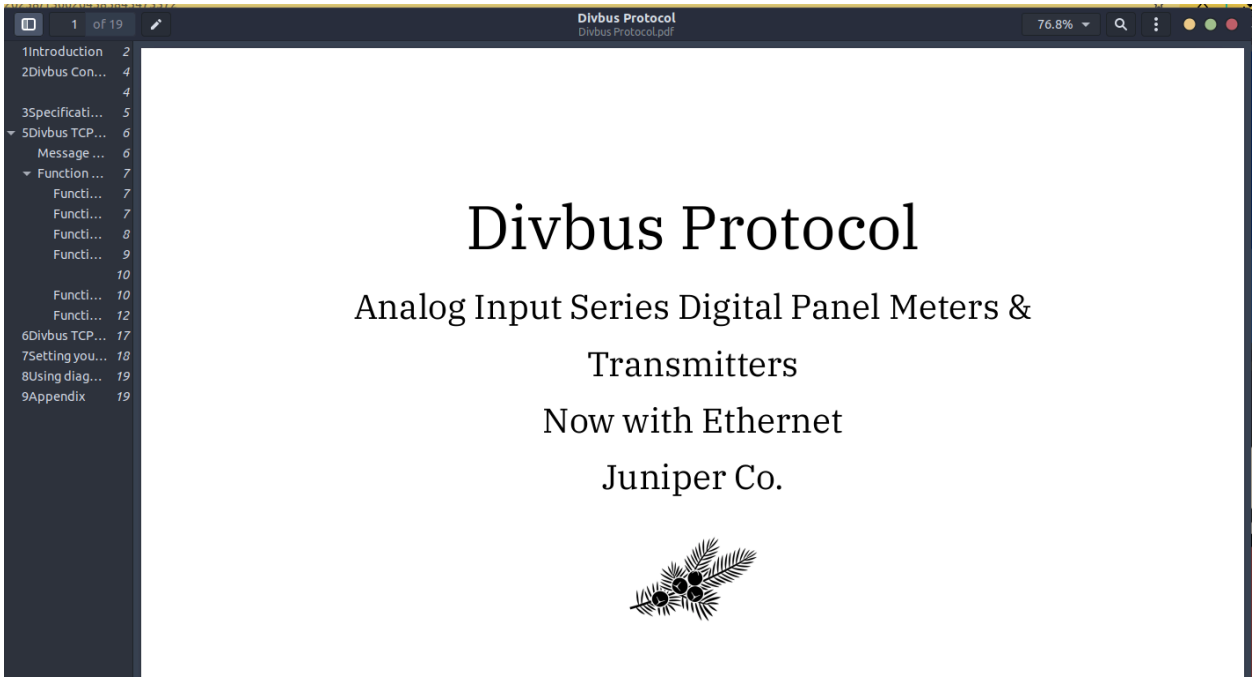
THERE WILL BE SIGNS WRITEUP

You are provided with a PCAP and a document

The document describes the made-up “Divbus Protocol” which is what is fond in the PCAP



Divbus is basically just Modbus, here’s a decent intro to Modbus:
https://www.csimn.com/CSI_pages/Modbus101.html

To understand this made-up protocol you probably need to understand Modbus, because the function codes used by Divbus are the same.

You were looking for specifically for the traffic controlling a 4 digit segment display.
Which is referenced here:

And here:

