Analysis of a door locking system

Rémi Audebert Pierre Surply

2014-07-17

Introduction

The situation

- The door lock is broken: fix-it!
- Power is working, the door is always locked
- Control is not working

This talk is about...

- Part 1: Electrical reverse engineering
  - Power, signals, 9N1, ...
- Part 2: Hardware reverse engineering
  - Microcontrollers, converters, PHY, ...
- Part 3: Software reverse engineering
  - PIC16F87, architecture, banking, ...
- Part 4: A practical use of this knowledge
  - The big picture, controlling a door ourself

Signals

Probing

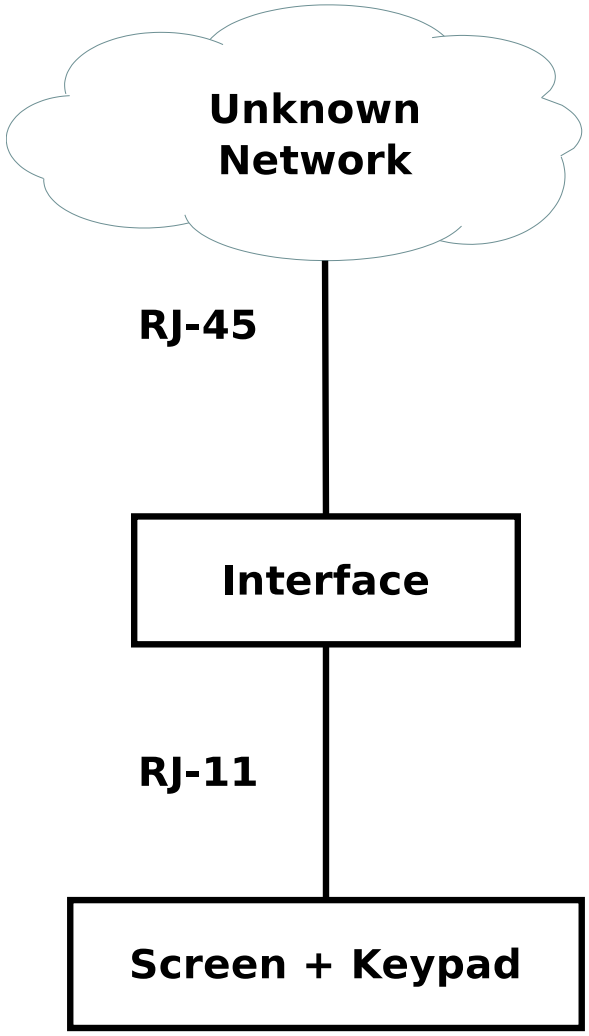


Figure 1: Probe points

Probing

- Our goal: identify signals
- Possible signals:
  - Power
  - **Ground(s)**
  - Clock
  - Data
  - Differential data
  - Pull up

Tools

- Multimeters
- Digital Oscilloscope (ATTEN ADS1102CAL 100MHz)
- Logic analyser (Saleae Logic)

Probing

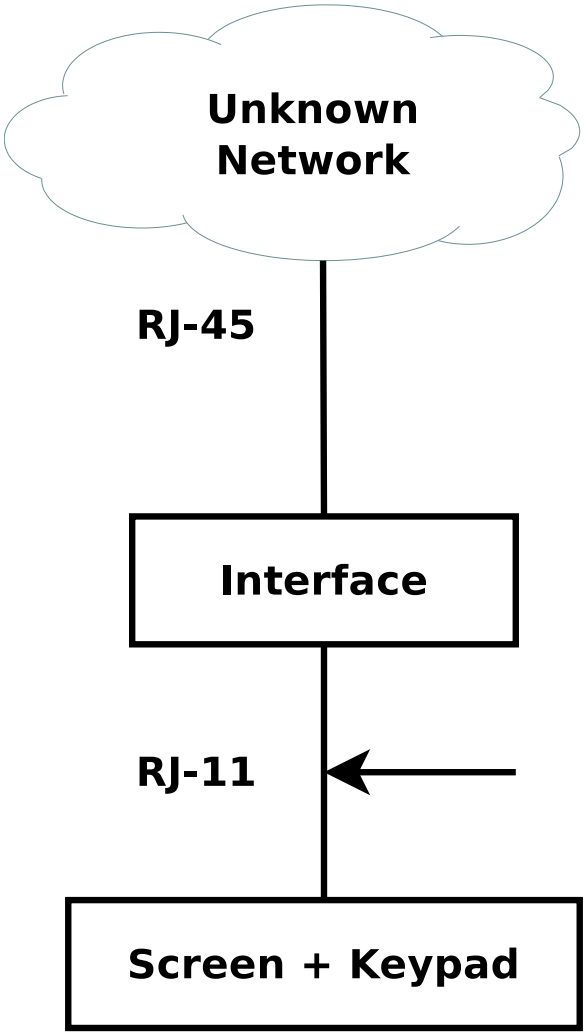


Figure 2: Probe points

Pinout

- 5V
- Ground
- ?
- ?
- ?
- ?

