is selected by a jumper on pin 1 – 2 of X2.
If you want to drive some indication that has its own AC power source, it probably also generates internally the synchro rotor signal. In that case you must use the AC power source as a reference (synchronization) for the stator signal generation on the SDI. The SDI offers 3 possibilities how to use the external signal source. No jumper at all connects the external signal source *directly* to the buffer amplifier. A jumper on pin 3 – 4 connects the external signal source via a DC blocking capacitor and the SENS trim potentiometer to the buffer amplifier. You can use this jumper if the external signal source is a sine wave. A jumper on pin 5 – 6 connects the external signal source via a DC blocking capacitor and the SENS trim potentiometer to a band pass filter. You can use this jumper if the external signal source is a block wave, or if you want to ‘clean up’ the applied AC signal.
Make sure that the voltage level of the external signal does not have excessive high amplitude. 115V is definitely too much, keep the level below +/-4V. With the SENS trim potentiometer you can adjust the voltage level to prevent clipping by the OpAmps.

5.2 Initial (minimal) connections

To operate the SDI you need at least two connections: power and communication. For an initial riskless check, install a jumper on X2 pin 1 – 2 (local 400 Hz oscillator selected), a jumper on RST pin 1 – 2 (standard DOA cable), and put (or remove) the jumper on USB/DOA appropriately to select the communication port.

Connect a power supply that outputs a clean –5V and +5V to the POWER connector. The SDI has *no provisions* against wrong (polarity) connection of the power supply inputs.

➔ **Make sure that you observe correct polarity to prevent damage to the electronics.**

Connect the selected communication port.
If you use a USB connection, you need a Type B to Type A USB cable. Type B is used on the SDI, Type A is used on the USB connection of the PC (or the USB hub). Note that the SDI does not use power from the USB connection.
If you use the DOA connection, connect the 10-wire ribbon cable to the DOA bus connector on the PHCC Motherboard. Note that the +5V connection from the PHCC Motherboard is not used on the SDI.

After you have set the jumpers and connected the power supply and communication connection you are ready to switch on the +/–5V power supply. Check the correct power supply connections one more time … The **DIAG** LED flashes at 1 Hz, if the DOA connection is selected. The **DIAG** LED flashes at 2 Hz (twice as fast), if the USB connection is selected.

Switch off the +/–5V power supply if the **DIAG** LED does not flash on/off. Check connections, correct IC orientation (pin #1), soldering joints and possible tiny short circuits on the PCB (use a lamp and magnifier glass). You did program the PIC processor …? (If you got a 'kit', the PIC *is* programmed *and* verified).

5.3 Initial communication with the SDI

If the **DIAG** LED flashes you can communicate with the SDI.

- DOA
  Note that the DOA protocol is "send only". The protocol only sends data from the PHCC Motherboard to a PHCC slave (the SDI in this case). It is not possible that the Motherboard receives data from a slave. You can use *lightning's* PHCC TestTool on the PC to send commands to the SDI.

- USB
  Based on the supported DOA connection, the USB connection behaves the same. That is, the SDI does not send data to the PC via the USB cable (there are two exceptions, see the IDENTIFY and USB DEBUG commands, chapter 6.6 and 6.7). You can use a terminal communication program on the PC to send commands to the SDI. PuTTY is an example of a terminal communication program.

A simple check to see whether the communication is working is changing the functionality of the **DIAG** LED. You can change the flashing "heart beat" indication to "always OFF", "always ON", or change state per received (and accepted) command. See chapter 6.

6. Communication protocol

If the SDI is used as a PHCC slave (DOA communication) you must send the DOA address byte, the sub-address byte and a data byte. The DOA address byte identifies that the data packet sent is for the SDI. The value of the DOA address is hard-coded in the PIC firmware, and can be any value as long as the value is unique on the entire DOA bus. The sub-address and data byte define for which functionality the data is intended.

If you use the USB connection it is clear what the destination is, and the address byte is not needed. The USB data packet thus consists of the sub-address and the data byte (in that order).

This chapter describes the implemented sub-addresses with their possible data byte values. Remember that the SDI is a “listen-only” device, thus if you want to know what data was sent (for example DIG_PWM output configuration bits) the sending program must keep a local copy.

SYNCHRO CONTROL

sub-address	data byte	function / description
0	0x00 … 0xFF	SSYNQ1 – move indicator in “quadrant 1”
1	0x00 … 0xFF	SSYNQ2 – move indicator in “quadrant 2” (= quadrant 1 + 256)
2	0x00 … 0xFF	SSYNQ3 – move indicator in “quadrant 3” (= quadrant 1 + 512)
3	0x00 … 0xFF	SSYNQ4 – move indicator in “quadrant 4” (= quadrant 1 + 768)
12	0x00 … 0xFF	SYN8BIT – move indicator 0x00 :: 0° , 0xFF :: 359° (coarse resolution)
	Sub-addresses 34 ~ 37 have immediate effect on the control of a stator	
34	0x00 … 0xFF	S1PWM – set amplitude of the stator signal S1
35	0x00 … 0xFF	S2PWM – set amplitude of the stator signal S2
36	0x00 … 0xFF	S3PWM – set amplitude of the stator signal S3
37	0x00 … 0x07	SXPOL – set polarity of the stator signals S1, S2, S3
	Sub-addresses 6 ~ 8 are “deferred”. They prepare a new setting for direct control of all 3 stators. The three deferred values are synchronously loaded using sub-address 9. *Make sure you update all 3 amplitudes* ***and then*** *set 3 polarity values with sending command “9”.*	
6	0x00 … 0xFF	S1PWMD – set amplitude of the stator signal S1 deferred
7	0x00 … 0xFF	S2PWMD – set amplitude of the stator signal S2 deferred
8	0x00 … 0xFF	S3PWMD – set amplitude of the stator signal S3 deferred
9	0x00 … 0x07	SXPOLD – set polarity of the stator signals S1, S2, S3 + LOAD deferred
4	0x00 … 0xFF	LIMIT_MIN – Indicator movement limit minimum (0 : no limit minimum). Default for ROLL : 0 (disabled). Default for PITCH : 35.
5	0x00 … 0xFF	LIMIT_MAX – Indicator movement limit maximum (0xFF : no limit maximum). Default for ROLL : 255 (disabled). Default for PITCH : 175.

WATCHDOG functionality

sub-address	data byte	function / description
10	(*)	Disable watchdog functionality
11	0x00 … 0xFF	**Watchdog control --** See text for description.

POWER DOWN feature

sub-address	data byte	function / description
13	0x00 … 0xFF	**PowerDown feature --** See text for description.

