FG085 miniDDS Function Generator

How to Control FG085 via Serial Port

Applicable Models: All FG085 variants
Applicable Firmware: 113-08512-013

The firmware 113-08512-XXX implemented a serial protocol by which FG085 can accept button codes from its serial port. This will allow the FG085 be controlled by PC or another microcontroller (hardware mod required for the latter). This design note explains the technical details of the protocol and presents a suggested scheme for the modification.

1. FG085 Serial Port and Communication Parameters

FG085 serial port is the UART of U5 (ATmega168). At existing design (PCB version 109-08500-00G) it has been connected to the Uart-USB bridge chip U15 (CP2102), which in turn can be connected to PC (see schematic for connection details). The communication parameters of the serial connection are fixed to the following values.

Data bits: 8
Stop bits: 1
Parity: None
Baudrate: 115200 bps
Flow control: None

External device must be set to the same parameters to be able to communicate with FG085.

2. Button Code Reception Protocol

Button codes are sent to FG085 in frames. The frame structure shown in the table 1.

Offset	Field Name(size)	Value
-1	Sync character (1 byte)	0xFE
0	Frame ID (1 byte)	0xFB
1	Frame Size (2 bytes)	0x0006 (little endian)
3	Button code (1 byte)	See table 2
4	Button parameter (1 byte)	See table 2
5	Reserved (1 byte)	0x00

Table 1.

Because microcontrollers (or PC) could send button codes much faster than that from key pad it will cause button codes lost if FG085 can not handle quick enough. To prevent this from happening you can send special button code 0xA1 which enables postpone code reception until the present code has been processed. Note that at powering-up this prevention is disabled by default. The 0xA1 code must be sent at least once to enable it.

The reception buffer of FG085 is 300 bytes in depth. It is big comparing to the frame size of button code. But you will still risk losing your button codes if they are sent fast consecutively without the protection turned on. It is strongly recommended to turn the function on.

For each successfully received frame FG085 returns with the character ‘G’ (means “Good”). Otherwise it returns the question mark ‘?’ or nothing.

3. FG085 Button Codes

In FG085 each button is represented by a number (button code) and an associated parameter. Table 2 below is a list all button codes and their parameters used in FG085. Please note that the button parameters are mandatory for some buttons while for others they are optional. For future compatibility is suggested to always send button codes with their correct parameters (as given the Table 2) followed.

Table 2.

Button Name	Button Code	Button Parameter		Comments
		Character	Value	
1	0x01	'1'	0x31	
2	0x02	'2'	0x32	
3	0x03	'3'	0x33	
4	0x04	'4'	0x34	
5	0x05	'5'	0x35	
6	0x06	'6'	0x36	
7	0x07	'7'	0x37	
8	0x08	'8'	0x38	
9	0x09	'9'	0x39	
0	0x0A	'0'	0x30	
+/-	0x0B	'+'	0x2B	
. (dot)	0x0C	'.'	0x2E	
ESC	0x0D	'E'	0x45	
WF	0x0E	'W'	0x57	
Hz	0x0F	'H'	0x48	
KHz	0x10	'K'	0x4B	
MODE	0x11	'M'	0x4D	
FREQ	0x12	'F'	0x46	
AMP	0x13	'A'	0x41	
OFS	0x14	'O'	0x4F	
AUX	0x15	'X'	0x58	ADJ dial pushbutton
CW	0x16		0x01	ADJ dial clockwise turn
CCW	0x17		0x00	ADJ dial counter-clockwise turn
	0xA0		0x00	This special button code turns code-loss protection off.
	0xA1		0x00	This special button code turns code-loss protection on.
	0xA2		0x00, 0x01, or 0x02	This special button code places cursor to location indicated by the parameter byte with 0x00 for Freq., 0x01 for Amp, and 0x02 for Offset. This code only works under CW mode.
	0xA3		0x00 – 0x07	This special button code sets waveform indicated by the parameter byte. Number/waveform relationship: 0x00 – Sine, 0x04 – RMP- 0x01 – Square, 0x05 – STR+ 0x02 – TRI, 0x06 – STR- 0x03 – RMP+, 0x07 – USER

4. Hardware Modification

If you want to control FG085 from a microcontroller you are probably want to send the signal directly in TTL level. In this case you need to modify hardware to make the connection because the existing design does not provide a dedicated TTL-level serial port. Suggested modification is shown in the schematic of Fig. 1. Fig. 2 shows the board area of modification.