Frame analysis

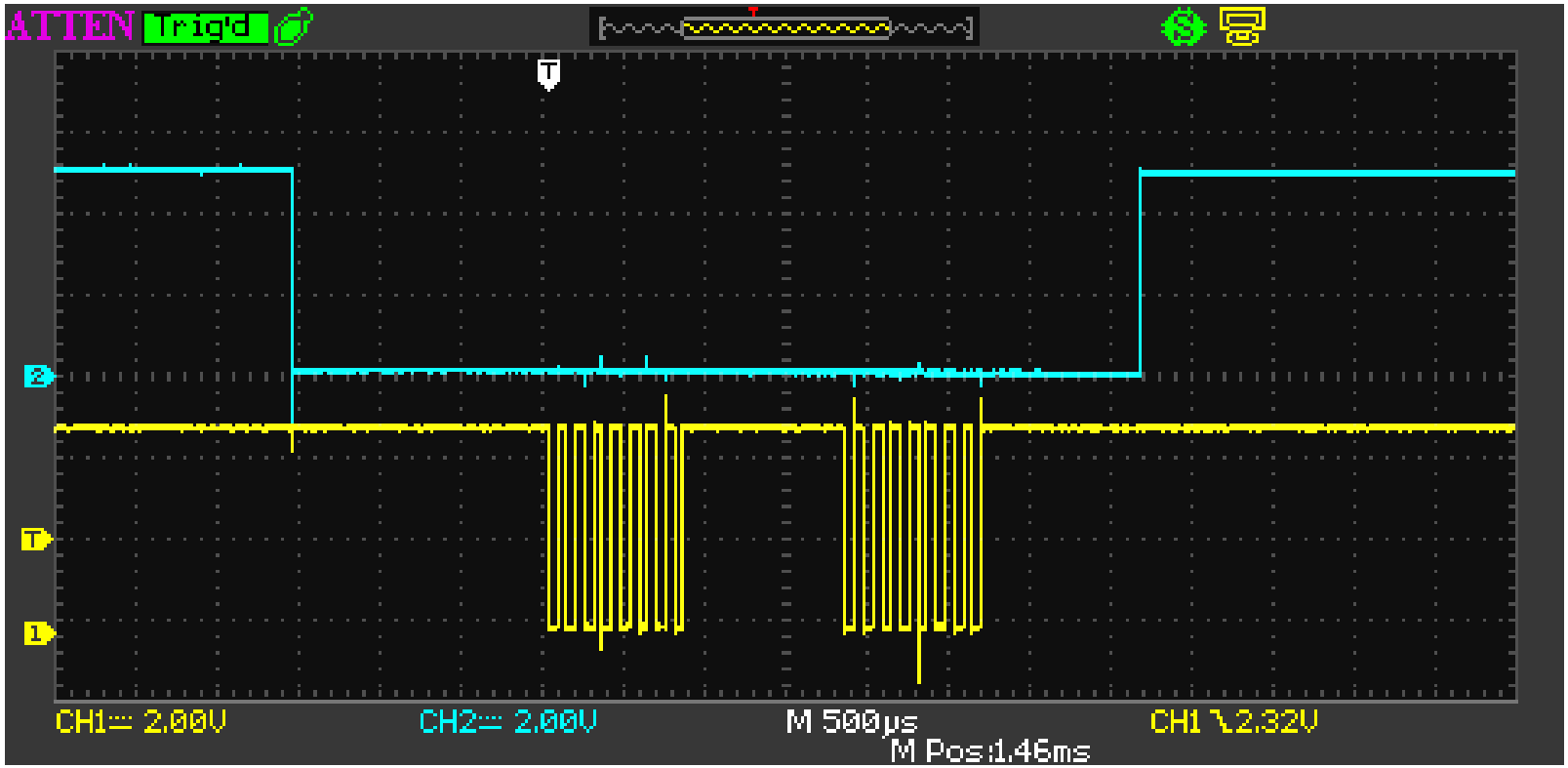


Figure 3: Frame

Frame analysis

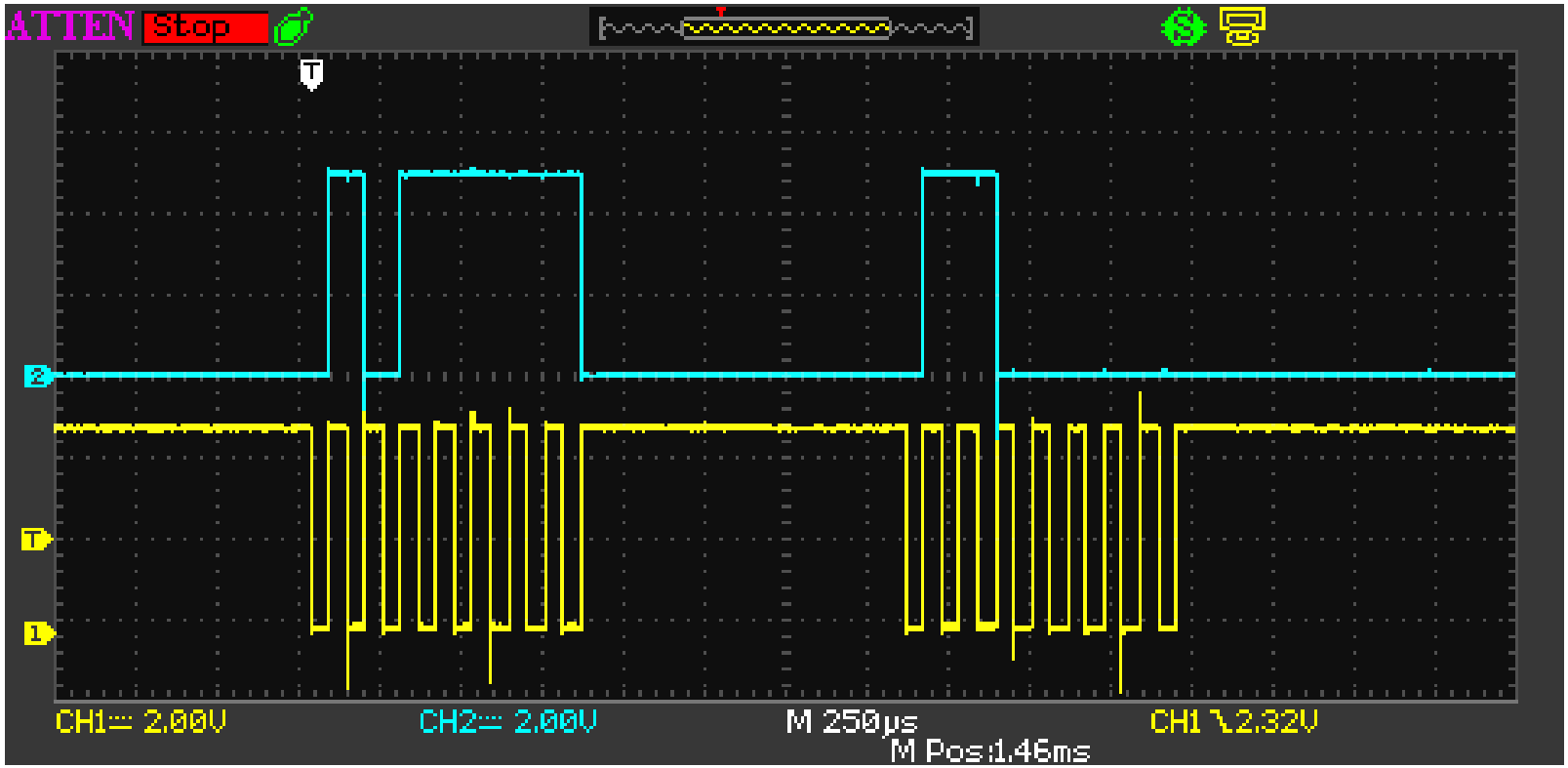


Figure 4: Frame

Standard data signals

- ~~UART TTL~~
- ~~I2C~~
- ~~RS232~~
- **SPI**
- ...