USER-DEFINED DIGITAL OUTPUTS

sub-address	data byte	function / description
14	0x00 … 0xFF	DIG_PWM output selection mask bits When a bit is set to 0, the associated output is configured as a standard digital output (possible value “ON” or “OFF”). When a bit is set to 1, the associated output is configured as a PWM output (possible duty-cycle 0 ~ 100%). bit 0 – not used, ignored bit 1 – DIG_PWM_1 bit 2 – DIG_PWM_2 bit 3 – DIG_PWM_3 bit 4 – DIG_PWM_4 bit 5 – DIG_PWM_5 bit 6 – DIG_PWM_6 bit 7 – DIG_PWM_7 The 7 digital outputs are configured as a standard digital output after reset, and the output value is “OFF” (0). ► Note that outputs may have dedicated use in an application.
15 … 21	0 or 1 0x00 … 0xFF	Standard digital output. 0 :: “OFF” , 1 :: “ON”. PWM output. 0x00 :: duty-cycle 0%, 0xFF :: duty cycle 100%. Sub-address 15 → DIG_PWM_1, Sub-address 16 → DIG_PWM_2, Sub-address 17 → DIG_PWM_3, Sub-address 18 → DIG_PWM_4, Sub-address 19 → DIG_PWM_5, Sub-address 20 → DIG_PWM_6, Sub-address 21 → DIG_PWM_7.

MISCELLEANOUS

sub-address	data byte	function / description
22	0x00 … 0xFF	On-board OpAmp buffered PWM output 0x00 :: duty cycle 0% , 0xFF :: duty-cycle 100%.
23	0x00 … 0xFF	**URC - Update Rate Control --** See text for description.
24	0, 1, 2, 3	**DIAG** LED operation mode 0 – LED always OFF 1 – LED always ON 2 – LED flashes at heart beat rate 3 – LED toggles ON/OFF state per accepted command
25	0x00 … 0xFF	Stator S1 base angle LSB value *(must be set before S1 base MSB)*
26	0x00 … 0x03	Stator S1 base angle MSB value *(must be set after S1 base LSB)*
27	0x00 … 0xFF	Stator S2 base angle LSB value *(must be set before S2 base MSB)*
28	0x00 … 0x03	Stator S2 base angle MSB value *(must be set after S2 base LSB)*
29	0x00 … 0xFF	Stator S3 base angle LSB value *(must be set before S3 base MSB)*
30	0x00 … 0x03	Stator S3 base angle MSB value *(must be set after S3 base LSB)*
31	0x00 … 0xFF	**DEMO MODE --** See text for description.
32	0x00 … 0xFF	Start position used for DEMO mode
33	0x00 … 0xFF	End position used for DEMO mode
38	don't care	**IDENTIFY** USB only: send identification "SDI vA.B $xy"
39	'N' or 'Y'	**USB** debug command (USB only ☺)

6.1 How to control the synchro

The three stator signals must have a correct voltage amplitude to set the mechanical position of a synchro axis. The correct voltage amplitude for each stator coil can be set *immediately* with the sub-addresses SxPWM and SxPOL. SxPOL sets the polarity of the stator signal which can be positive or negative. SxPWM sets the amplitude of the stator signal. The algorithm is simple. For a setpoint angle alpha of the synchro axis, the algorithm (on the PC) calculates the sine values of alpha, (alpha +120°), and (alpha+240°). These values are made absolute and then scaled (value between 0 and 255). The sign of the values is sent to SxPOL, the absolute value is sent to SxPWM. To prevent "jumps" of the axis, you must change the polarity of the stator signal only when SxPWM is zero.

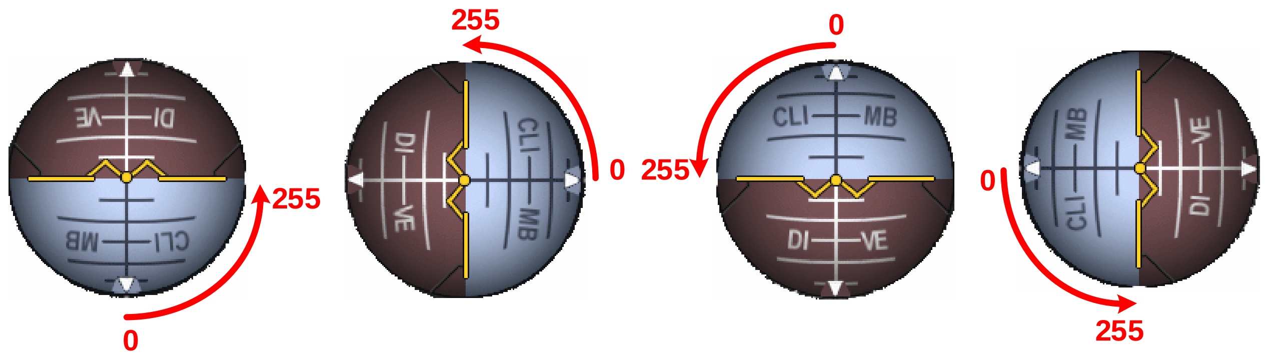
The SxPOL/SxPWM commands set outputs immediately after the command is received. This implies that there might be small delays between the set action of the 3 stator values. The delay can be caused by the PC (*Windows is not a real-time operating system!*) To prevent delays between the actual set actions of the 3 stator coil values, so-called "deferred" commands are available. When you send data using the "deferred" commands, the data is stored but the stator outputs are not updated. The update of all three stator amplitudes is executed synchronously when the command SxPOLD (Stator Polarity Load deferred values) is received. The last received amplitude values are used, if several values are sent to a specific stator. If no value is received, the last sent values are reloaded.

The SSYNQx (where x can be 1, 2, ,3 or 4) commands move the indicator to the “position” set by the 8-bit data byte. Value 0 represents (angle) position “0°” in the quadrant “x”, value 255 represents (angle) position “90°” in the quadrant “x”. An increasing value moves the sphere counter-clockwise. The SYN8BIT command is like the SSYNQx commands, but uses 8 bit (range 0 – 255) for the full 360°. It is up to the application to define what 0° and 359° represent. It is obvious that the interpretation will differ for a simple “needle-based” indicator such as HYDRAULIC PRESSURE, or an axis of the ADI. Both will require a specific correct offset in the application.

→ For the SSYNQx and SYN8BIT commands the base angle offset for the 3 stators must be set.

6.1.1 SSYNQx command data

Using the single byte data support by the DOA protocol would limit the resolution of a synchro to 8 bits. If the movement represents 360° rotation the resolution achieved using 8 bits would be 1.4°, which is rather coarse. The solution is *four* commands where each command specifies with 8 bits a position in a *quadrant*. Effectively, the position definition is now 10 bits (8 bits in 4 quadrants), thus the resolution is now 0.35°, which is adequate. Higher resolution is probably not achievable, due to limitation in the hardware (1 bit is a change in signal amplitude of ~10mV).



QUADRANT 1 QUADRANT 2 QUADRANT 3 QUADRANT 4

6.2 How to control the DIG_PWM outputs

The DIG_PWM outputs are outputs of the PIC processor. Keep the current from these outputs below 20 mA. This means that you can actually connect an LED directly (using a current limiting series resistor) to these outputs. First you define how a DIG_PWM output is to be used, as standard “digital” output (ON/OFF) or as a PWM “analog” signal output. This definition is set via sub-address 14. The default setting is standard “digital” output for all DIG_PWM outputs. The configuration of the DIG_PWM outputs can best be done at startup of the software.

After a DIG_PWM output is configured, explicit (or implicit at power up) as standard “digital” output, or explicit as PWM “analog” output, you can send data to the corresponding sub-address. For a standard “digital” output the accepted data value is either 0 or 1. Data 0x00 sets the output to logic “0”, data 0x01 sets the output to logic “1”. Any other data value (0x02 … 0xFF) is ignored. For a PWM “analog” output any data value is accepted. Data 0x00 sets the duty-cycle at

0% (the output is always OFF, logic “0”). Data 0xFF sets the duty-cycle at 100% (the output is always ON, logic “1”). A value between these limits sets a corresponding duty-cycle value. For example, 0x7F roughly matches a 50% duty cycle.