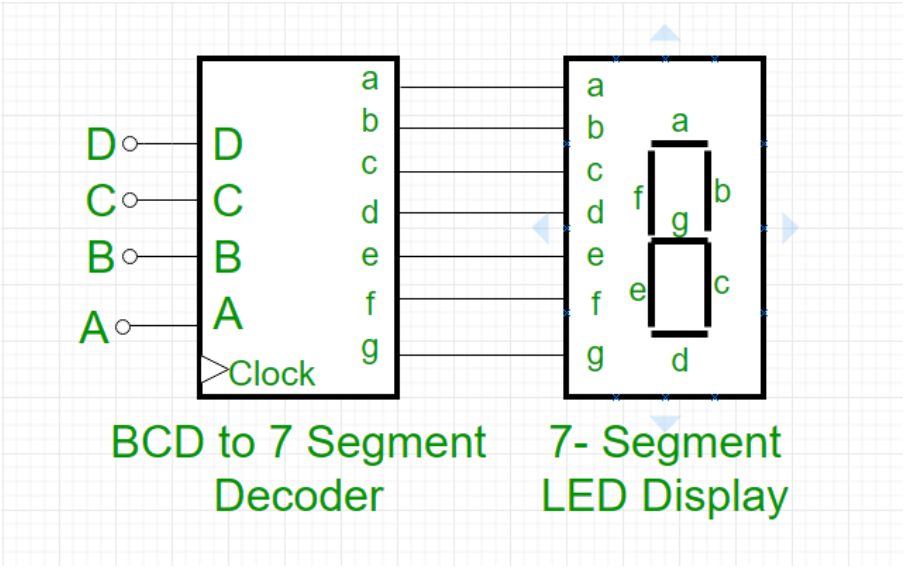
Function Code 01 - Connection Setup

Register		Name	Description
Dec	Hex		
110	00 74	Device Type	Bits 7:0 01 = DPM Meter 02 = Scale Meter 03 = Counter/timer Meter 05 = DPM transmitter 06 = Scale transmitter 07 = Counter/timer transmitter 08 - Segment Display

And most importantly here in the “Function Code 06: Write Single (Holding) Register” table:

		Scale Meter)	
96	00 60	Digit 1 Segment Display Data	Binary Coded Decimal (BCD)
97	00 61	Digit 2 Segment Display Data	Binary Coded Decimal (BCD)
98	00 62	Digit 3 Segment Display Data	Binary Coded Decimal (BCD)
99	00 63	Digit 4 Segment Display Data	Binary Coded Decimal (BCD)
100	00 64	Lockout 1	LCK_t

Digit N Segment Display Data - is just the digit being displayed on the physical segment display.
Binary Coded Decimal:



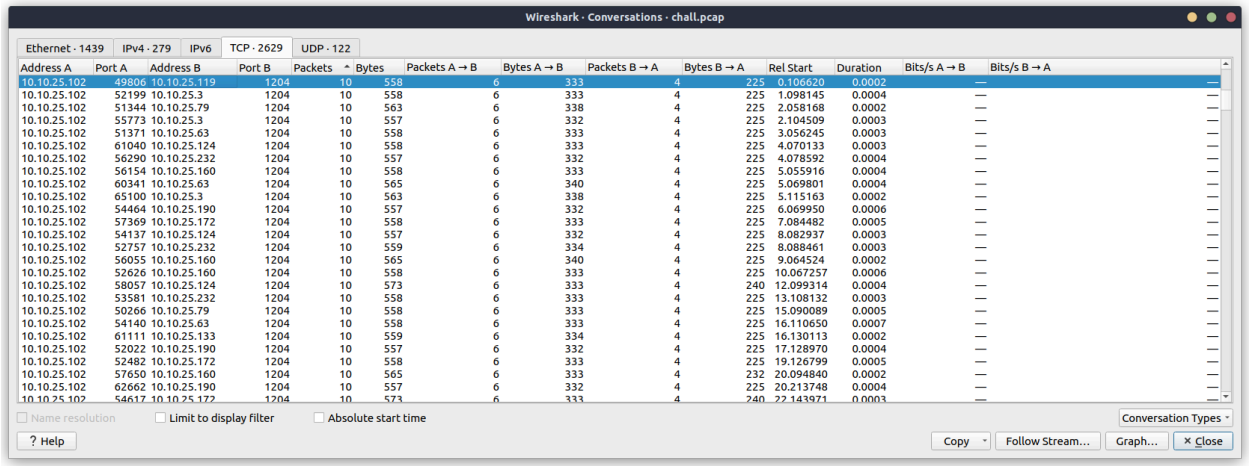
BCD		Decimal
0000	=	0
0001	=	1
0010	=	2
0011	=	3
0100	=	4
0101	=	5
0110	=	6
0111	=	7
1000	=	8
1001	=	9

The format of these packets is also described int the PDF:

05	Request	1	MA	NB	FC	RA	RA	WW	WW	CL	CH			
	Response	1	MA	NB	FC	RA	RA	WW	WW	CL	CH			
06	Request	> 2.5	MA	NB	FC	RA	RA	DD	DD	CL	CH			
	Response	> 2.5	MA	NB	FC	RA	RA	DD	DD	CL	CH			
08	Request	1	MA	NB	FC	SF	SF	WW	WW	CL	CH			
	Response	1	MA	NB	FC	SF	SF	DD	DD	CL	CH			
10	Request	1	MA	NB	FC	NR	NR	RA	RA	NBD	DD	DD	CL	CH
	Response	1	MA	NB	FC	NR	NR	RA	RA	CL	CH			

MA	Meter Address
NB	Number of Bytes in Message Frame
FC	Function Code
DD	Data Bytes