SPI

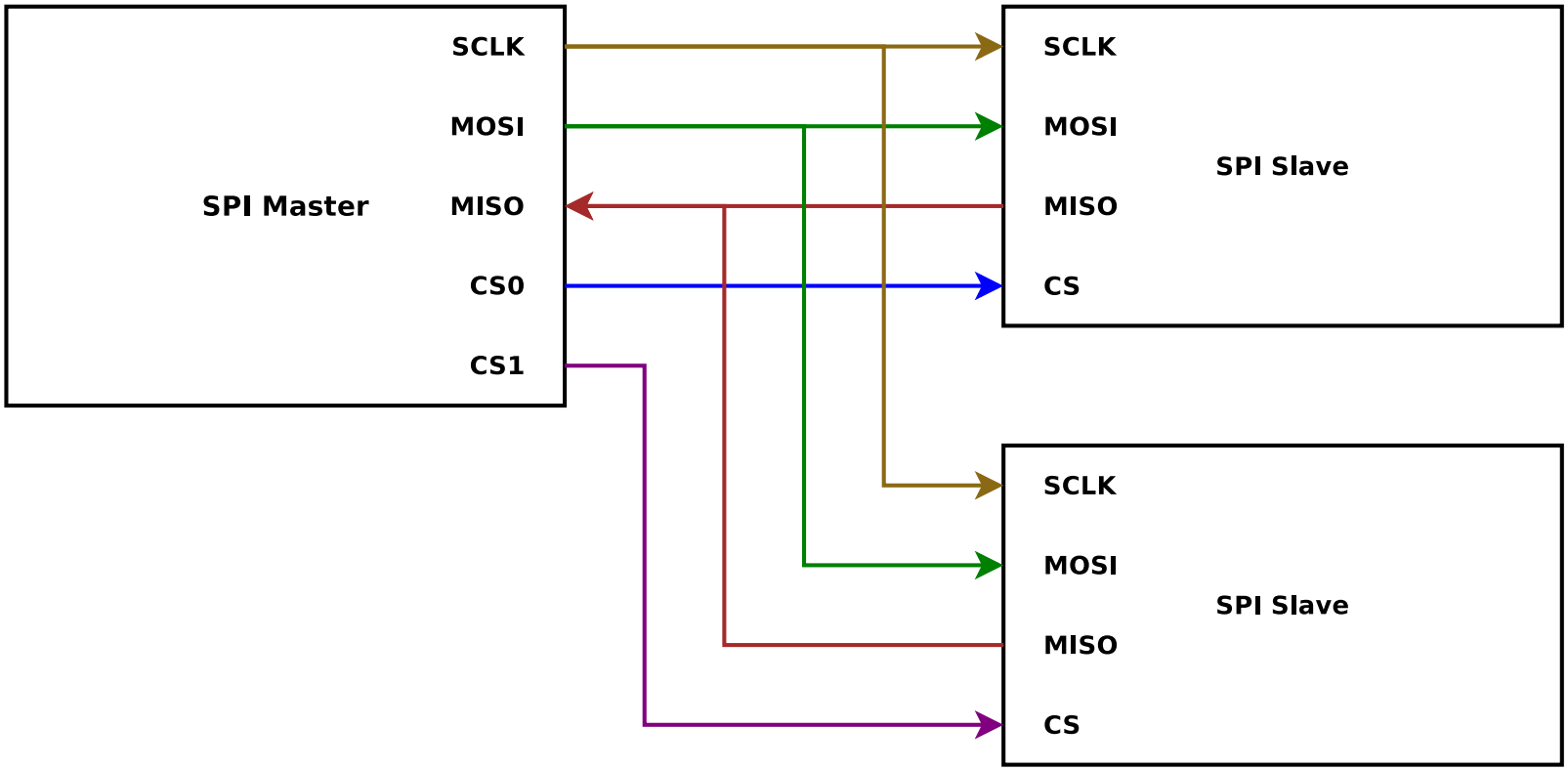


Figure 5: Frame

Chip Select

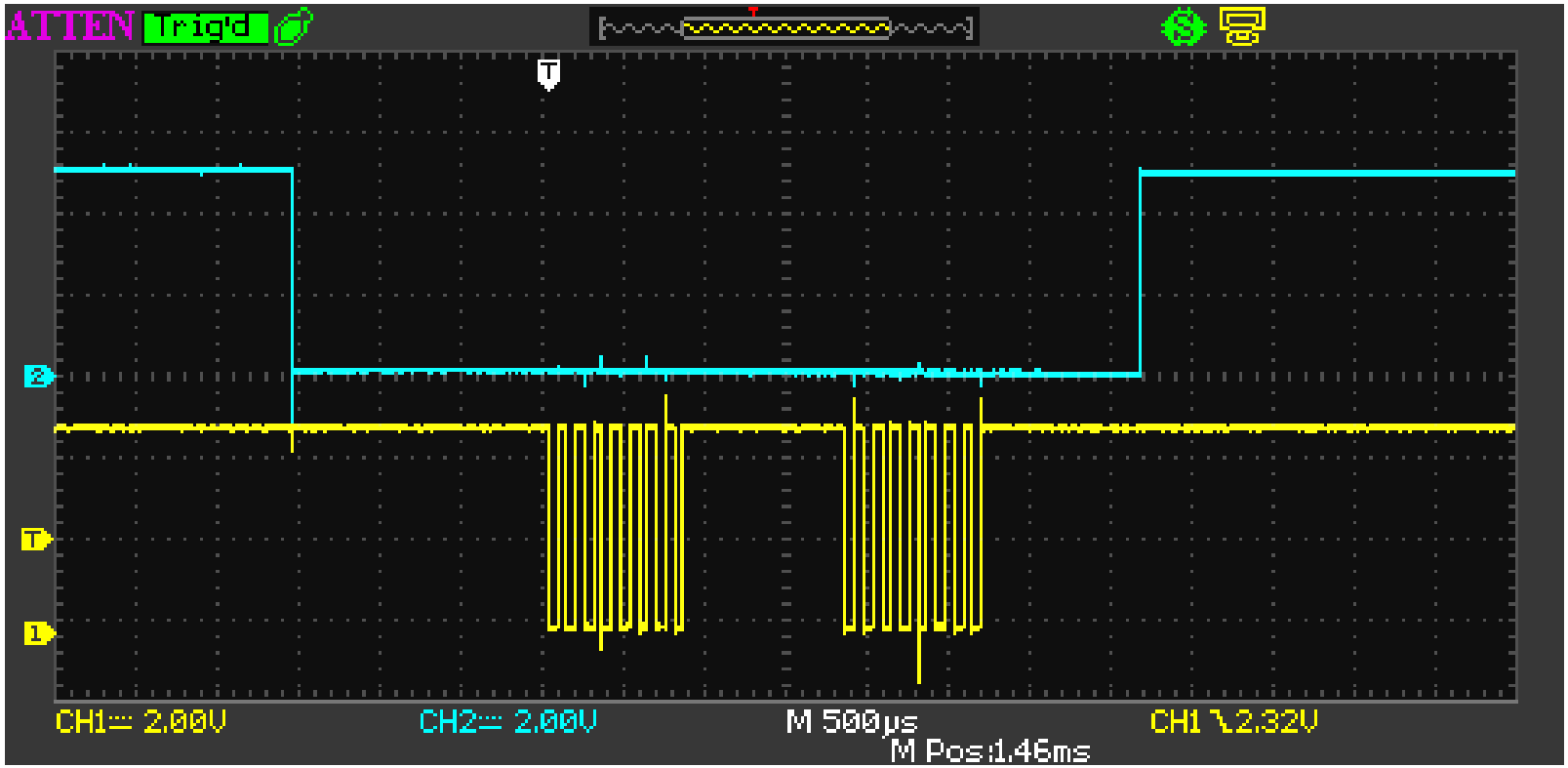


Figure 6: Frame

MOSI

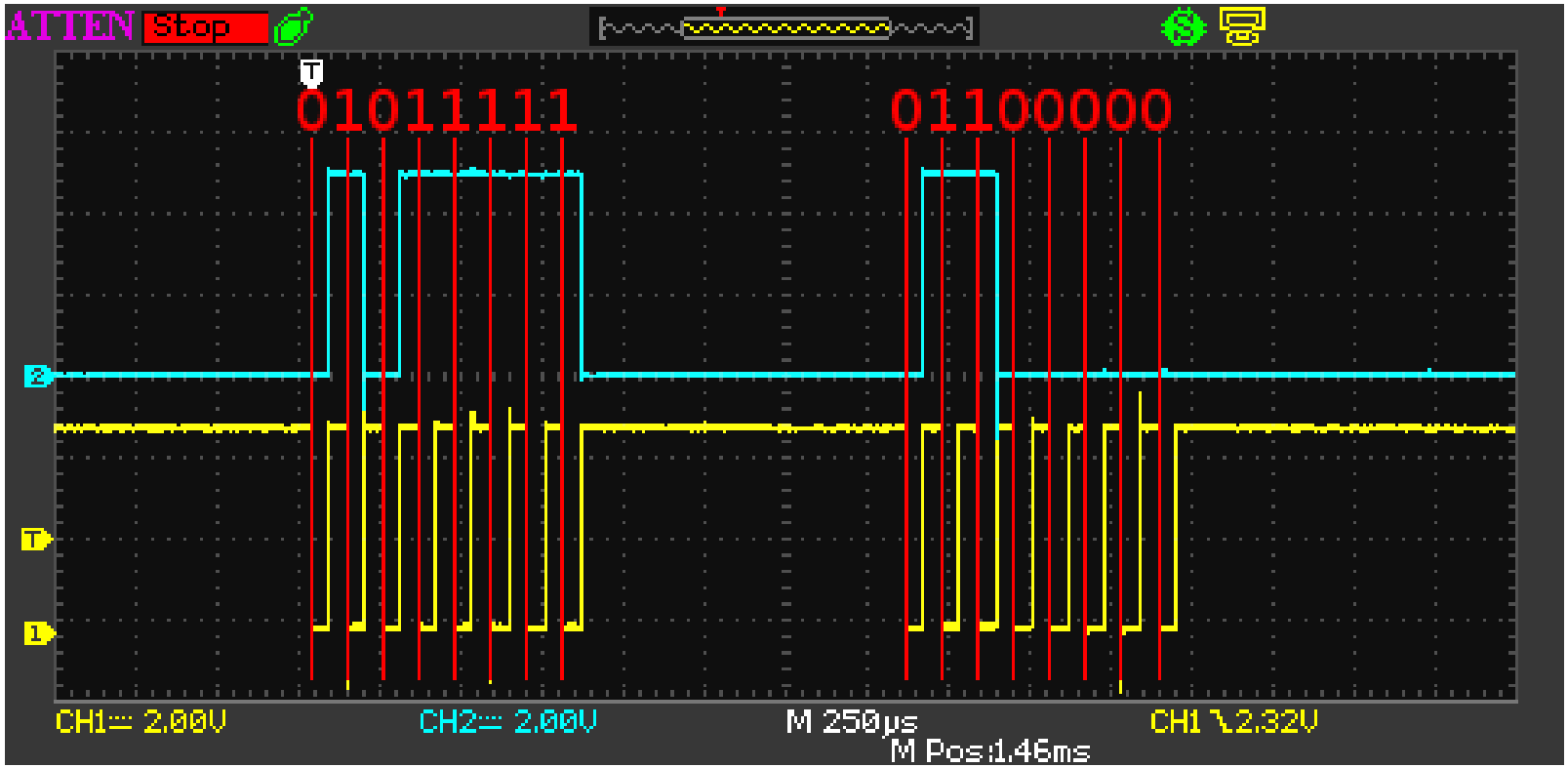


Figure 7: Frame

Pinout (SPI)

- 5V
- Ground
- MOSI
- MISO
- SS
- SCK

MISO

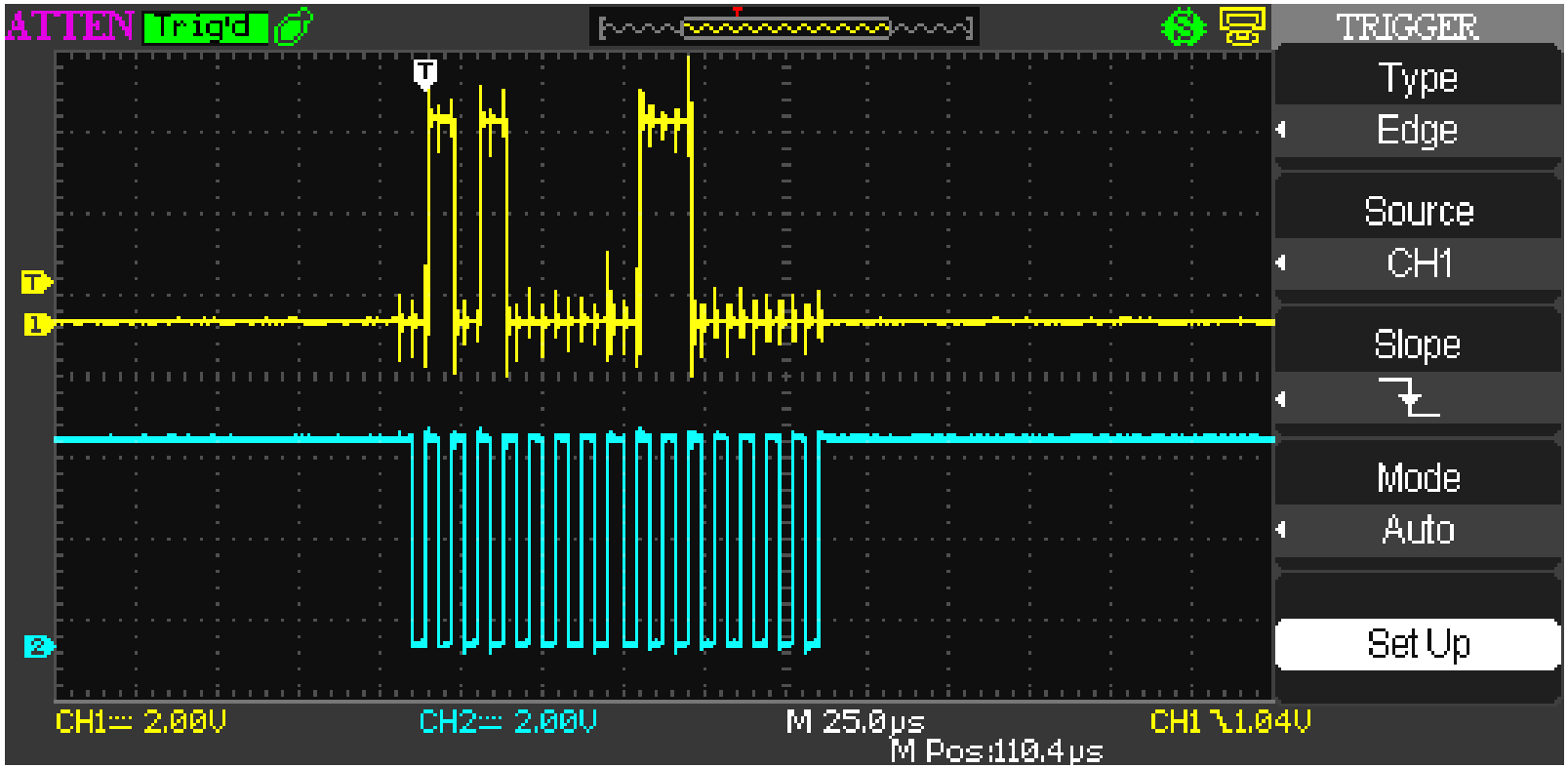


Figure 8: Frame

Probing

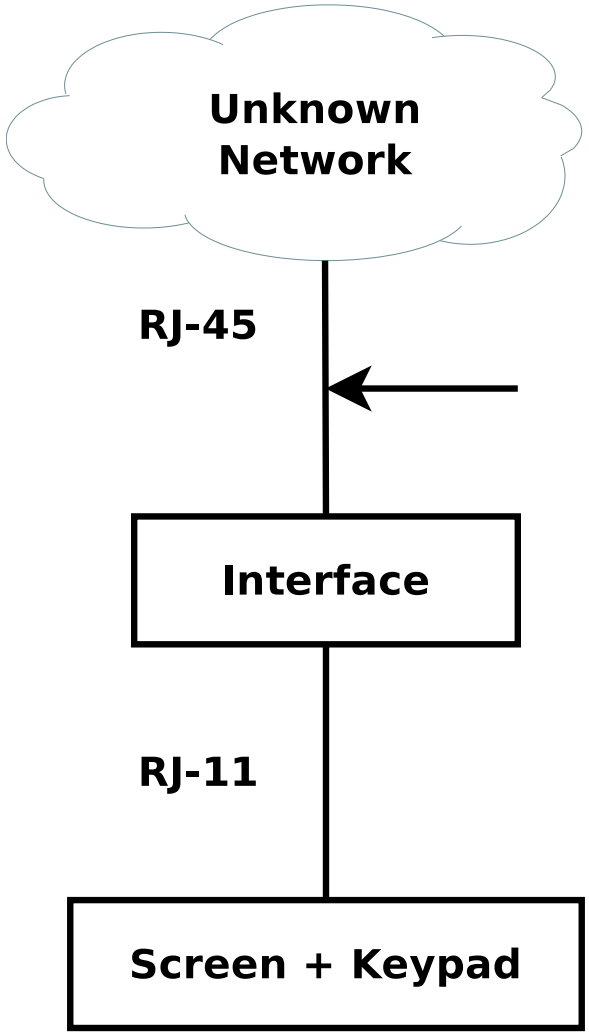


Figure 9: Probe points

The hanging connector

Figure 10: Ceiling cable

Probing

- Our goal: identify signals
- Possible signals:
  - Power: 12V
  - Ground: Yes
  - ~~Clock~~
  - ~~Data~~
  - Differential data: On two wires