⊙ You can use the DIG_PWM output to control an LED, either ON/OFF or (PWM) dimmable. DIG_PWM outputs can be used with external electronics to drive other analog indicators.

6.3 POWER DOWN feature

The sphere of the ADI is moved to and held at a position by the resulting equilibrium of the magnetic forces between the rotor and 3 stator magnetic fields. It is possible that this position is continuously fluctuating around the equilibrium. This will create a “humming” sound. In my observations the PITCH movement does not make a humming sound, but the ROLL movement does make a humming sound at several specific positions. I guess that mass of the mechanism is a factor (PITCH only moves the sphere, ROLL moves everything). With the POWER DOWN feature you can switch off the stator voltages after a defined delay time. The delay time starts every time after a new position setpoint is sent to the ADI. When the delay time expires the stator supply voltages are switched off. You must choose the delay time of course larger than the time needed to make the complete position movement! This feature is after power-up of the SDI default *not* enabled. The data format for the POWER DOWN feature is as follows.

7	6	5	4	3	2	1	0
PWRDWN	**FULL**	6 data bits representing the power down delay time					

Bit 7 (“PWRDWN”) controls whether the POWER DOWN feature is enabled or disabled. When bit 7 is ‘1’ the POWER DOWN feature is enabled. When bit 7 is ‘0’ the POWER DOWN feature is disabled. Bit 6 (“FULL”) controls the “power down” level. When bit 6 is ‘1’ the “power down” level is maximum (0). As the drive of the stators is 0, the humming will stop. When bit 6 is ‘0’ the “power down” level is halve the actual set level value. The stators are still driven, but at halve the setpoint power in an attempt to decrease the humming sound. I found out that the ADI sphere has “preferred positions” when the drive power is decreased, so the “POWER DOWN” feature may not be useful. The 6 bits delay time value defines the delay time (in ms) between a new setpoint position is sent and power down “kicking in” (if the POWER DOWN feature is enabled). The time-out is the value of the 6 data bits multiplied by 32. If all 6 bits are zero (‘0000000’), the firmware default time-out value is set, which is 16, resulting in a delay of 512 ms.

6.4 DEMO mode

With the DEMO mode command you can drive the connected indicator as if a series of position commands is continuously sent to it. The start position of the movement and the end position of the movement are set with commands 32 and 33. They are defined in 8-bit precision, thus a value of 43 decimal represents and angle of 43 * 360/256° ≈ 60°, and a value of 255 decimal is approximately 360°.

→ *For the DEMO command the base angle offset for the 3 stators must have been correctly set.*

The DEMO command (number 31) has the following format.

7	6	5	4	3	2	1	0
MOVEMENT SPEED		MOVEMENT STEP SIZE				MODUS	START

The movement speed is the time delay between two position updates. For a smooth movement the update speed can be set to 100 ms, and for testing ("seeing what happens") the slowest speed is 2 seconds per update. The movement speed is defined with bit 7 and 6 of the DEMO command. "`00`" is 100 ms, "`01`" is 500 ms, "`10`" is 1 second, and "`11`" is 2 seconds.

The movement step size defines the "increment" in position per update, and is defined with bits 5 thru 2. When you "see" these 4 bits as a number, the value "`0000`" represents an increment of 1. Any other value ("`0001`" thru "`1111`") is multiplied by 2 and used as the increment value. Thus, "`1111`" will be an increment value of 30 decimal. Note that the value is a "byte value", and that 30 decimal on a 360° scale represents ≈ 42° which is probably a rather coarse movement.

The MODUS bit (1) can be either clear or set. When clear ("`0`") the movement is sweeping "up" from the defined start position (command 32) up to the defined end position (command 33) and then sweeping "down" from the end position back to the start position. The position value increases and decreases with the movement step size at the movement speed.
When the MODUS bit is set ("`1`") the movement sweeps "up" from the defined start position up to the defined end position and then *jumps* back to the start position.

The START bit is initially "`0`". When set to "`1`" the movement starts. The indicator is moved to the defined start position, regardless of its current position. The movement is active and repeated as long as the START bit is "`1`". When the START bit is set to "`0`", the movement is immediately stopped.

6.5 Update Rate Control (URC)

With the "Update Rate Control" you can limit a "sudden" large movement of the sphere (what I am calling "limit" mode) or have a possibility to decrease the amount of (position) data sent to the SDI (what I am calling "smooth" mode). As the two options (limit or smooth) are quite similar, they are mutual exclusive, that is, you can choose between no update rate control, limit mode or smooth mode.

If update rate control is not enabled, every new setpoint position is accepted "as is" and executed as one single movement. This works great if the position updates are sent frequently (say 10 position updates per second) and the difference between the positions is small. This will produce a smooth moving sphere, but at the cost of quite some communication from the PC side.

In limit mode you set a threshold limit value. If you send a new setpoint (for the position of the sphere) to the SDI, the firmware checks whether the difference between the current position and the new setpoint is smaller than the limit threshold value. If the difference is indeed smaller, the movement to the new setpoint position is executed in *one* update. If the difference is larger than the limit threshold value, the movement to the new setpoint position is executed in a number of position updates where each update equals the limit threshold value. The last update will be smaller than or equal to the limit threshold value to accurately position at the specified end position. The time delay between the successive updates can also be defined with the update rate control data. Limit mode can be useful to protect the indicator against position updates that are so large that the brusque movement might be “not so good” for it.

In smooth mode you first set a threshold limit value and then a smoothing threshold value. If you send a new setpoint (for the position of the sphere) to the SDI, the firmware uses an algorithm to split the otherwise position update movement in one single step into a number of steps with a smaller movement. If the difference between the current position and the new setpoint is smaller than the smoothing threshold value, the movement is done in one single update. If the difference is larger than the smoothing threshold value, the movement to the new setpoint position is executed in a number of position updates where each update depends on the difference, but is limited to the limit threshold value. As conditions may vary with each type of indicator that you control, you can set the time delay between the successive updates and specify the smoothing behavior. The smoothing effect can be realized using (always) 2, 4, or 8 update steps, or “adaptive” where the firmware selects 8 updates at the specified time delay, 8 updates at twice the specified update time (slower), or 16 updates at half the specified update time (faster). The smooth mode could be regarded the opposite of no update rate control. With smooth mode enabled you can send position updates at a much lower rate and allow for larger difference between the positions. The result will still be a smooth moving sphere, but now at the cost of some lag behind. *Every advantage has a disadvantage.* ☺

6.5.1 URC command data

All possibilities and options of URC are packed in one single 8-bit data byte.

The URC command (number 23) has the following format.

7	6	5	4	3	2	1	0
MODE		data bits representing information depending on the 2 mode bits					

- **Limit mode / enable/disable URC**

7	6	5	4	3	2	1	0
0	0	6 data bits representing the limit threshold value					

If the difference between the current setpoint and the new setpoint is larger than the specified limit threshold value (bits 5~0), the large movement is split in up to 8 movements of the limit threshold value. The last movement update is smaller than or equal to the limit threshold value to set the sphere at the requested setpoint position. As 6 bits represent $2^6 * 360/1024 = 22.5°$ rotation, this represents the maximum limit value when limit mode is enabled.
If the 6 data bits are all “0”, effectively the whole data byte is 0, URC is disabled.

