http://www.didel.com/ info@didel.com

www.didel.com/TerSer.pdf

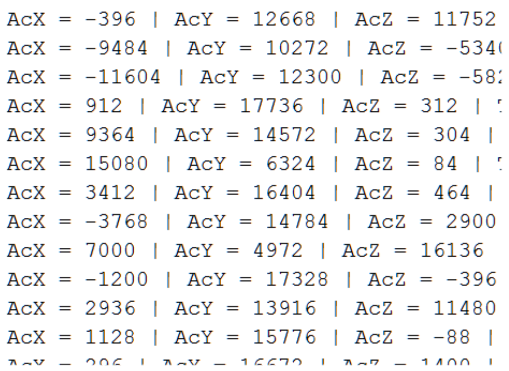
# TerSer.h

**Use an include file to replace the Arduino Serial.print**

***Now on Git: https://git.boxtec.ch/didel/TerSer***

**What is the problem with Arduino Serial.print ?**

Serial.print is most of the time used as a debugging tool, and with limited resources it is even more important to have the most lighweight solution for this task.
Also when Serial debugging is used to display sensor values, as shown opposite, suppressing non significative zeros is not adequate for tabular data, especially when the screen is scrolling.
Serial.print is a good example of the "law of the instrument": We use a bad tool, easily available, and we do not think to find or invent a better one.
We propose to use a portable "include" file, easy understand and adapt if necessary. Compatibility with an Oled display is a further plus. The SerTerm.h is demonstrated on the popular Arduino environment, but it is just plain C.



```
AcX = -396 | AcY = 12668 | AcZ = 11752
AcX = -9484 | AcY = 10272 | AcZ = -534
AcX = -11604 | AcY = 12300 | AcZ = -58
AcX = 912 | AcY = 17736 | AcZ = 312 |
AcX = 9364 | AcY = 14572 | AcZ = 304 |
AcX = 15080 | AcY = 6324 | AcZ = 84 |
AcX = 3412 | AcY = 16404 | AcZ = 464 |
AcX = -3768 | AcY = 14784 | AcZ = 2900
AcX = 7000 | AcY = 4972 | AcZ = 16136
AcX = -1200 | AcY = 17328 | AcZ = -396
AcX = 2936 | AcY = 13916 | AcZ = 11480
AcX = 1128 | AcY = 15776 | AcZ = -88 |
AcX = 296 | AcY = 16672 | AcZ = 1400 |
```

Testing the Gy521 accelerator/gyro with Serial.print

It is admittedly very convenient to just use Serial.print(var); since it doesn't require you to specify the type of variable. Though when trying for example to output data in tabular style the processor needs to know the data type used so that it can reserve the adequate space for it in its output. It might seem like a big inconvenience to specify the data type with any statement that outputs data over serial but then again for debugging purposes it makes perfectly sense in terms of speed AND size of the resulting code. By the same reasons we also do not use a buffer as it is needed only in specific situations.
The TerSer.h offer the choice of 4 print format for numbers. Signed variables have a + or – sign in front.

| Normal moz=0 | Spaces moz=1 | Zeros moz=2 | Compact moz=3 | Serial.print |
|---|---|---|---|---|
| `     0` | `+     0` | `+00000` | `0` | `0` |
| `    +5` | `+     5` | `+00005` | `+5` | `5` |
| `+23124` | `+23124` | `+23124` | `+23124` | `23124` |
| `  -200` | `-   200` | `-00200` | `-200` | `-200` |
| `   200` | `    200` | ` 00200` | ` 200` | `200` |
| 800- 900 bytes, 11 variables | | | | 1880 b, 192 var |

## How to switch?

You are used to the Arduino terminal. Remove `serial.begin(9600);` in the Setup.
Add `#include "TerSer.h"` before the setup and `SetupTerSer();` inside the setup()
Use our names, or add an include file to be compatible with Arduino names. However, you need to change on your program all Serial.print by Serialprint (the dot is not accepted in #define).
Easy to change: Ctrl H .p -→ p

Example of #define list:

```
 #define Serialprint(x) Text(x)
#define Serialprint(x,BIN) Bin8(x)  or Bin16(x)
 #define Serialprint(x) Dec16(x)
```

## TerSer installation

Search for Github/Boxtec/TerSer and download TerSer.h and TestTerSer.zip that includes TestTerSer.ino test progam. Check functions, modify, get familiar.