Standard data signals

- ~~UART TTL~~
- ~~SPI~~
- ~~I2C~~
- ~~RS232~~
- ~~*Ethernet PHY*~~
- ~~CAN~~
- **RS485**

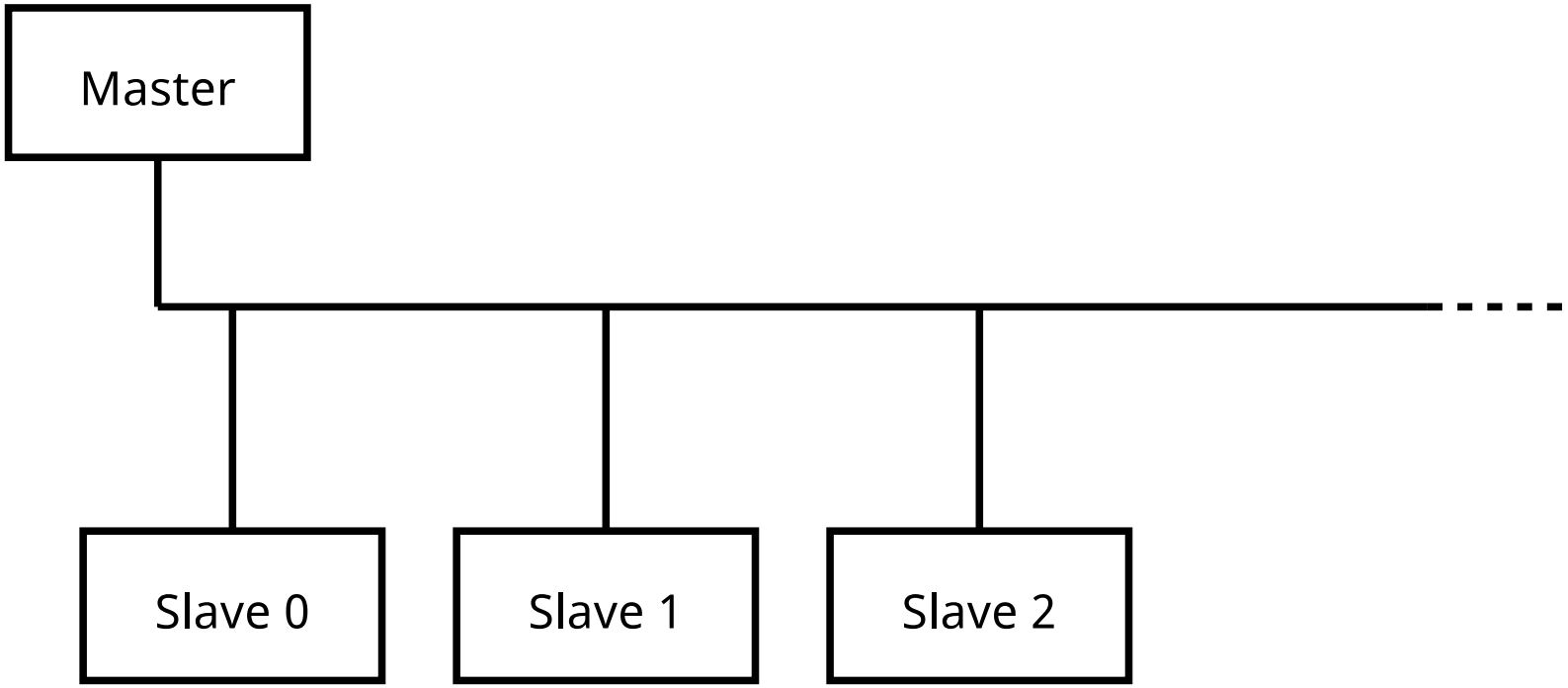
Standard data signals

- ~~UART TTL~~
- ~~SPI~~
- ~~I2C~~
- ~~RS232~~
- ~~*Ethernet PHY*~~
- ~~CAN~~
- **RS485**

RS485 in short

- Two wires: A and B
- Differential signal:
  - A - B <-200mV is 1
  - A - B >+200mV is 0

Multidrop RS485



- Same bus
- One master, many slaves
- Bidirectional communication:
  - Master polls slaves periodically
  - Slave answer when talked to

Frame

- No clock!

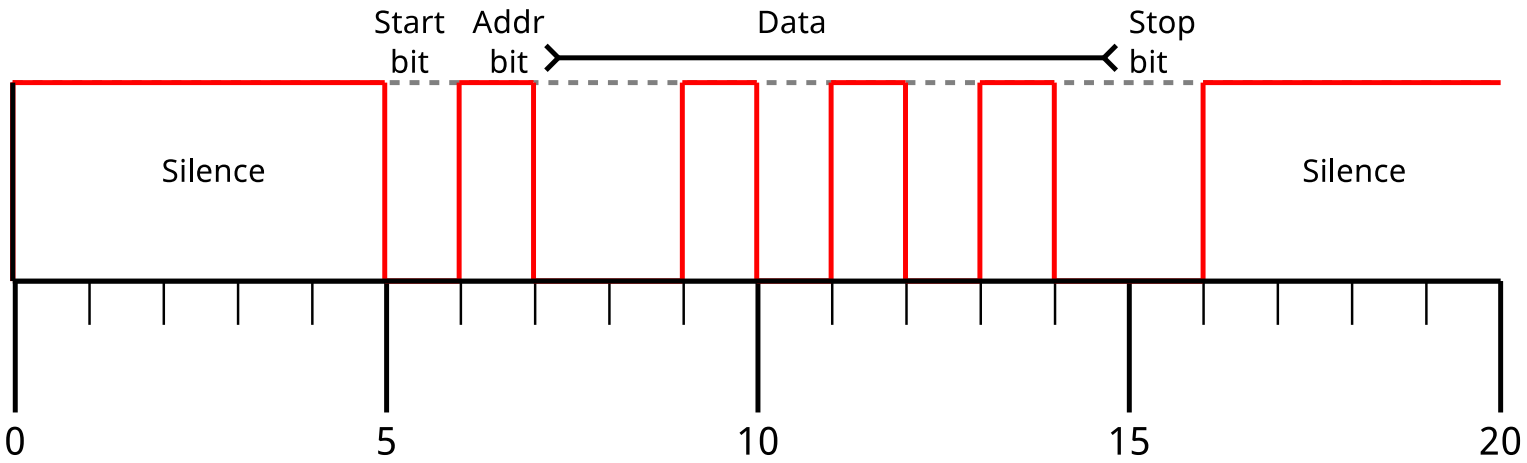


Figure 11: A word in this protocol

Principles of 9bit data mode

- 9th bit is used to signal an address
- The slave only listen when the address matches it's own

Messages

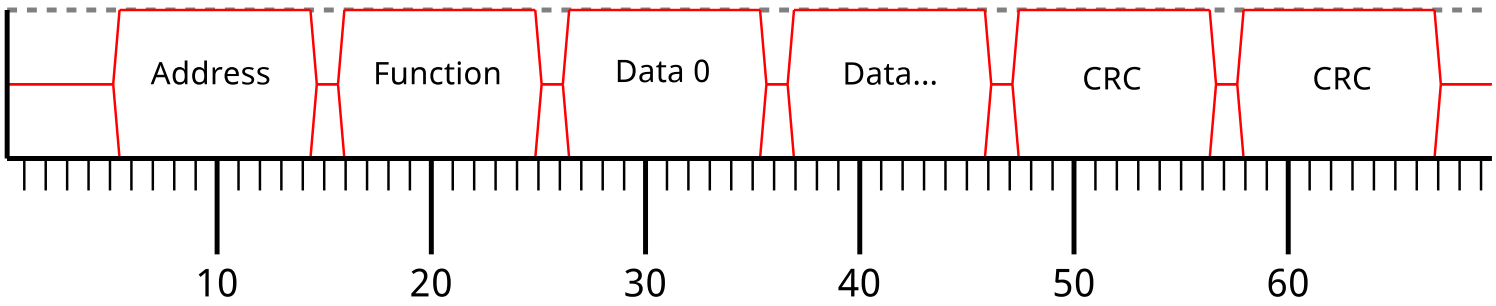


Figure 12: Message structure

Modbus

Modbus RTU

- Another serial communication protocol

Similarities

- Same CRC polynom
- Message format

Differences with modbus RTU

- Not the same function
- Use an address bit
- Broadcast address is `FF`

Probing: done!