- **SMOOTH mode**

7	6	5	4	3	2	1	0
0	1	smoothing minimum threshold value				smooth update	

In the “smoothing mode” data byte bits 0 and 1 define how the smoothing of the movement is realized. The difference between the current position and the new setpoint position is used to create the “in between” positions. If bits [1,0] is ‘00’ then the smoothing is “adaptive”. If bits [1,0] is ‘01’ then the smoothing is realized in 2 updates. If bits [1,0] is ‘10’ then the smoothing is realized in 4 steps, and if bits [1,0] is ‘11’ then the smoothing is realized in 8

steps. If adaptive smoothing is set, the SDI will choose 8 or 16 updates and may modify the delay time (half or double) depending on the position difference, defined limit threshold value and smoothing minimum value. More updates makes the movement smoother, but may take more time, resulting in a possible more “lag behind” visibility.
The smoothing minimum threshold value bits [5~2] defines the value at which smoothing will be used. The smoothing minimum threshold value equals the specified value times 4 plus 4, thus ‘`0001`’ sets the threshold level to 8, ‘`1111`’ sets the threshold level to 64. Note that on a 360° movement range the value 64 represents 64 * 360/1024 =22.5°. ‘`0000`’ is the lowest threshold level, which is 4 (approximately 1.4°)
If the difference is smaller than the smoothing minimum threshold level, the sphere moves immediately (in one step) to the new setpoint position.

- **SPEED setting**

7	6	5	4	3	2	1	0
1	0	not used	5 data bits representing step update delay in 8ms increments				

The speed setting defines the delay time used between position update steps in increments of 8 milliseconds. ‘`00000`’ is the minimum delay time (8 ms), ‘`11111`’ is the maximum delay time, 256 ms.

- **Miscellaneous**

7	6	5	4	3	2	1	0
1	1	not used					shortPath

The movement direction to a new setpoint is determined by the sign of the difference between the current position and the new position. However if the difference is larger than 180° rotation, you may want the movement rotate in the opposite direction. Moving in the opposite direction makes the roll movement go via the “shortest path”. Instead of a movement over say, 220°, the movement will be over 140° in the opposite direction. This behavior is default active (taking the shortest path). If you do not want this behavior there are two options to avoid it. First, you can send positions to the ADI that will move ROLL less than 180°. The other method is to set bit 0 to ‘`0`’, effectively disabling “shortest path”. Note that this mechanism is only available for ROLL, not for PITCH (as PITCH has end stops).

6.6 IDENTIFY command

The IDENTIFY command is only available when the USB communication connection is used. When the IDENTIFY command and an “any value” data byte is sent, the SDI returns a message in the form of “`SDI vA.B $xy`” followed by a CR (0x13) and LF (0x0A). The “`A`” and “`B`” represent the major and minor version number of the SDI firmware. “`xy`” is the hexadecimal device code as used for recognition of the DOA PHCC commands. Although the DOA device code is irrelevant for the USB communication, it can still be useful. Suppose you have more than one USB-connected SDI. Using the IDENTIFY command you can find out which SDI specific instrument is connected to a virtual COM port, because the device code (should be) unique. Note that because of this, the firmware for each SDI board is unique, but in case of PHCC DOA that is always the case.

6.7 USB debug command

The USB debug command enables or disables debug output sent to the USB connection. Obviously, this command only operates when you use a USB connection. Default USB debug is enabled. To disable USB debugging output you must send the command with the data byte ASCII character 'N'. Likewise, to enable the USB debugging output send the command with the data byte ASCII 'Y'. When USB debug is enabled a specific character is returned every time a data byte is received via the USB connection, followed by the data byte itself (in hexadecimal format). Depending on the interpretation (by the SDI firmware) of the data byte one of the following specific characters is returned leading the echoed (hexadecimal) data byte.

returned character	SDI interpretation
–	following 2 characters are hexadecimal command (first byte)
=	following 2 characters are hexadecimal data (second byte)
x	"disable watchdog" command received
#	invalid sub-address in command
!	USB receive data state machine in illegal state (firmware error ☹)

6.8 Watchdog functionality

The watchdog functionality makes sure that the data communication stays "synchronized". Commands sent via the PHCC DOA channel are 3 "bytes" (device address, sub-address and data byte, where the sub-address is 6 bits!). Commands sent via the USB channel are 2 bytes. The firmware uses a state machine to keep track of the received data bytes. If, due to some external disturbance, one byte is not correctly received the SDI receive routine may treat the next byte as the first, because the state machine is in the wrong state. If the firmware gets in this state, all subsequent commands received will be wrong. This condition is not recognized by the sender (PHCC Motherboard or the PC), but you will notice that all functionality of the SDI board no longer seems to work (because the "commands" are wrong). You could say that the firmware has become "deaf".

To solve this problem a so-called watchdog timer is implemented in the SDI firmware. In a normal condition the bytes that belong together (they form the command) are sent one after the other without much delay. Every time a data *bit* is received the watchdog timer is reloaded to some value. Every millisecond the value is decreased by one. When the watchdog timer reaches 0, the firmware state variables that control the data reception are reset to the initial state. Thus, as long as a command transmission is active the watchdog will not expire. If there is some (predefined) time no communication activity the watchdog will expire and by resetting the state variables that control the data reception the communication channel is forced to be "in sync". If some disturbance causes the communication to go out of sync, it is corrected as soon as a short pause between commands occurs. That the mechanism works is proved by "deaf" PHCC daughter boards after power-up. Without the watchdog functionality, they remain "deaf", but with the watchdog functionality they work OK. (Power-up can cause spurious pulses which can mistakenly be detected as data bits).

The watchdog functionality is by default enabled. However, if you are *manually* testing commands it may be possible that the data bytes that belong to each other to form a complete command are transmitted with delay between them. Due to the watchdog mechanism you will never succeed in sending a complete command, because the time delay between the first and second data byte is sufficient to let the watchdog do its work. Thus, your second data byte will be treated as if it is the first data byte of a next command. For this reason, it must be possible to disable the watchdog. But here you get into a "chicken – egg" problem. The watchdog command itself consists of 3 data bytes (DOA) or 2 data bytes (USB). As long as the watchdog is active, a manually sent command to disable the watchdog will fail, because of the described watchdog action. Therefore, the "disable watchdog" command has a special "variant", which does not require the data byte. This solution only works for USB (only sub-address and data byte, no address byte) as it will make the command a single byte. Sending commands via the USB connection with a PC terminal communication program (for example PuTTY) will work fine. For DOA the solution will not work, because the "disable watchdog" command is still a 2 byte message (device address, sub-address). However, if you use the PHCC TestTool for testing via the DOA connection, the watchdog activity will not be a problem. (Note that with the PHCC TestTool a message always consists of 3 data bytes).

A separate command is available to define the watchdog timer count-down value, and optionally, enable/disable the watchdog. The data format is as follows.