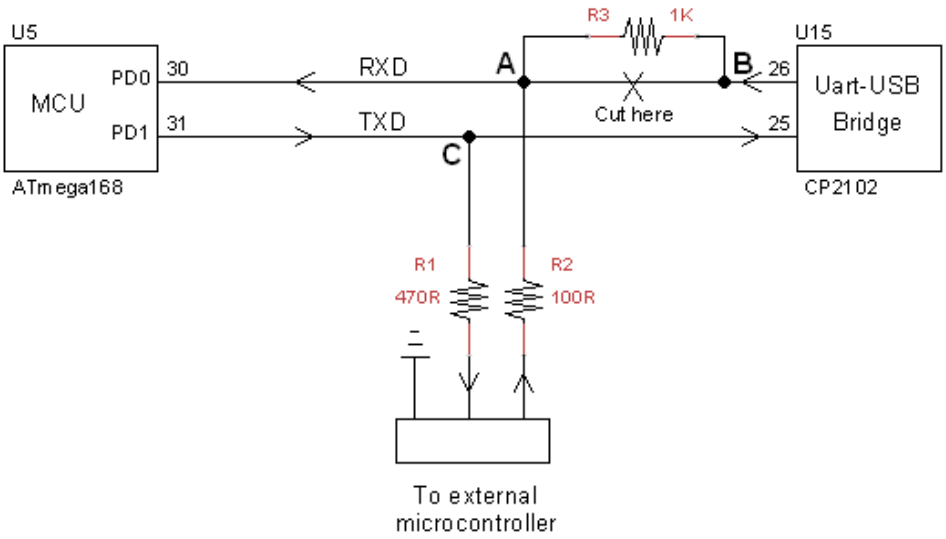

Fig. 1

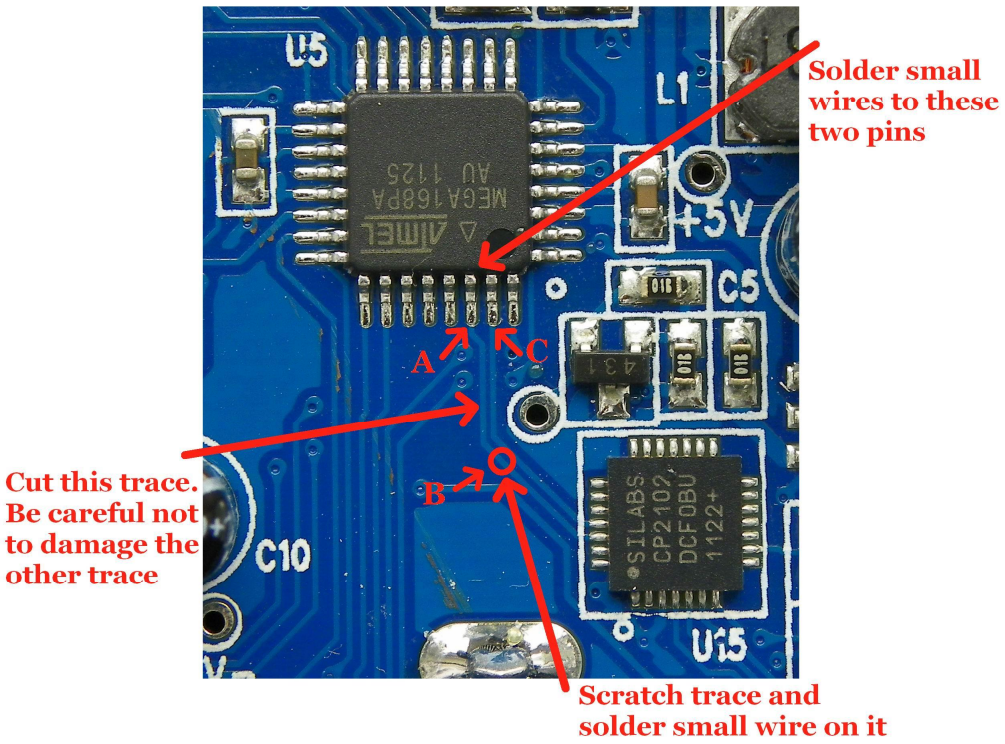

Fig. 2

Steps to follow:

1) Cut the trace carefully as illustrated in Fig. 2. Pay great attention not to damage the

adjacent trace.

2) Scratch a small section of trace at spot B on the side leading to U15. Tin it. It is not recommended to solder on pins of U15 because it is too easy to get pins bridged.
3) Solder small (not bigger than AWG 32) hook-up wires to the spots A, B, and C. Fasten them with hot glue or silicone.
4) Install resistors at the other ends as proper to your project and bring them to your microcontroller.

Considerable soldering experience is asked for this modification. If you feel your skill is not up to the job train yourself using some garbage boards before real work.

The resistor R3 can be omitted if USB connection is not to be used any more. But the trace cut is still required.

Revision History

Version	Date	Summary
v01	2013.07.29	First created
v02	2013.08.16	Changes for firmware revision 113-08512-11. Removed NoCodeLoss command. Added special button code 0xA1 as substitute.
v03	2013.10.07	Added explanation for special button code 0xA2.
V04	2013.10.28	1) Added explanation for special code 0xA3 2) Fixed a bug that made amplitude can not be set to zero.