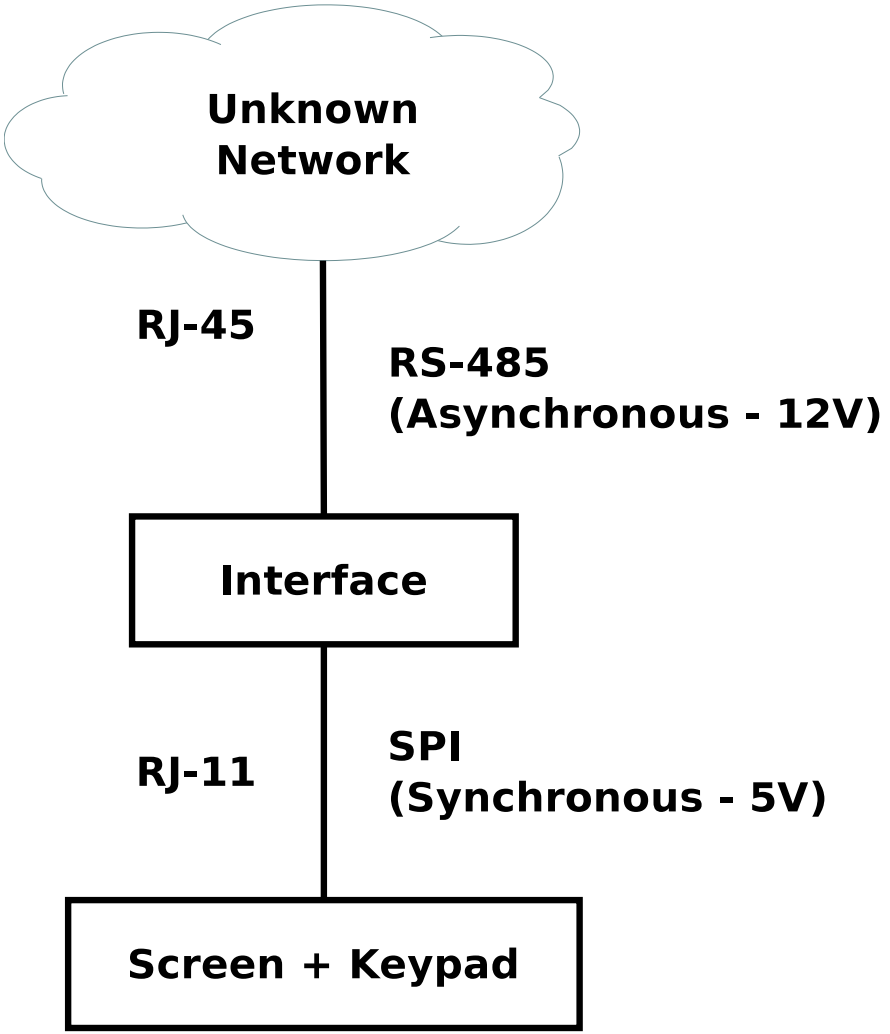
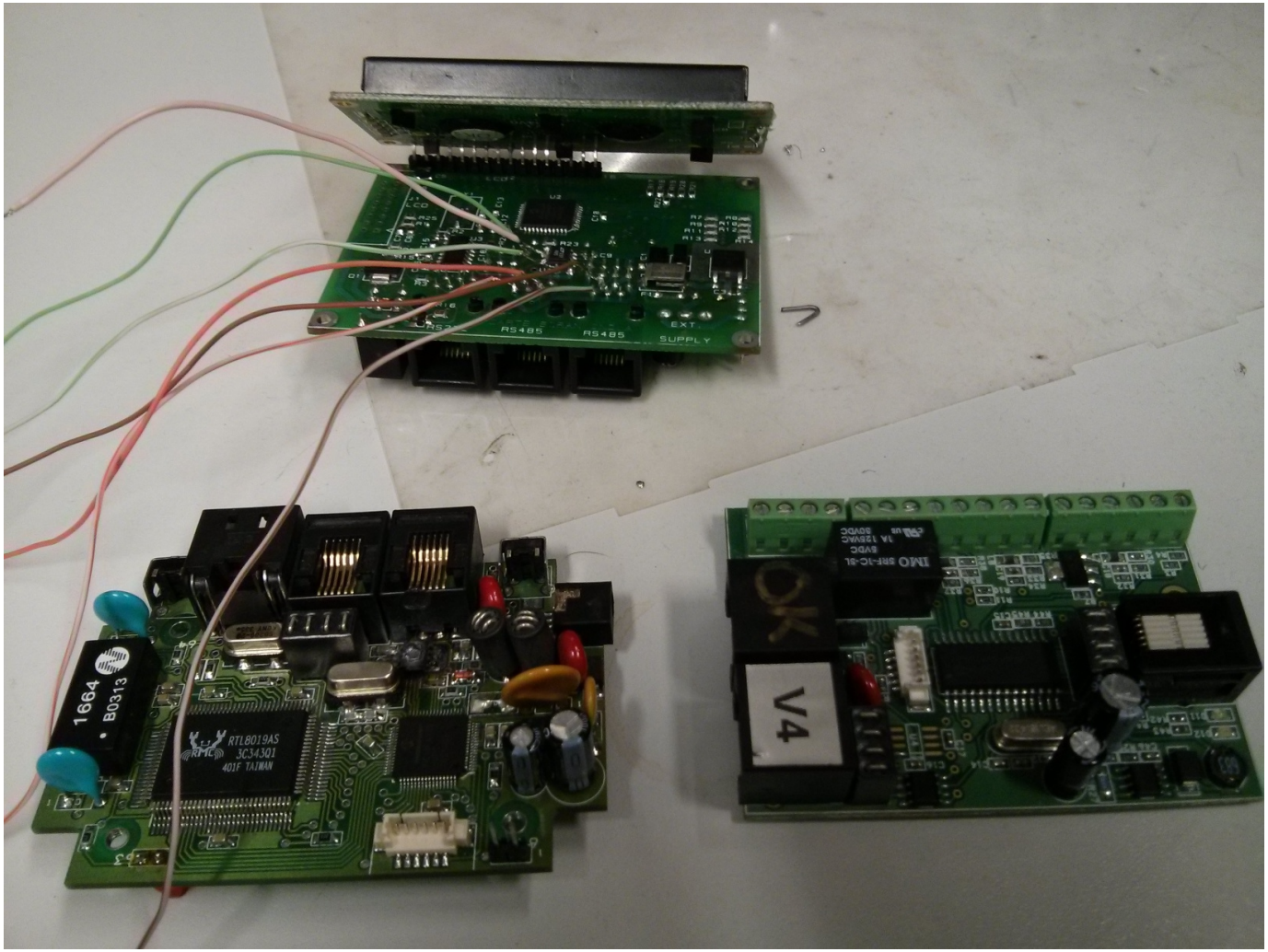


Figure 13: Probe points

Hardware

We have boards



- Identify the parts
- Dump what you can

Recognizing the parts

- "Passive" components:
  - Resistors
  - Capacitors
  - Inductors
- "Active" components:
  - Microcontrollers
  - PHY, signal converters