7	6	5	4	3	2	1	0
ENA	**0**	6 data bits representing the watchdog count-down value					

Bit 7 ("ENA") controls whether the watchdog is enabled or disabled. When bit 7 is '1' the watchdog is enabled. When bit 7 is '0' the watchdog is disabled. So, you can disable the watchdog with this command and you can disable the watchdog with the special "disable watchdog" command. The 6 bits count-down value allows a less strict setting than default set by the firmware (8). If the 6 bits are all zero ('000000'), the firmware default value is set. Note that you can set the count-down value also when the watchdog is disabled ("ENA" bit is '0').

7. Application examples

7.1 Introduction

This chapter describes real-world applications for the SDI. The SDI is used to control a synchro that drives the sphere of the F-16 ADI, and used in small "needle pointer" instruments such as the HYD PRESS A and B indicators.

The connection for the synchro coils are TL084 OpAmp outputs. These outputs are not capable of driving these coils directly, so external amplifiers are required. The amplifiers are deliberately *not* included in the design, because the amplifier design depends on the type of coil or instrument to be driven (think of output drive current or output voltage amplitude). This choice makes the SDI more a generic device for several specific instrument solutions.

You can design your own amplifiers, but check eBay and electronic kits suppliers. They sell complete kits for one, two, or even four channel amplifiers. These kits contain everything that you need, including a PCB. Sometimes you have to get an appropriate heat sink yourself, but the costs of these kits are hard to beat by Do-It-Yourself / Build-It-Yourself designs.

The specifications of the amplifier depend on the type of indicator you want to drive, and it is self-evident that you must have the specifications and connection information of the indicator.

Many instruments will probably require AC voltages of > 10V amplitude, which would imply that the amplifier power supply voltage would need to be +15V and –15V (symmetrical). However, this is not always the case, so it is important to know the specifications of the instrument! For example, the rotor voltage of small synchros is 26V and the stator voltages are 11V (all AC 400 Hz). So, sometimes a smaller and simpler amplifier is OK, sometimes you need a "bit more amplifier" due to the required higher power supply voltages. *But you can always try to drive a synchro at a lower voltage level.* If the indicator works at lower voltage levels, the magnetic forces in the instrument are just a bit weaker, causing a slower response to setpoint changes (or no response at all for small setpoint changes), and small inaccuracy of the setpoint indication. These "degradation" is acceptable for, for example, the HYDRAULIC PRESSURE indicators, but it is not acceptable for the sphere of an ADI.

7.2 Attitude Direction Indicator (ADI) ARU-50/A

The F-16 ADI (ARU-50/A) has several indicators that can be controlled.

component	drive signal type	Specifications
roll position of the sphere	synchro	3-phase 11.8V 400 Hz
pitch position of the sphere	synchro	3-phase 11.8V 400 Hz
horizontal glide slope deviation	analog	±2 mA / 1000 Ω
vertical glide slope deviation	analog	±2 mA / 1000 Ω
rate of turn indicator	analog	±1 mA / 1000 Ω
OFF flag	power supply	+24V ... +28V DC
GS flag	digital	+15V DC 3750 Ω
LOC flag	digital	+15V DC 3750 Ω
AUX flag	digital	+24V DC 1000 Ω

7.2.1 Power supply

The ADI requires two power supply voltages, +28V DC and 115V AC 400 Hz.
Pin #3 must be connected to +28V DC. I tested with +24V DC and that works fine, so there is no need for a “special” +28V power supply. +24V DC is most likely available in the pit for indicator lamps in push-buttons. Pin #26 of the ADI is 115 V AC 400 Hz.
Pin #27 of the ADI is "ground" (common zero).

7.2.2 ROLL and PITCH synchro

It is ***absolutely important*** for proper operation that the stator base angles have been correctly defined (command 25/26, 27/28, and 29/30). The base angle is specified in 10-bit precision. You must first set the low-byte 8 bits, then the high-byte 2 bits. The following table gives the values in degrees and in 10-bit values decimal (where 10 bit represents the full 360 degree range).

	ROLL synchro		**PITCH synchro**	
	offset (degree/decimal/hex)	connection pin	offset (degree/decimal/hex)	connection pin
s1baseAngle	210° / 597 / 0x255	2	240° / 682 / 0x2AA	7
s2baseAngle	330° / 938 / 0x3AA	9	0° / 0 / 0x000	8
s3baseAngle	90° / 256 / 0x100	10	120° / 341 / 0x155	19

These values are only correct when the stator coils are connected as indicated. Further, the movement range may not be a full 360°. ROLL can move the full 360°, but the PITCH range is limited from approximately 49° (140 decimal) to 247° (702 decimal) to prevent slamming against the internal mechanical end stops. If you use the "demo mode", make sure to set all values before starting a movement. Further, the values in the table are correct when the phase difference between the stator signals and the rotor is zero. *If the 2nd order 400 Hz filter is used, this filter introduces a phase shift between the rotor (filter input) signal and the stator signals (from the filter output). This phase shift introduced by the filter must be compensated with the base offset values for all 3 stator signals. The filter in the prototype introduced a phase shift of ~220°.*

⇨ Make sure that you connect ADI pin #22 to ADI pin #23 to enable the ROLL synchro.
⇨ Make sure that you connect ADI pin #30 to ADI pin #31 to enable the PITCH synchro.

7.2.3 Glide slope deviation indicators

Both horizontal and vertical glide slope deviation indicators require approximately +/-5V DC for full deflection. This analog voltage can be realized with a PWM output of the SDI, but external amplification is needed. The dedicated PWM output of the SDI neither has the required voltage swing for full deflection, so with this PWM output you also need an external OpAmp (LM324 using a symmetrical power supply, +/-12V is OK. Add a series-resistor to limit the maximum deviation.
The horizontal glide slope deviation indicator connection pins are #11 (+) and #12 (–).
The vertical glide slope deviation indicator connection pins are #16 (+) and #17 (–).

7.2.4 Rate of turn indicator

The rate of turn indicator needs approximately +/-2V for full deflection. The dedicated PWM output of the SDI is barely sufficient to drive this indicator for full swing deflection. Again, you

need an external OpAmp (LM324 using a symmetrical power supply, +/-12V). Add a series-resistor to limit the maximum deviation.
The rate of turn indicator connection pins of the ADI are #4 (+) and #5 (–).

7.2.5 GS and LOC flag

These flags disappear when +12V is connected to its pin. I use a digital output of the SDI which is connected to an UDN2981 driver connected to +24V DC. Each output has a series-resistor.
The GS flag connection pins of the ADI are #24 (+) and #25 (–), the LOC flag connection pins are #28 (+) and #29 (–). Note that the ARU-50/A also requires 115VAC 400 Hz to be present, otherwise the GS and LOC flag will not operate.

7.2.6 AUX flag

The AUX flag disappears when +24V is connected to its pin. You can use a digital output of the SDI which is connected to an external UDN2981 driver connected to +24V DC. To limit the maximum current I have added a series-resistor. The AUX flag connection pin is #6.

7.3 F-104 Hydraulic Pressure indicators

→ This has not been tested!

