PHCC Programmers Guide - Talking to DOA devices

Manuel Bessler

24 April 2005

This document is targetted at both hardware and software developers. It describes the data packets and board-specific configuration variables for the PHCC DOA (Digital Out Type A) daughterboards.

1 Protocol Overview

Communication with PHCC always involves the motherboard as the central ”hub” of the PHCC system. On the hardware level, data is transferred back and forth between the host (PC) and PHCC via:

- serial port
- USB port
- (maybe in the future) ethernet

Data transfer to or from PHCC is broken down into **packets**. A **packet** consists of one or more **bytes**. In this document we shall only regard packets *from the host to PHCC* as DOA transfers are one way only.

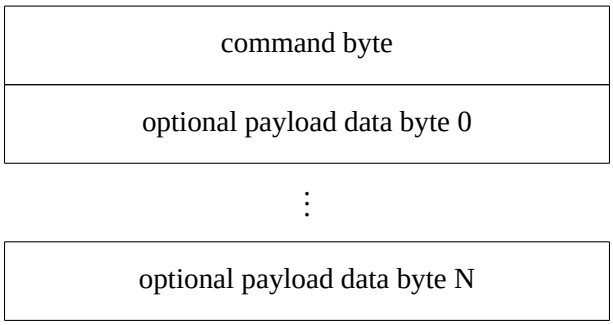

Figure 1: Packet Layout

As you can see from Figure 1, only the **command byte** is required in any case. The number of **payload data bytes** that are expected depends on the command byte.
Figure 2 lists a few common **command bytes**.
For this document, we will only concern ourselves with **DOA send** (command byte 0x07[1]).

[1]The 0xnn notation is used here as in the C programming language. It refers to numbers in the hexadecimal system.

Command Byte	Meaning
0x00	IDLE
0x01	RESET
0x02	START TALKING
0x03	STOP TALKING
0x04	dump keymatrix map
0x05	dump analog map
0x06	I2C send
0x07	DOA send
0x08	DOB send

Figure 2: common command bytes

2 The DOA Command Packet

The **DOA Command Packet** consists of the **DOA send command byte** and three bytes as payload: When PHCC receives a **DOA send command byte**, it expects three bytes to follow the **command bytes**. See figure 3.

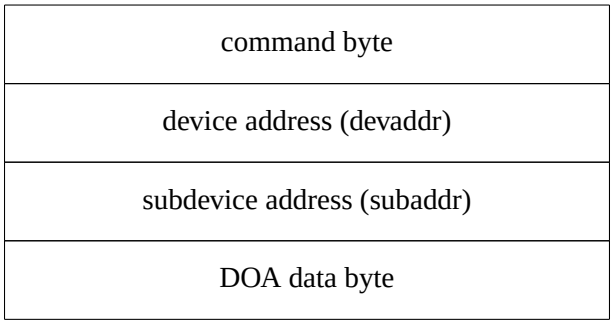

Figure 3: DOA Command Packet Layout

2.1 Device Address

The device address (or **devaddr**) is a 8bit value to address individual DOA boards. In most cases, each DOA board will have a unique **devaddr**. When the firmware for a particular DOA daughterboard is compiled or assembled (using a C compiler or PIC assembler) this address is set[2]. The 8bit width means that 256 DOA daughterboards could be theoretically addressed.
In certain cases, a single DOA daughterboard might require more than one **devaddr**, for example when the number of subaddresses is not sufficient or in case of other special requirements.

2.2 Subdevice Address

The subdevice address or subaddress (or **subaddr**) is a 6bit value to address certain functions on a particular DOA daughterboard. Note that while this is a 6bit value, the smallest unit that can be transferred

[2] usually via a `#define`

via the serial port is a byte (8bit). This means that when the host sends a **subaddr** to PHCC, it will be the size of a byte, with the two most significant bits unset. PHCC ignores the value of those two bits.
The size of the **subaddr** might change in the future, depending on needs.
The interpretation of a **subaddr** depends on the type of DOA daughterboard and possibly also firmware version. Further below (see subaddress maps, starting figure 4) is a list of subaddresses and their meaning for certain board firmware versions.

2.3 DOA Data Byte

The **data byte** is a 8bit (well of course, its a byte) value whose meaning is determined by the firmware of the receiving daughterboard in combination with the **subaddr**
Detailed information is presented in the subadress maps section.

3 Subaddress Maps

The following figures show maps for individual DOA daughterboards. The subaddr is given in hexadecimal format along with the meaning of each particular subaddr.

3.1 Subaddress Map for Daughterboard PHCC_DOA_40DO

The valid subaddresses for the 40DO board are shown in figure 4. The 40 outputs are bit-mapped onto the subaddresses. Each byte contains 8bits, so each subaddr corresponds to 8 outputs.