## Functions list

**`Car(cc);`** Send code `cc` to the UART

`Car(' ');` `Car(32);` `Car(0x20);` All three print a space

**`cc=Get();`** Wait for a key depressed. Use Teraterm, too tricky with Arduino Terminal

`Car (Get();` Echo of the typed character

**`Text("abcd");`** Send the text and add a space.

**`Textln("abcd");`** Same with CRLF.

**`CR();`** Send a CR-LF

**`Bin8(v);`** Display the 8 low bits of the variable v, converted to integers (all add a space . )

`Bin8(33);` → `00100001.`

**`Bin16(v);`** Display the 16 bits of the variable v, with a dot in the middle

`Bin8(1035);` → `00000100.00001011.`

**`Hex8(v);`** Display the 2 nibbles of the variable v, converted to integers

`Hex8(33);` → `21.`

**`Hex16(v);`** Display the 4 nibbles of the variable v, converted to integers

`Hex16(1035);` → `040B.`

**`Hex32(v);`** Display the 8 nibbles of the variable v, converted to integers

`Hex32(260);` → `00000104.`

**`Dec8(v);`** Display the variable v, with its signs if signed and non-significative space or zeros

`Dec8(33);` → `+.33.` or `+033.` or `.+33.`

**`Dec16(v);`** Display the variable v, with its signs if signed and non-significative space or zeros

`Dec16(1035);` → `+.1035.` or `+01035.` or `.+1035.`

`Dec8u(unsigned);` `Dec8s(signed);` Shorter code

`Dec16u(unsigned);` Dec16s(signed); Shorter code

`Normal();` `..+nnn.` default mode ( a dot represent a space

`Spaces();` `+..nnn.` best for tabular data

`Zeros();` `+00nnn.` if you prefer

`Compact();` `+nnn.` same as Arduino, but a + is shown if positive signed variable

## Limitations

TerSer does not print floating point numbers. This may happen some day with one more file to import, e.g. named TerFloat.h.

## Baud rate

Speed depends on UBRR0 register. Update value if required at the beginning of TerSer.h file.

```
void SetupTerSer() {
  UBRR0= 103;  //  9600 bits/s
  UCSR0B=0x18; // -- -- -- rxe txe -- -- --
  UCSR0C=0x06; // set mode: 8 data bits, no parity, 1 stop bit
}
```

| 2400 | 4800 | 9600 | 19200 | 115200 |
|---|---|---|---|---|
| **`UBRR0= 416;`** | **`UBRR0= 207;`** | **`UBRR0= 103;`** | **`UBRR0= 51;`** | **`UBRR0= 6;`** |

Atmega 328 datasheet document all possibilities with several clock frequencies.

## Note

Dec8(); and Dec16(); use a tricky macro to recognize the signed or unsigned data type. The parameter must be a single variable, e.g. Dec8(var); Dec8(var+2); will give wrong results. Code will be shorter and there will be no limitation if you use Dec8u(any unsigned expr); and Dec8s(any signed expr);, same of course Dec16u(); and Dec16s();

## Test programs on TerSer.zip - *load [www.didel.com/TerSer.zip](www.didel.com/TerSer.zip)*

```
//TestTerSer.ino Serial Terminal

#include "TerSer.h"

int16_t v16s; uint16_t v16; uint8_t v8;
void setup() {
  SetupTerSer();
}
void loop() { // empty 456 bytes 16 var
  Textln("Test TerSer");
  v8= 19; Bin8(v8); CR();
  v16= 1000; Hex16(v16); CR();
  v16s= 0; Dec16(v16s);CR();
  v16s= 5; Dec16(v16s);CR();
  v16s= 23124; Dec16(v16s);CR();
  v16s= -200; Dec16(v16s);CR();
  v16= 200; Dec16(v16);CR();
 for(;;);
}
```

```
/*
Size is 1242 bytes,
       33 var
(on my PC)
Result on the
terminal:
(mode Compact)
Test TerSer
00010011
03E8
      0
     +5
+23124
  -200
   200
```

```
// TestSerial.ino   Compare with
TestTerSer.ino
void setup() {
 Serial.begin(9600);
}
int16_t v16s; uint16_t v16; uint8_t v8;
void loop() {
  Serial.println("Test Serial");
  v8= 19; Serial.println(v8,BIN);
  v16= 1000; Serial.println(v16,HEX);
  v16s= 0; Serial.println(v16s);
  v16s= 5; Serial.println(v16s);
  v16s= 23124; Serial.println(v16s);
  v16s= -200; Serial.println(v16s);
  v16= 200; Serial.println(v16);

 for(;;);
}
```

```
/*
Size is 1966 bytes,
        205 var
(on my PC)
Result on the terminal:
Test Serial
10011
3E8
0
5
23124
-200
200
/*
```

Last update 190327