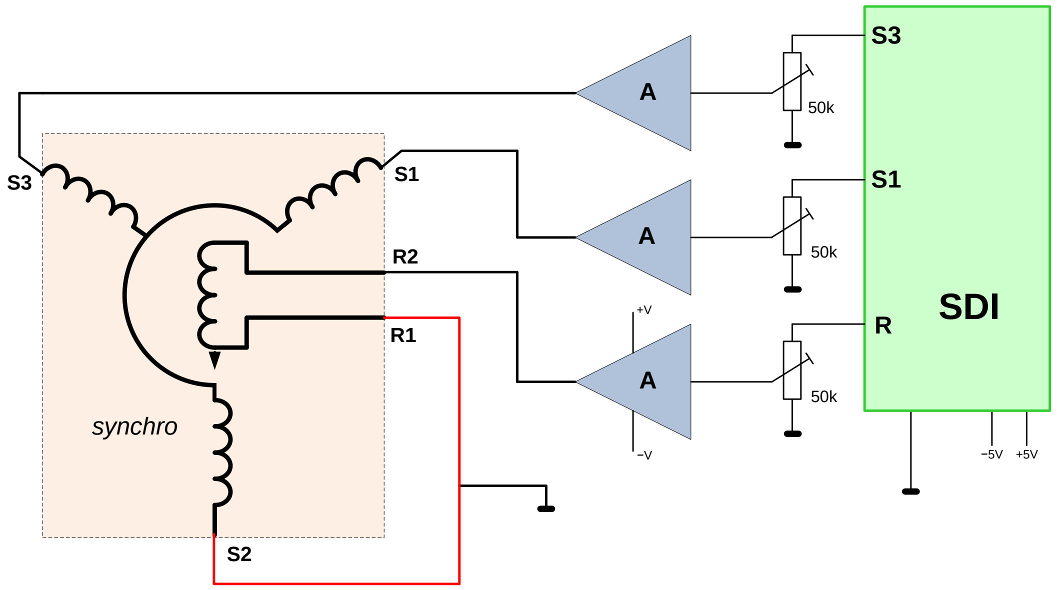
The F-104 Hydraulic Pressure indicators use a small synchro to drive the needle pointer. The rotor voltage is 26V AC and the stator voltage is 11V AC. For an initial experiment you can try whether the indicator works when the rotor voltage is also 11V AC. In that case you can use identical amplifiers and use a common (symmetrical) power supply. If the rotor amplitude really needs to be higher than the stator amplitude, you can still use amplifiers with a common symmetrical power supply. Use the trim potentiometers to set the rotor amplitude to the required level, and adjust the trim potentiometers for the stator signals such that the maximum amplitude level is below or equal to the maximum specified level.

➡ Note that an amplifier that uses a single (for example +12V) power supply voltage always includes a large capacitor to block a DC voltage level in its output and the amplitude will be at best (theoretically) just +6V/–6V. Thus, the operating voltage for a single voltage supply amplifier must be at least twice as high. *Avoid this type of audio amplifiers. Choose amplifiers that require a +/– symmetrical power supply.*

When looking for suitable amplifiers, check

- ✓ dual power supply voltage (positive and negative),
- ✓ type of amplifier IC used (so you can check its data sheet).

Normally, you connect the *three* stator coils to the S1, S2 and S3 connection of the SDI, and the rotor to the R connection of the SDI. Thus you need *four* amplifiers. However, small synchros also operate correctly when one coil is connected internally to the rotor connection, and then the S2 coil connection is not available. This is the case with the F-104 HYD PRESS indicator synchro. So, you only need three external amplifiers. The power supply for the SDI is +5/–5V, and the power supply for the amplifiers is (at least) +12/–12V.



Synchro connected to SDI

Do not connect the synchro in the initial setup, because you do not know what voltage levels are output by the amplifiers! First, set the wiper of the 50kΩ adjustment potmeters to “minimum”, that is, the wiper contact is at the GND side. Then turn on the power supplies. Connect a Voltmeter (set to AC Volt range) to the R1 and R2 connection of the synchro. Adjust the rotor potmeter until the Voltmeter reads 10V.

Send commands with sub-address 34, 36, and data value 255 to the SDI to set the output signals from the SDI to maximum. Then connect the Voltmeter to the R1 and S1 connection and adjust the S1 potmeter to 8V. Then connect the Voltmeter to the R1 and S3 connection and adjust the S3 potmeter to 8V.

Now you are ready to set the HYD PRESSURE indicator to any setpoint using the command with sub-address 0, 1, 2, 3 or 12. Try to set the indication to any position in increments of 30° physically. If you get erratic movements or no movements at all, you need to define different values for the base angle offsets (sub-address 25/26 and 29/30). If you see any “jumps” in the indication, the synchro is not correctly connected. Note that swapping the S connections can be done without damage to the synchro. Observe the indicator position with different setpoint values and make notes. From that you can reason which base angle offsets might apply or which connection(s) are wrong. Modify the base angle offsets in “increments” of 30° (converted to a decimal value 0 … 1023 or 255), keep the difference between the offset values 120°. If you swap connection, swap only one (with another), and try again.

8. Assembly

Needed tools, besides a fine low-wattage soldering iron are a multi-meter, and a small screwdriver.
Take your time to solder the components on the PCB. Better spend a few more minutes working accurately now, than searching for that little solder excess that causes a short circuit.
Soldering the components in order from smallest height to higher has the advantage that the board lays stable on your desk while soldering and keeps the component against the PCB. Therefore, the following soldering order is proposed. All components are placed on the component side of the PCB. The component side has the white text painted on the PCB (the so-called silkscreen). See the “Appendix A – Component locator” for reference.

Observe ESD safety measures to prevent static discharge damage.
(This applies to diodes, the LED and ICs)

1. Solder the two diodes. The cathode of the diode is the ‘bar’ in the symbol. The ‘bar’ shows the orientation where the ‘ring’ of the diode must be.
2. Solder all resistors, except the two trim potentiometers and R56. Make sure that the resistors with the different values are in their correct position. Check the chapter 9 for reference. If you are not sure that you read the color code correctly, use an Ohm meter.
3. Solder the crystal. I always put four strips (4 mm width) of paper sheet between the crystal and the PCB. That way you are sure that the metal housing can never make contact with traces on the PCB. Remove the paper strips after soldering the leads of the crystal.
4. Solder all 1 nF, 10 nF and 100 nF capacitors (they are small).
5. Solder all IC sockets, if you want to use sockets. I always use sockets, not to protect the ICs, but the PCB! If an IC is defective, it can easily be swapped. A soldered IC is difficult to remove and you likely damage the PCB traces or the through-hole plating. The PCB is more valuable than any of the ICs. Make sure that the notch, which indicates pin #1 location, is at the correct side. The silkscreen shows the notch. Note that the functionality checks further down this chapter are based on using IC sockets *(and only then possible)*.
6. Solder the disc-shaped capacitors (18 pF, 470 pF, and 560 pF).
7. Solder the LED. The longer lead of the LED (the anode) is “towards” IC3, the shorter lead of the LED (cathode) is “towards” the PHCC DOA connector.
8. Solder all headers. First the DOA header, then the pin headers, finally the USB connector.
9. Solder the high capacitors (220 nF, 680 nF, 1 µF). Do not yet solder the polarized capacitor C33.
10. Solder the two trim potentiometers.

Before you proceed, do a visual inspection of the board with a bright light and magnifying glass.