Subaddr	Description
0x00	outputs 1-8
0x01	outputs 9-16
0x02	outputs 17-24
0x03	outputs 25-32
0x04	outputs 33-40

Figure 4: Subaddr Map for daughterboard DOA_40DO

3.2 Subaddress Map for Daughterboard PHCC_DOA_7seg

The valid subaddresses for the LED/7segment display driver board DOA_7seg are shown in figure 5. Up to 32 displays can be controlled with one board, so 32 subaddresses are needed for direct control, one for each display. Most 7-segment displays actually have 8 LEDs embedded (7 segments and the decimal point) which suits us well as each byte also consists of 8 bits. This board allows direct control of all segments (as well as the decimal point), so no BCD or other conversion is done on the incoming data. It is the host's responsibility to „encode" the data eg. to display numbers 0-9 via the 7 segments. This also give the user freedom in wiring of the segmented displays. Instead of (or in addition to) the 7-segment displays, individual LEDs can also be used. In this case, it is necessary to connect the cathodes of

every block of eight LEDs together to form a common cathode (if the board is used in common cathode configuration, otherwise connect the anodes together).
The board has two sets of 16 displays grouped together as a ,,channel". The purpose of these channels is to group the commons together. This means that the common return of display 1 (on channel A) is connected to the first display of channel B, namely display 17.

3.3 Subaddress Map for Daughterboard PHCC_DOA_8servo

The valid subaddresses for the servo driver board DOA_8servo are shown in figure 6. To control these 8 RC-type servos, there are subadresses for position and calibration. The position is a 8bit value, so 256 different positions are possible for each servo. Each servo can be calibrated for movement range and for offset on one side of the movement range. The offset calibration value is 16bits, so two subadresses are needed for each servo. The movement range calibration value (also called ,,gain") is a 8bit value. The best strategy for calibration is to set the offset value first, and then the gain.

3.4 Subaddress Map for Daughterboard PHCC_DOA_AnOut1

The valid subaddresses for the servo driver board DOA_AnOut1 are shown in figure 7. With the standard firmware, each channel has a resolution of 8bits. For non-standard firmware versions (eg. with higher resolution but less channels) this subaddr map is not correct. Note the **gain calibration value** at subaddr 0x10, which is currently preset to 128 (decimal). Normally the gain shouln't need adjustment and it might completely disappear completely in later firmware versions.

3.5 Subaddress Map for Daughterboard PHCC_DOA_char_lcd

TBD. Standard (generic) firmware: Here, the subaddr is divided into several fields, namely * display number (0-7): 3bit, contained in the lower 3 bits (bits 0-2) * display data/control data 0/1: 1bit, contained in bit 333 display data=0, control data=1
The lcd firmware buffers the incoming data/commands in a circular buffer. This buffer can hold up to 128 data/command sequences

3.6 Subaddress Map for Daughterboard PHCC_DOA_stepper_293

TBD.

Subaddr	Description	
0x00	display 1	
0x01	display 2	
0x02	display 3	
0x03	display 4	
0x04	display 5	
0x05	display 6	
0x06	display 7	
0x07	display 8	Displays 1-16
0x08	display 9	on Channel A
0x09	display 10	
0x0A	display 11	
0x0B	display 12	
0x0C	display 13	
0x0D	display 14	
0x0E	display 15	
0x0F	display 16	
0x10	display 17	
0x11	display 18	
0x12	display 19	
0x13	display 20	
0x14	display 21	
0x15	display 22	
0x16	display 23	
0x17	display 24	Displays 17-32
0x18	display 25	on Channel B
0x19	display 26	
0x1A	display 27	
0x1B	display 28	
0x1C	display 29	
0x1D	display 30	
0x1E	display 31	
0x1F	display 32	

Figure 5: Subaddr Map for daughterboard DOA_7seg

Subaddr	Description	
0x00	servo 1	Servo Positions
0x01	servo 2	
0x02	servo 3	
0x03	servo 4	
0x04	servo 5	
0x05	servo 6	
0x06	servo 7	
0x07	servo 8	
0x08	low byte of calibration offset for servo 1	Offset Calibration value (low byte)
0x09	low byte of calibration offset for servo 2	
0x0A	low byte of calibration offset for servo 3	
0x0B	low byte of calibration offset for servo 4	
0x0C	low byte of calibration offset for servo 5	
0x0D	low byte of calibration offset for servo 6	
0x0E	low byte of calibration offset for servo 7	
0x0F	low byte of calibration offset for servo 8	
0x10	high byte of calibration offset for servo 1	Offset Calibration value (high byte)
0x11	high byte of calibration offset for servo 2	
0x12	high byte of calibration offset for servo 3	
0x13	high byte of calibration offset for servo 4	
0x14	high byte of calibration offset for servo 5	
0x15	high byte of calibration offset for servo 6	
0x16	high byte of calibration offset for servo 7	
0x17	high byte of calibration offset for servo 8	
0x18	gain calibration value for servo 1	Gain Calibration value
0x19	gain calibration value for servo 2	
0x1A	gain calibration value for servo 3	
0x1B	gain calibration value for servo 4	
0x1C	gain calibration value for servo 5	
0x1D	gain calibration value for servo 6	
0x1E	gain calibration value for servo 7	
0x1F	gain calibration value for servo 8	
0x20	store calibration values to EEPROM (not implemented yet)	

Figure 6: Subaddr Map for daughterboard DOA_8servo

Subaddr	Description
0x00	channel 1
0x01	channel 2
0x02	channel 3
0x03	channel 4
0x04	channel 5
0x05	channel 6
0x06	channel 7
0x07	channel 8
0x08	channel 9
0x09	channel 10
0x0a	channel 11
0x0b	channel 12
0x0c	channel 13
0x0d	channel 14
0x0e	channel 15
0x0f	channel 16
0x10	gain calibration value for all channels

Figure 7: Subaddr Map for daughterboard DOA_AnOut1