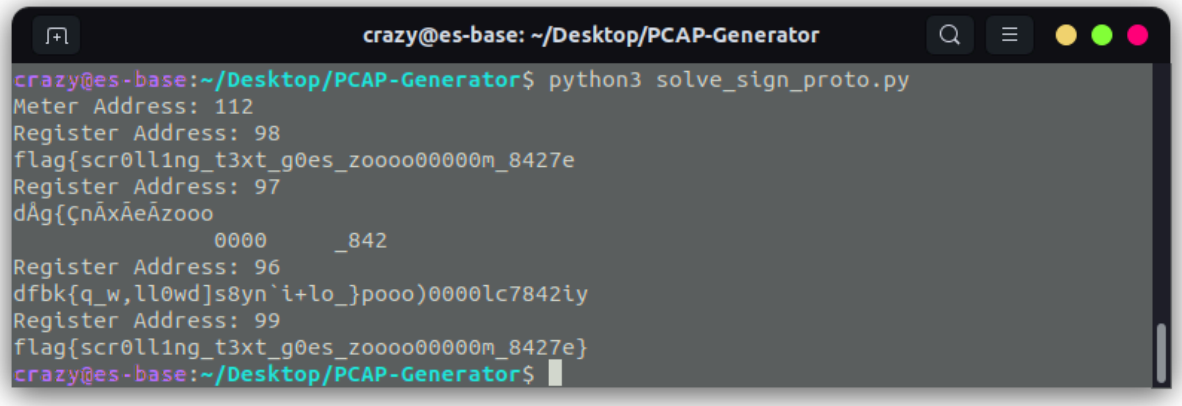
Divbus uses ports 1201-1204, we can tell by looking at the protocol statistics:



Now we write a script that gets traffic on ports 1201-1204 that writes to one of the segment display registers (ie. `[96, 97, 98, 99]` described above):
We keep store the data written to each data separately:

```
from scapy.all import *
import struct

controller_ip = "10.10.25.102"
node_ip = "10.10.25.119"
divbus_ports = [1201, 1202, 1203, 1204]

noToSegmentHex = {
   0x3f: 0, 0x6: 1, 0x5b: 2, 0x4f: 3, 0x66: 4, 0x6d: 5,  0x7d: 6, 0x07: 7,
0x7f: 8, 0x6f: 9
}


reg_address = [96, 97, 98, 99]

r = rdpcap("chall.pcap")
n = len(r)
x = ""
flag = ""
meters_display_registers = {}
for i in range(n):
   pkt = r[i]
   if Raw in pkt and pkt.haslayer(TCP) and (pkt[IP].src in controller_ip or
pkt[IP].dst in node_ip):
       # and (pkt[TCP].sport in divbus_ports or pkt[TCP].dport in
divbus_ports):
```

```
        payload = pkt[TCP].load

        #print(payload)
        ma = struct.unpack_from('<B', payload, 0)[0]
        nb = struct.unpack_from('<B', payload, 1)[0]
        fc = struct.unpack_from('<B', payload, 2)[0]
        ra = struct.unpack_from('<H', payload, 3)[0]
        #print(fc)
        if fc == 6 and ra in reg_address:
            data = struct.unpack_from('<H', payload, 5)[0]
            if data in noToSegmentHex:
                # add the strs of these 2 outputs convert to int then char
                d = str(noToSegmentHex[data])
                x += d
                if len(x) == 3:
                    if not ma in meters_display_registers:
                        meters_display_registers[ma] = {}
                    if not ra in meters_display_registers[ma]:
                        meters_display_registers[ma][ra] = []

                    c0 = chr(int(x))
                    meters_display_registers[ma][ra] += [c0]
                    x = ""


for k, v in meters_display_registers.items():
   print(f'Meter Address: {k}')
   for k0, v0 in v.items():
       print(f'Register Address: {k0}')
       print(''.join(v0))
```


crazy@es-base: ~/Desktop/PCAP-Generator
crazy@es-base:~/Desktop/PCAP-Generator$ python3 solve_sign_proto.py
Meter Address: 112
Register Address: 98
flag{scr0ll1ng_t3xt_g0es_zoooo00000m_8427e
Register Address: 97
dÅg{ÇnĀxĀeĀzooo
0000 _842
Register Address: 96
dfbk{q_w,ll0wd]s8yn`i+lo_}pooo)0000lc7842iy
Register Address: 99
flag{scr0ll1ng_t3xt_g0es_zoooo00000m_8427e}
crazy@es-base:~/Desktop/PCAP-Generator$