- ✓ Are the two diodes installed in the correct orientation?
- ✓ Are all soldering joints clean and shiny? A dull soldering joint may be a bad soldered joint.
- ✓ No small droplets of solder near the soldering joints? You can use an old toothbrush to remove residue and small droplets from the board.

TIP *You can use an old tooth brush to brush off solder residue and tiny solder droplets.*

© First functionality check that you can do.

1. Connect a +5 Volt power supply to the header POWER. GND is pin #2, +5V is pin #3. Do not connect a –5V power supply to pin #1. Although the PCB has no active components at this moment, observing correct polarity is always a good second nature ☺
2. Connect a wire from pin #6 to pin #8 of IC3.
3. Switch on the +5V power supply.
4. Check that the LED is ON.

If the LED does not turn ON, switch off the +5V power supply and check the polarity of the wires connected to the POWER connector! Check soldering joints. You can use an Ohm meter to check traces on the PCB. The anode of the LED is directly connected to the pin #3 of the POWER connector. The cathode of the LED is connected to the "top" side of R10 (next to the RST header). The other side of R10 goes to IC3 pin 6. Further, pin #8 goes directly to pin #2 of the POWER connector. If all this checks out correctly … did you solder the LED observing polarity (cathode/anode)?

Finalizing the assembly of the PCB.

1. Switch off the +5V power supply, disconnect wires connected to the POWER header.
2. Wait until the LED is OFF.
3. Remove the test wire from pin #6 to pin #8 of IC3.
4. Solder the polarized capacitor C33. *Observe correct polarity.*
5. Install all 10 ICs in their socket. See the "Appendix A – Component locator" for reference and the parts list. *Observe the correct orientation; pin #1 is at the notch side.*
   ➔ ***ICs with wrong orientation on the PCB will be damaged when power is applied!***

© Final functionality check.

1. Connect a –5/+5 Volt power supply to the header POWER. –5V is pin #1, GND is pin #2, +5V is pin #3. *Observe correct polarity!*
   ➔ ***All ICs on the PCB will be damaged if the polarity is wrong!***
2. Switch on the power supply. Observe that the DIAG LED starts flashing.
3. Switch off the power supply.
4. Put a jumper on the USB/DOA header.
5. Switch on the power supply.
6. The DIAG LED starts flashing but at the different (faster) frequency.
7. Use a Type A – Type B USB cable to connect the SDI board to a PC.
   Note that the DIAG LED flash frequency is now half the frequency seen in step 6.
8. Disconnect the USB cable. The DIAG LED flashes again with the faster frequency.
9. Switch off the power supply.

9. Parts list

9.1 Full parts list

Quantity	Component	description
1	IC1	µA741
5	IC2, IC4, IC5, IC6, IC10	TL084
1	IC3	PIC 18F2550
1	IC7	CD4053N
1	IC8	TL082
1	IC9	V2164P
2	IC socket DIL, 8-pin	high-quality machined pin socket
5	IC socket DIL, 14-pin	high-quality machined pin socket
2	IC socket DIL, 16-pin	high-quality machined pin socket
1	IC socket DIL, 28-pin	high-quality machined pin socket
2	D1, D2	1N4148
1	LED “DIAG”	3 mm, any color
1	Q1	XTAL, 20 MHz, HC49U-S
13	C1, C2, C8, C9, C11, C12, C13, C14, C15, C16, C17, C18, C21	100nF
1	C3	1µF
3	C4, C5, C32	10nF
2	C6, C7	18pF
1	C10	220nF
2	C19, C20	1nF
4	C22, C23, C24, C28	680nF
3	C29, C30, C31	470pF
3	C25, C26, C27	560pF
1	C33	47µF / 25V polarized (0.1”)
5	R1, R4, R37, R38, R39	39k
1	R2	10k trim potentiometer
2	R3, R5	1k
11	R6, R13, R21, R22, R23, R24, R25, R26, R27, R54, R55	10k
2	R7, R16	2k2
3	R8, R49, R56*	100k
1	R9	6k8
1	R10	330 Ω
13	R11, R14, R15, R17, R34, R35, R36, R40, R41, R42, R51, R52, R53	47k
1	R12	2k7
2	R18, R20	390k
1	R19	50k trim potentiometer
4	R28, R29, R30, R43	470k
4	R31, R32, R33, R47	120k
3	R44, R45, R46	560 Ω
2	R48, R50	5k6

1	USB/DOA	2-pin male header
3	POWER, RST, CTR0	3-pin male header
1	X2	2x3-pin male header
2	X1, SV2	4-pin male header
1	SYNCHRO	5-pin male header
1	DOA	2x5-pin male header with shroud
1	SV1	6-pin male header
1	USB	USB connector Type B
≤ 4	Jumpers	for selection and configuration
1	PCB	SYNCHRO DRIVE SDI V2.0

- All capacitors are non-polarized.
- Component values may change without notice.
- (*) Value may be different depending on usage.

Parts are available from Reichelt: https://secure.reichelt.de/index.html?&ACTION=20&AWKID=1127092&PROVID=2084
Note that this list does not include the VCA (IC9 – V2164P) and the SDI V2.0 PCB.

9.2 Local oscillator 400 Hz

If an external 400 Hz signal is used or required, you can omit the 400 Hz local oscillator and then following parts are not needed. This will also decrease the load of the power supplies (a little ☺). As the money saved is less than 1$, I propose to solder all components, but leave the IC socket empty.

Quantity	Component	description
1	IC1	µA741
1	IC socket DIL, 8-pin	high-quality machined pin socket
2	D1, D2	1N4148
2	C4, C5	10nF
2	R1, R4	39k
1	R7	2k2
1	R9	6k8
1	R3	1k

9.3 Band pass filter

If the 400 Hz local oscillator is used (and thus no external 400 Hz signal is available nor needed), you do not need the band pass filter for sine wave shape improvement. The following parts are not needed and can be omitted. This will also decrease the load of the power supplies (a little ☺). Again, as the money saved is less than 1$, I propose to solder all components, but leave the IC socket empty.

Quantity	Component	description
1	IC2	TL084
1	IC socket DIL, 14-pin	high-quality machined pin socket
2	C19, C20	1nF
4	R11, R14, R15, R17	47k
1	R12	2k7
2	R18, R20	390k