Components

Screen Board

- UART to RS232
- PIC16F877

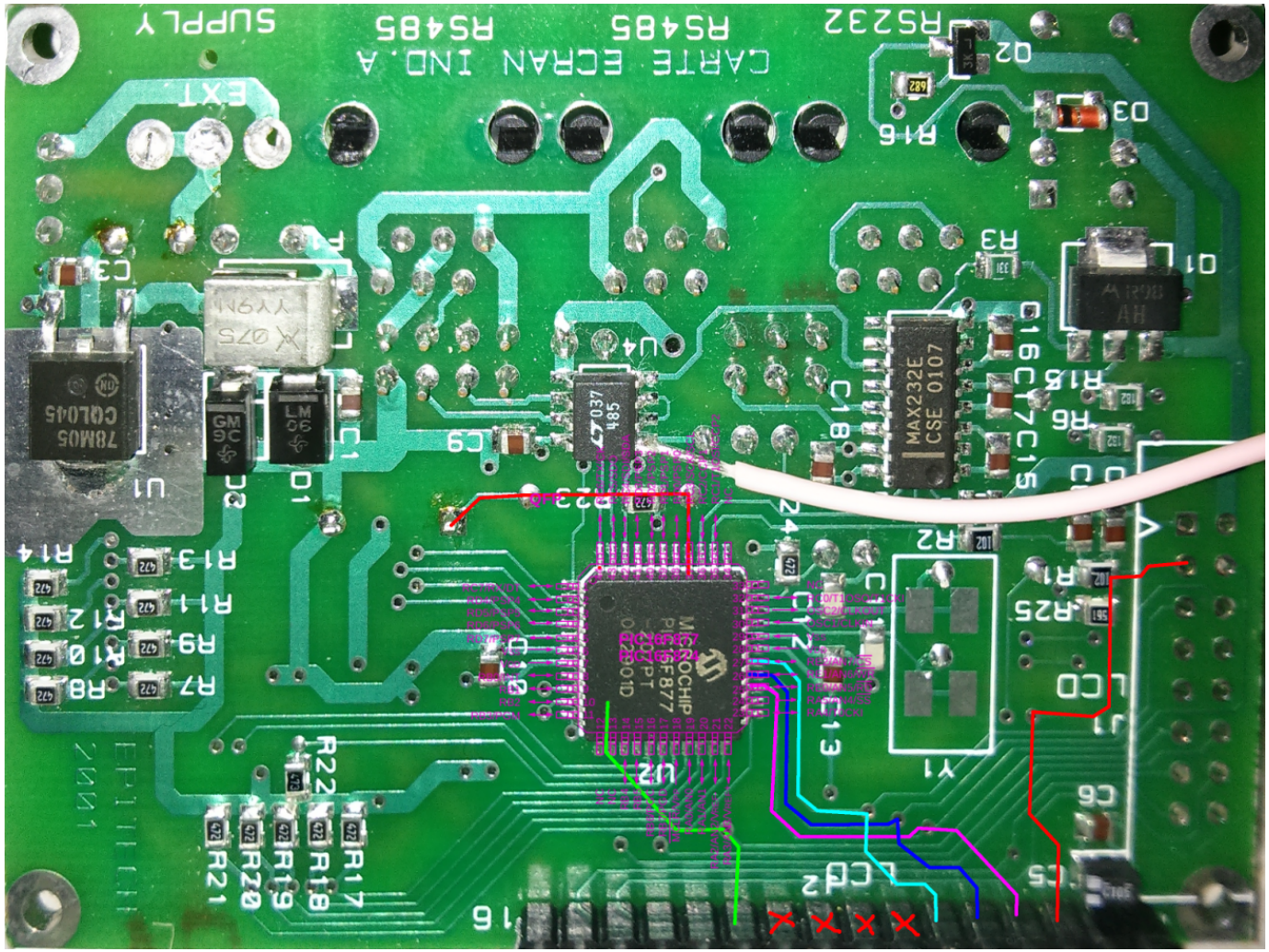
Secu Board

- UART to RS485
- Relay
- PIC18F6720

Unknown Board

- UART to RS485
- Ethernet PHY
- PIC18F2480

The board we worked on



Software

Microcontroller: PIC16F877

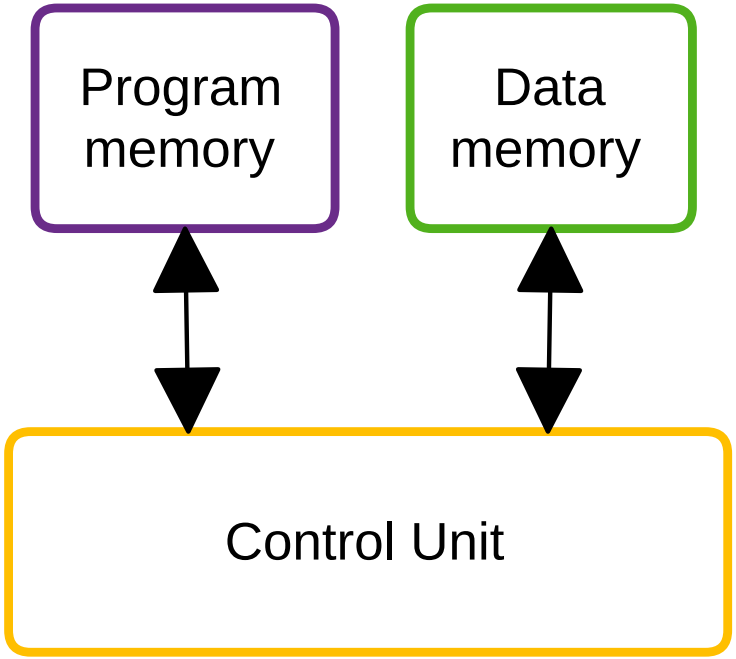
- A very common 8bit µC.
- RISC: 35 instructions
- 8K Flash program memory
- 368 bytes of RAM
- 256 bytes of EEPROM
- Program instruction bus: 14bits
- Program counter: 13bits
- Data bus: 8bits

Dumping the flash

- Code protection: No!

Harvard computer architecture

- Code and data are stored in different memories

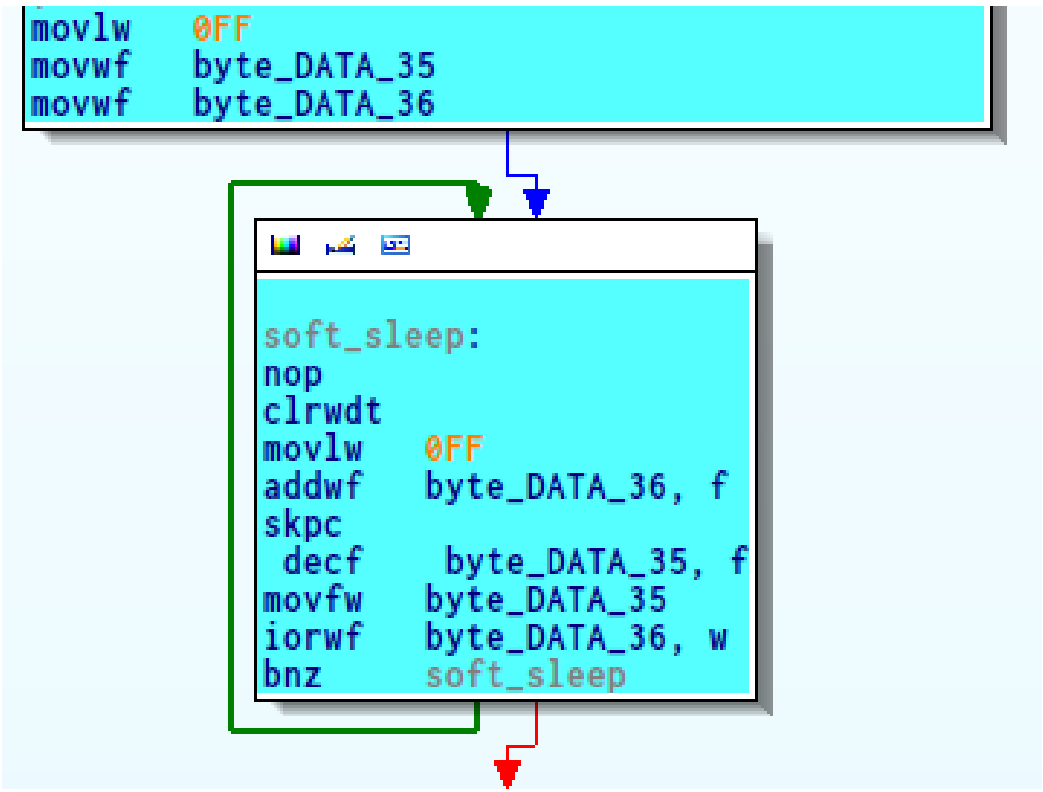


Harvard

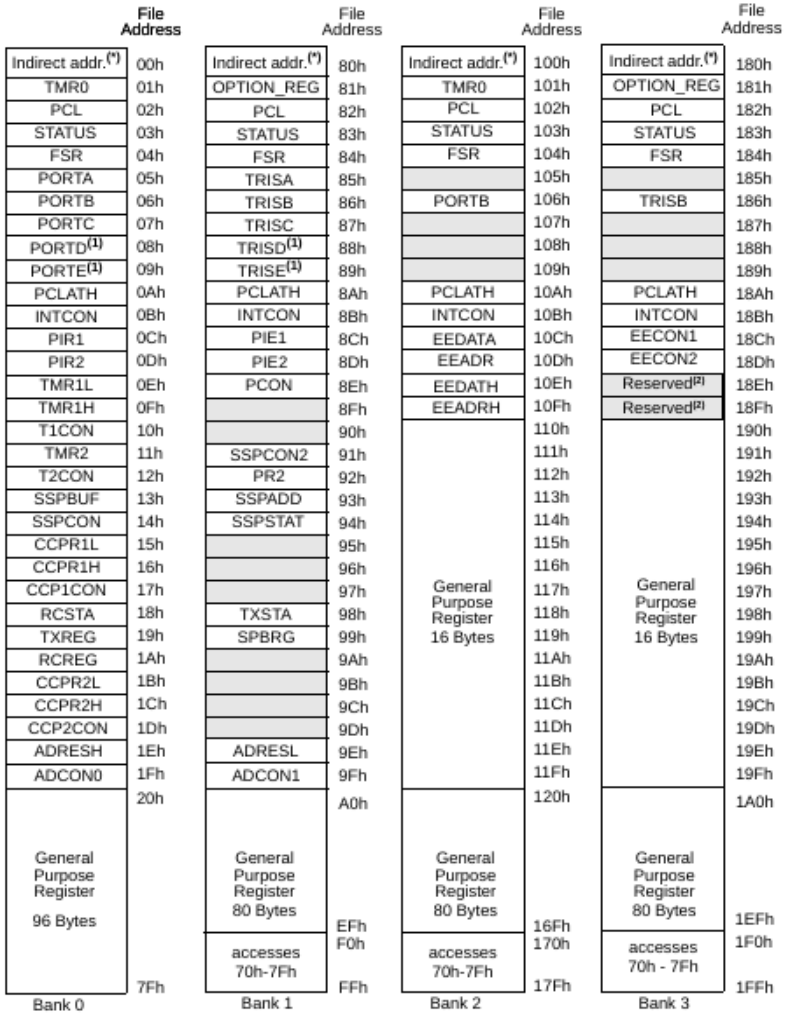
PIC16F87: reverse engineering hints

- Software sleeps
- Banking systems
- Indirect read/writes
- PIC's version of progmem/progspace

PIC16F87: Sleeping

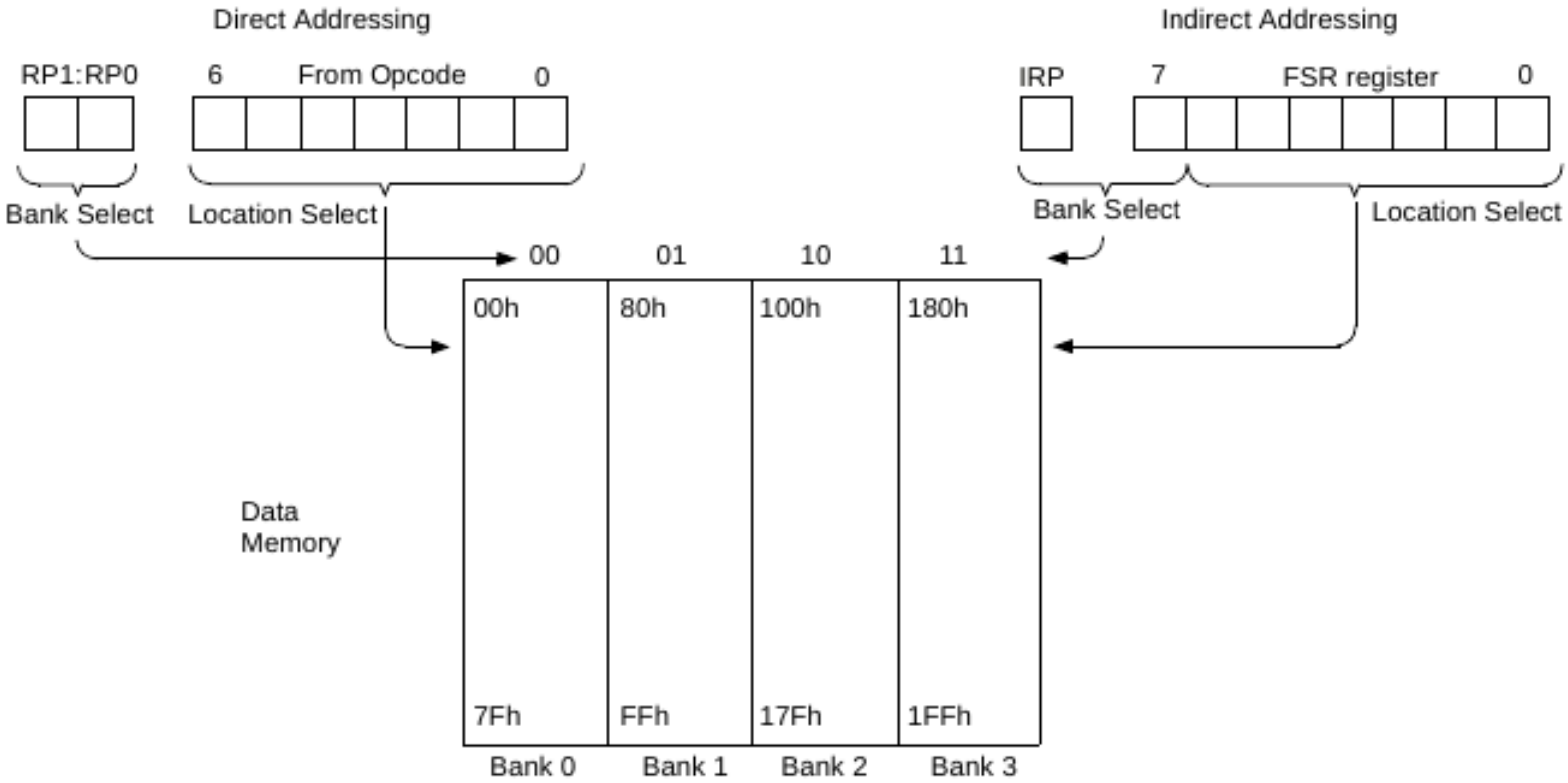


PIC16F87: Memory Banks

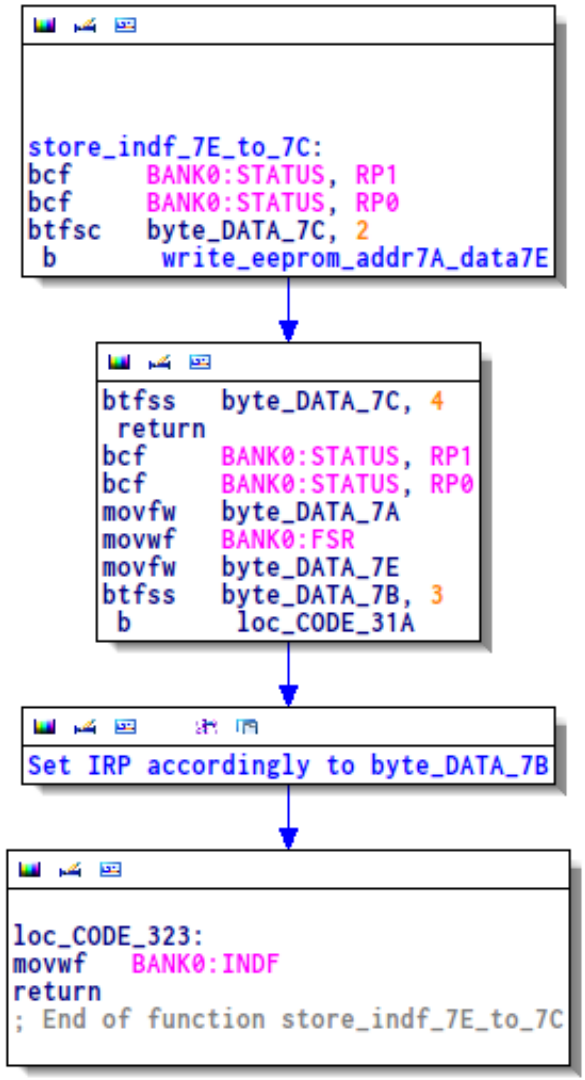


Bank 0	File Address	Bank 1	File Address	Bank 2	File Address	Bank 3	File Address
Indirect addr.(*)	00h	Indirect addr.(*)	80h	Indirect addr.(*)	100h	Indirect addr.(*)	180h
TMR0	01h	OPTION_REG	81h	TMR0	101h	OPTION_REG	181h
PCL	02h	PCL	82h	PCL	102h	PCL	182h
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h	FSR	84h	FSR	104h	FSR	184h
PORTA	05h	TRISA	85h		105h		185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
PORTC	07h	TRISC	87h		107h		187h
PORTD(1)	08h	TRISD(1)	88h		108h		188h
PORTE(1)	09h	TRISE(1)	89h		109h		189h
PCLATH	0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah
INTCON	0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch	EEDATA	10Ch	EECON1	18Ch
PIR2	0Dh	PIE2	8Dh	EEADR	10Dh	EECON2	18Dh
TMR1L	0Eh	PCON	8Eh	EEDATH	10Eh	Reserved(2)	18Eh
TMR1H	0Fh		8Fh	EEADRH	10Fh	Reserved(2)	18Fh
T1CON	10h		90h		110h		190h
TMR2	11h	SSPCON2	91h		111h		191h
T2CON	12h	PR2	92h		112h		192h
SSPBUF	13h	SSPADD	93h		113h		193h
SSPCON	14h	SSPSTAT	94h		114h		194h
CCPR1L	15h		95h		115h		195h
CCPR1H	16h		96h		116h		196h
CCP1CON	17h		97h	General Purpose Register 16 Bytes	117h	General Purpose Register 16 Bytes	197h
RCSTA	18h	TXSTA	98h		118h		198h
TXREG	19h	SPBRG	99h		119h		199h
RCREG	1Ah		9Ah		11Ah		19Ah
CCPR2L	1Bh		9Bh		11Bh		19Bh
CCPR2H	1Ch		9Ch		11Ch		19Ch
CCP2CON	1Dh		9Dh		11Dh		19Dh
ADRESH	1Eh	ADRESL	9Eh		11Eh		19Eh
ADCON0	1Fh	ADCON1	9Fh		11Fh		19Fh
General Purpose Register 96 Bytes	20h	General Purpose Register 80 Bytes	A0h	General Purpose Register 80 Bytes	120h	General Purpose Register 80 Bytes	1A0h
	7Fh	accesses 70h-7Fh	EFh F0h FFh	accesses 70h-7Fh	16Fh 170h 17Fh	accesses 70h - 7Fh	1EFh 1F0h 1FFh

PIC16F87: Memory Banking



PIC16F87: Indirect read/write macros



PIC16F87: Storing data in program space

```
CODE:1598 s_access_opened:
CODE:1598                 retlw   41 ; 'A'
CODE:1599 ; ---------------------------
CODE:1599                 retlw   63 ; 'c'
CODE:159A ; ---------------------------
CODE:159A                 retlw   63 ; 'c'
CODE:159B ; ---------------------------
CODE:159B                 retlw   65 ; 'e'
CODE:159C ; ---------------------------
CODE:159C                 retlw   73 ; 's'
CODE:159D ; ---------------------------
CODE:159D                 retlw   73 ; 's'
CODE:159E ; ---------------------------
CODE:159E                 retlw   20 ; ' '
CODE:159F ; ---------------------------
CODE:159F                 retlw   4F ; 'O'
CODE:15A0 ; ---------------------------
CODE:15A0                 retlw   70 ; 'p'
CODE:15A1 ; ---------------------------
CODE:15A1                 retlw   65 ; 'e'
CODE:15A2 ; ---------------------------
CODE:15A2                 retlw   6E ; 'n'
CODE:15A3 ; ---------------------------
CODE:15A3                 retlw   65 ; 'e'
CODE:15A4 ; ---------------------------
CODE:15A4                 retlw   64 ; 'd'
```

PIC16F87: Storing data in program space


```
; assume pclath = 0


; goto $7b$7a

lookup_table_get:
bcf     BANK0:STATUS, RP1 ; reset bank
bcf     BANK0:STATUS, RP0 ; reset bank
movfw   byte_DATA_7B      ; bank0:7b == 16
movwf   BANK0:PCLATH      ; pclath <- 16
movfw   byte_DATA_7A      ; bank0:7a == 18
movwf   BANK0:PCL         ; pcl <- 18
; End of function lookup_table_get ; GOTO 0x1618
```

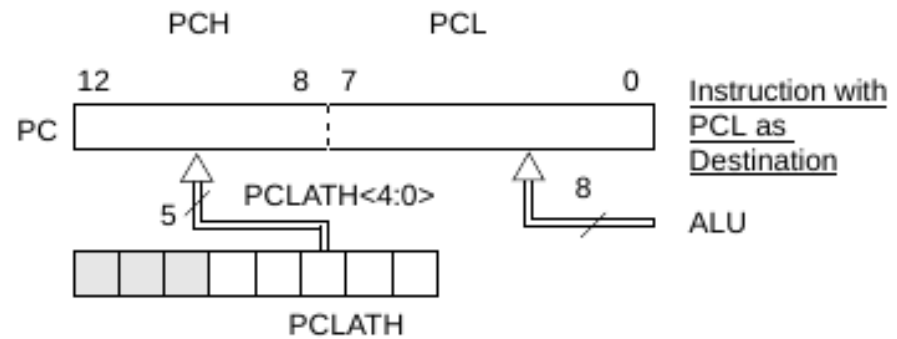


Figure 14: PIC paging

In situ

The situation

- The door lock is broken: fix-it!
- Power is working, the door is always locked
- Control is not working