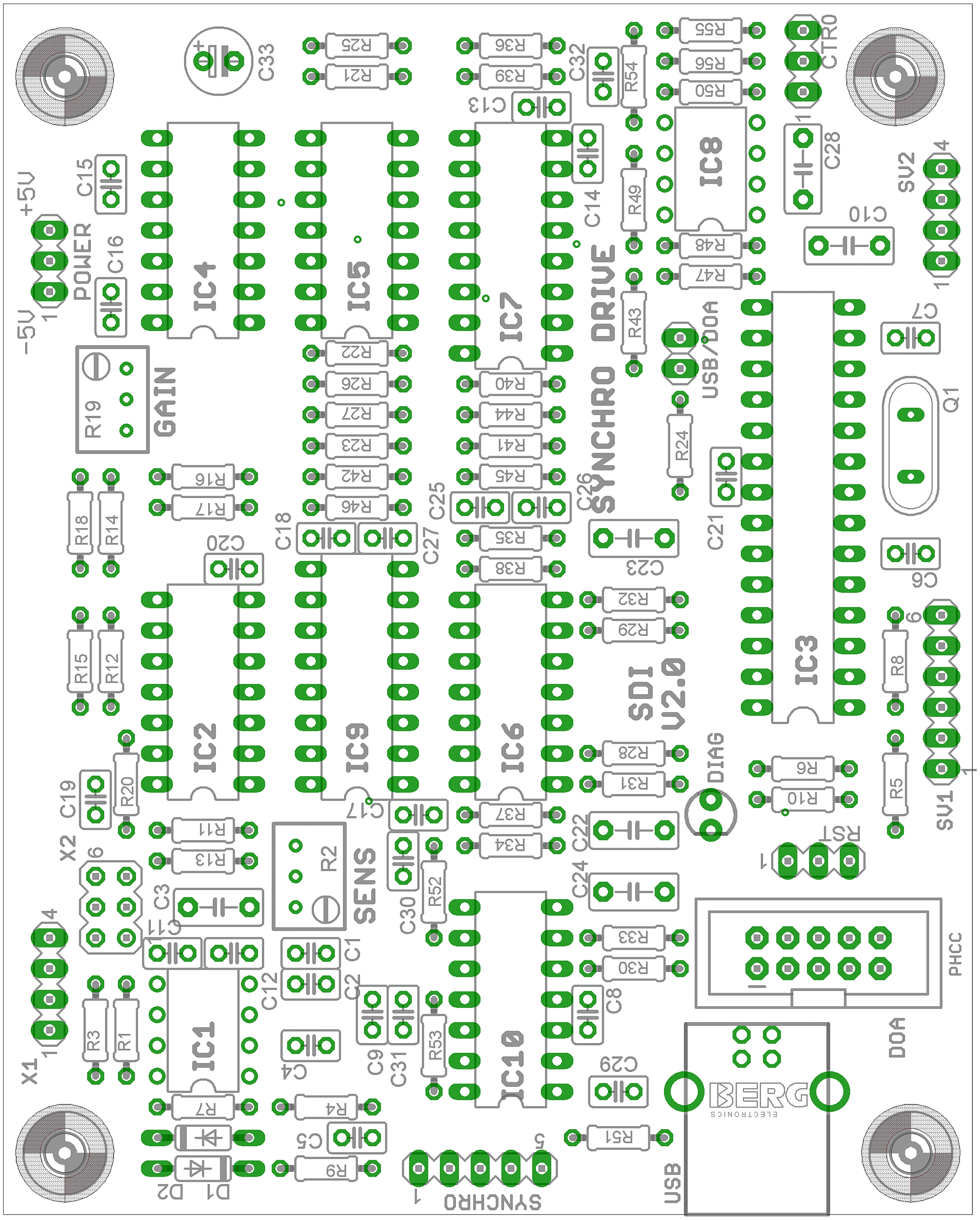
9.4 On-board analog output

If the analog output (via a PWM signal generated) is not needed, the following parts can be omitted. And again, as the money saved is less than 1$, I propose to solder all components, but leave the IC socket empty.

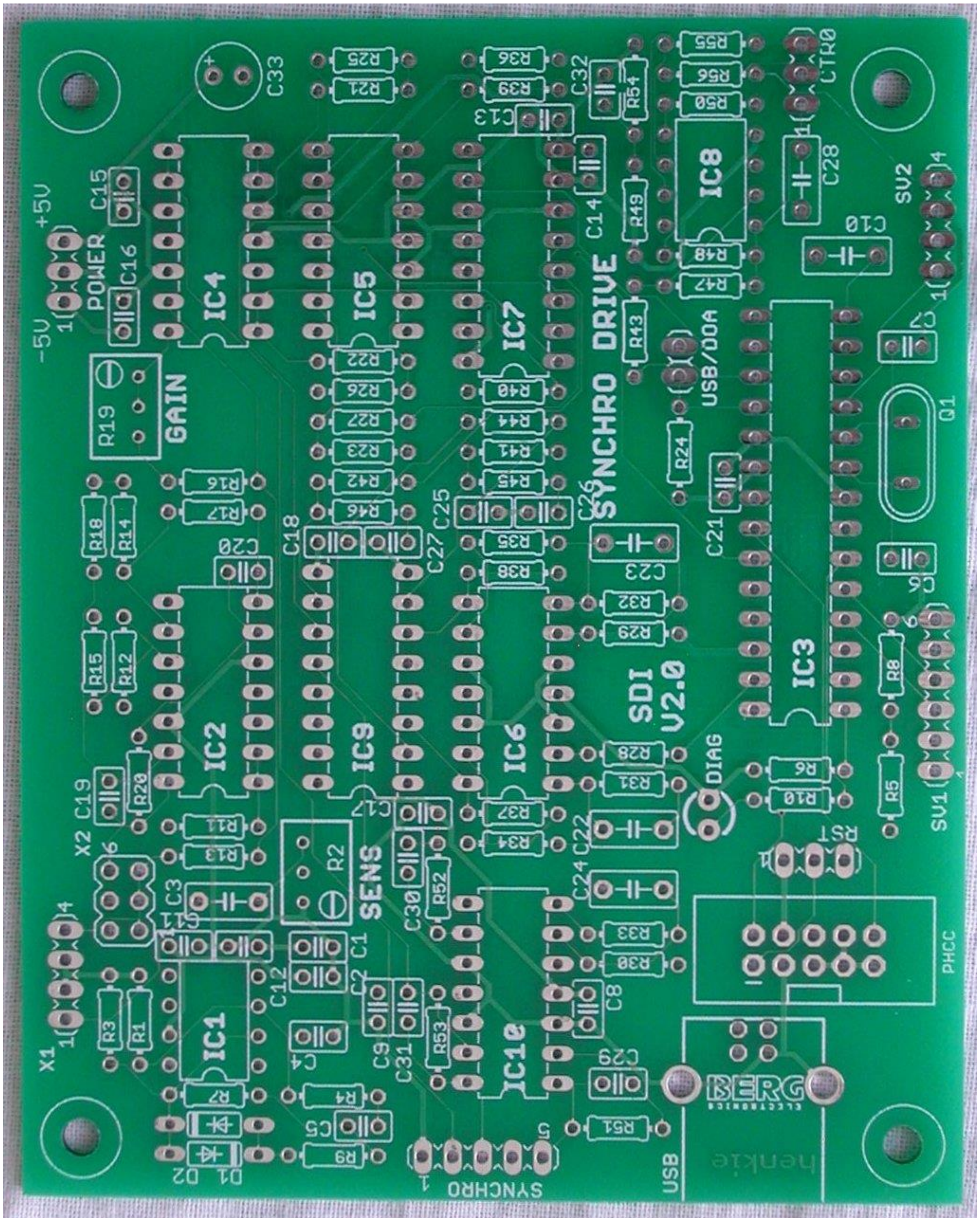
Quantity	Component	description
1	IC8	TL082
1	IC socket DIL, 8-pin	high-quality machined pin socket
1	C32	10nF
1	C28	680nF
2	R54, R55	10k
2	R49, R56	100k
1	R43	470k
1	R47	120k
2	R48, R50	5k6
1	CTR0	3-pin male header
1	Jumper	for selection and configuration

Appendix A – Component locator

This image is retrieved from the actual Eagle .BRD file. For clarity all PCB traces are removed.



Appendix B – PCB component side



Page intentionally left blank