Topology

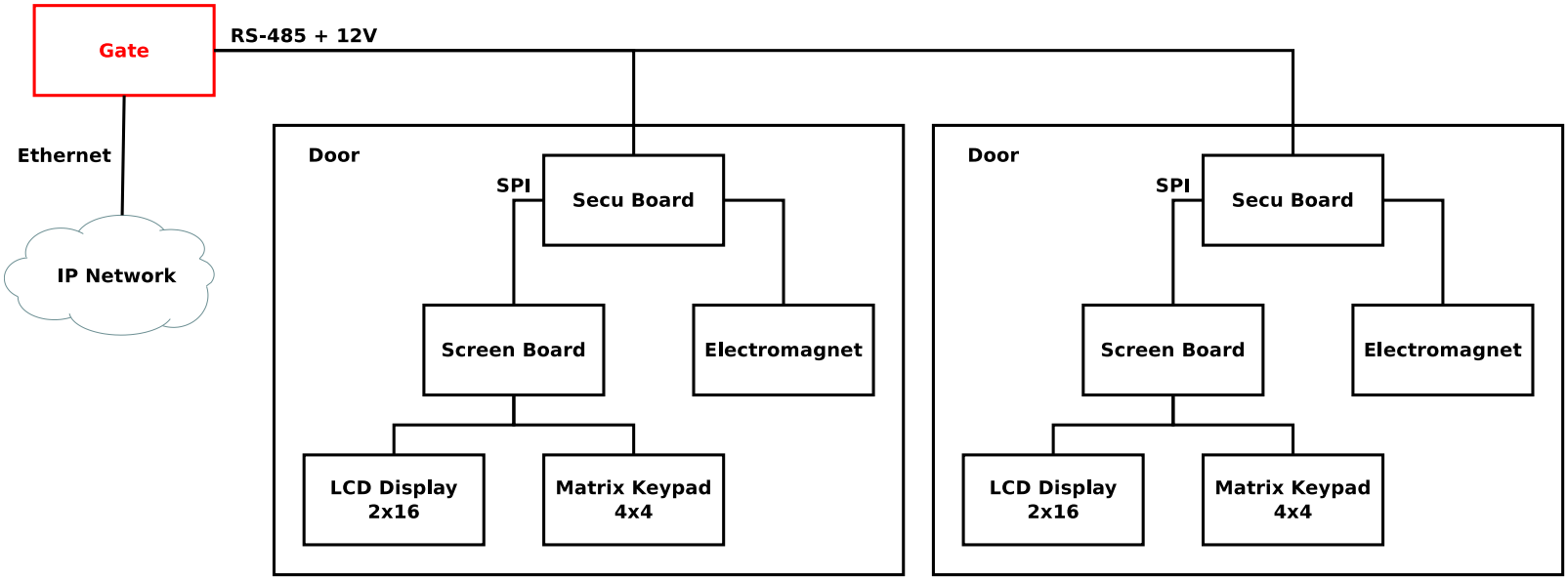


Figure 15: Topology

Hook

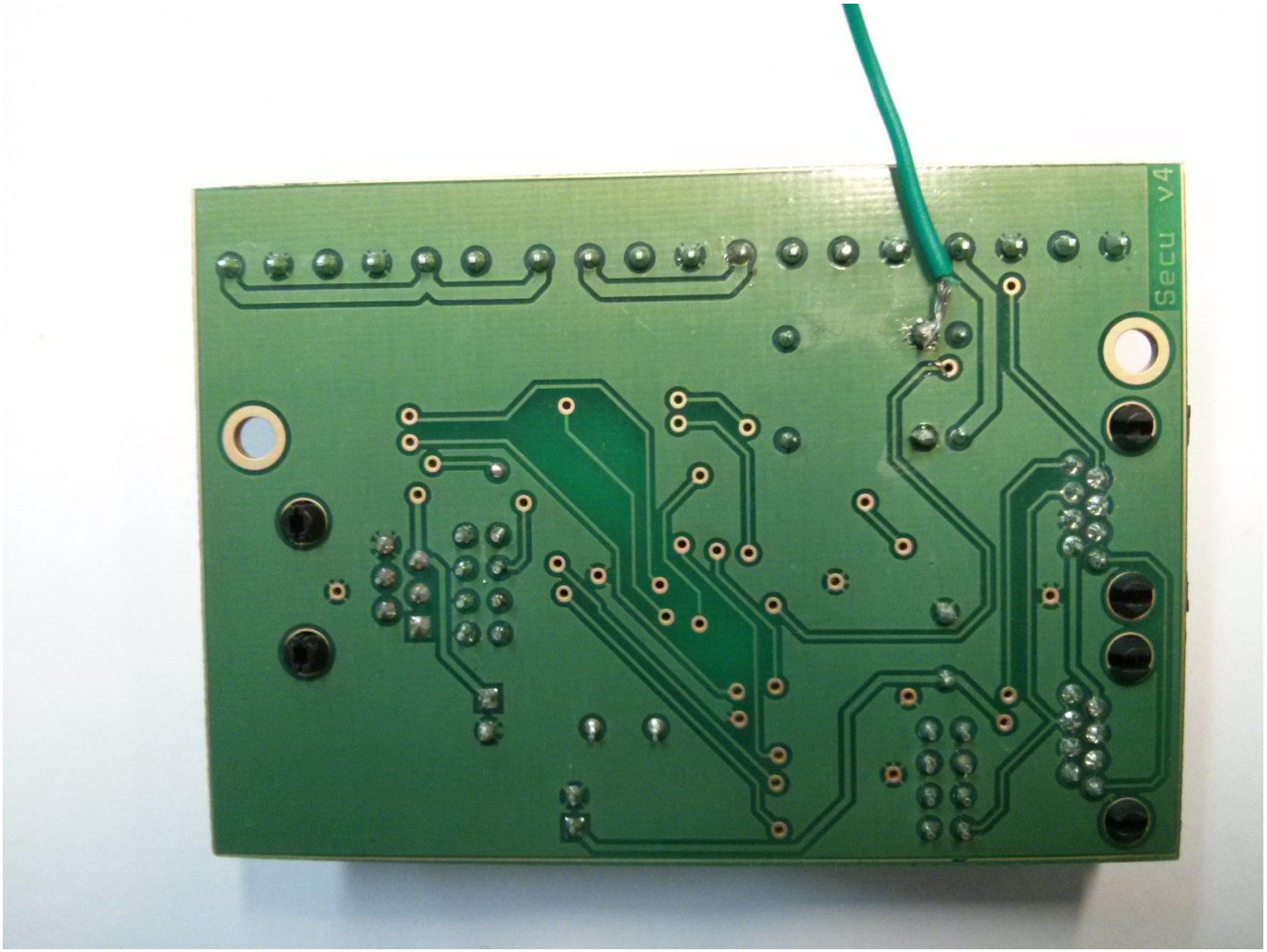


Figure 16: Our wire

Topology

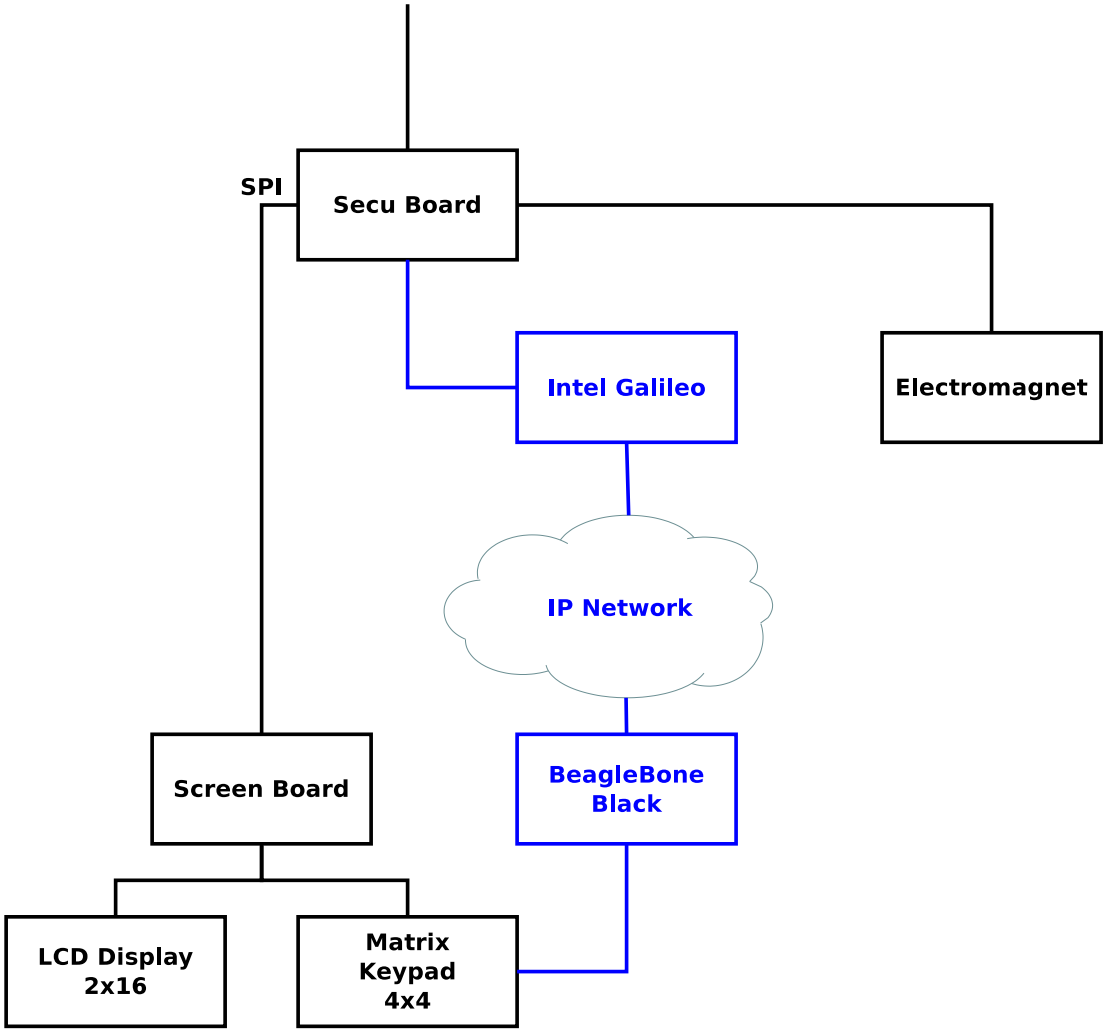


Figure 17: Topology

Conclusion

Our analysis

- This system is too rigid: custom hardware, half-duplex
- Not resilient too power failure: every thing is online, multiple SPOF
- No security: clear text communication

Going further

Misc. good reads

- http://www.bunniestudios.com
- http://www.spritesmods.com
- http://www.devttys0.com

Hardware hacking exercices

- http://blog.scrt.ch/2013/03/26/insomnihack-2013-life-is-hardware/
- http://www.balda.ch/posts/2014/Apr/01/ins14-life-is-even-harder/
- https://microcorruption.com

Thanks

- Pierre Bourdon
- Théo
- Christian Dujardin
- Evolutek<<
- Prologin

Conclusion

Contact

- Rémi 'halfr' Audebert
  - IRC: `halfr@irc.rezosup.org`
  - Mail: `halfr@lse.epita.fr`
  - Twitter: `@halfr`
- Pierre 'Ptishell' Surply
  - IRC: `Ptishell@irc.rezosup.org`
  - Mail: `surply@lse.epita.fr`
  - Twitter: `@Ptishell`