OVERMEDICATED

BREAKING THE SECURITY BARRIER OF A B.BRAUN INFUSION PUMP

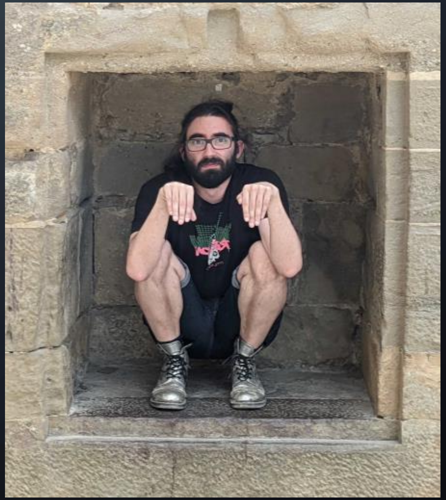
Douglas McKee

Philippe Laulheret

WHO ARE WE?

- Douglas McKee
  - Principal Engineer & Sr Security Researcher for McAfee's Enterprise Advanced Threat Research team
  - 12+ years experience in vulnerability research, penetration testing and forensics
  - @fulmetalpackets

- Philippe Laulheret
  - Senior Security Researcher for McAfee's Enterprise Advanced Threat Research team
  - 10+ years in hacking the planet: C/C++ dev, CTFs, Embedded Security, Reverse Engineering, Hardware Hacking, Vulnerability Research, etc.
  - @phLaul



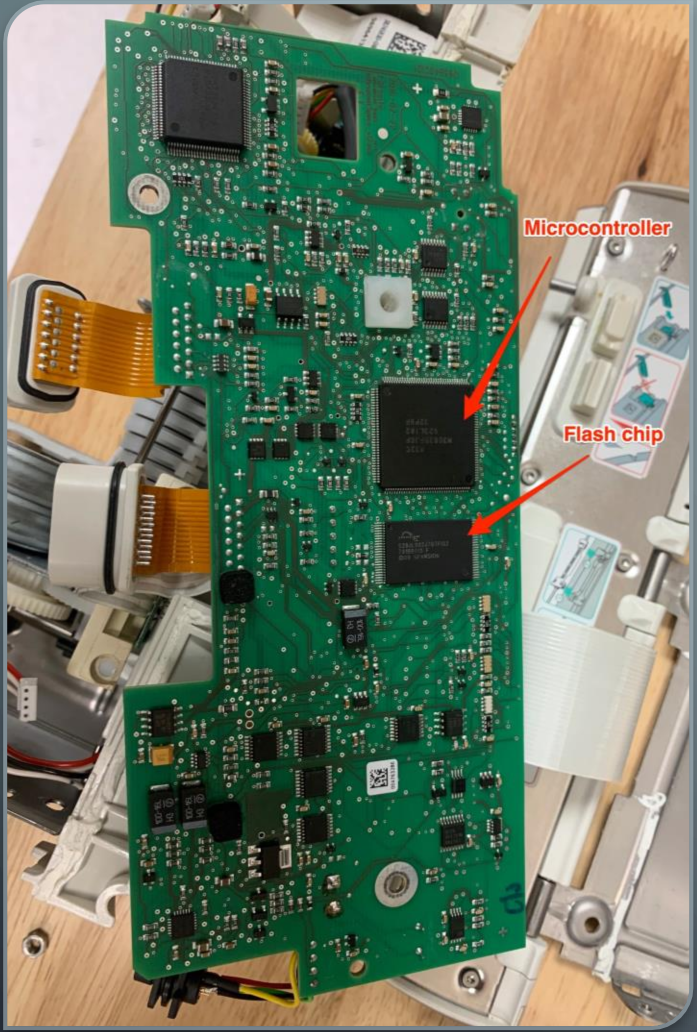
B. BRAUN INFUSOMAT

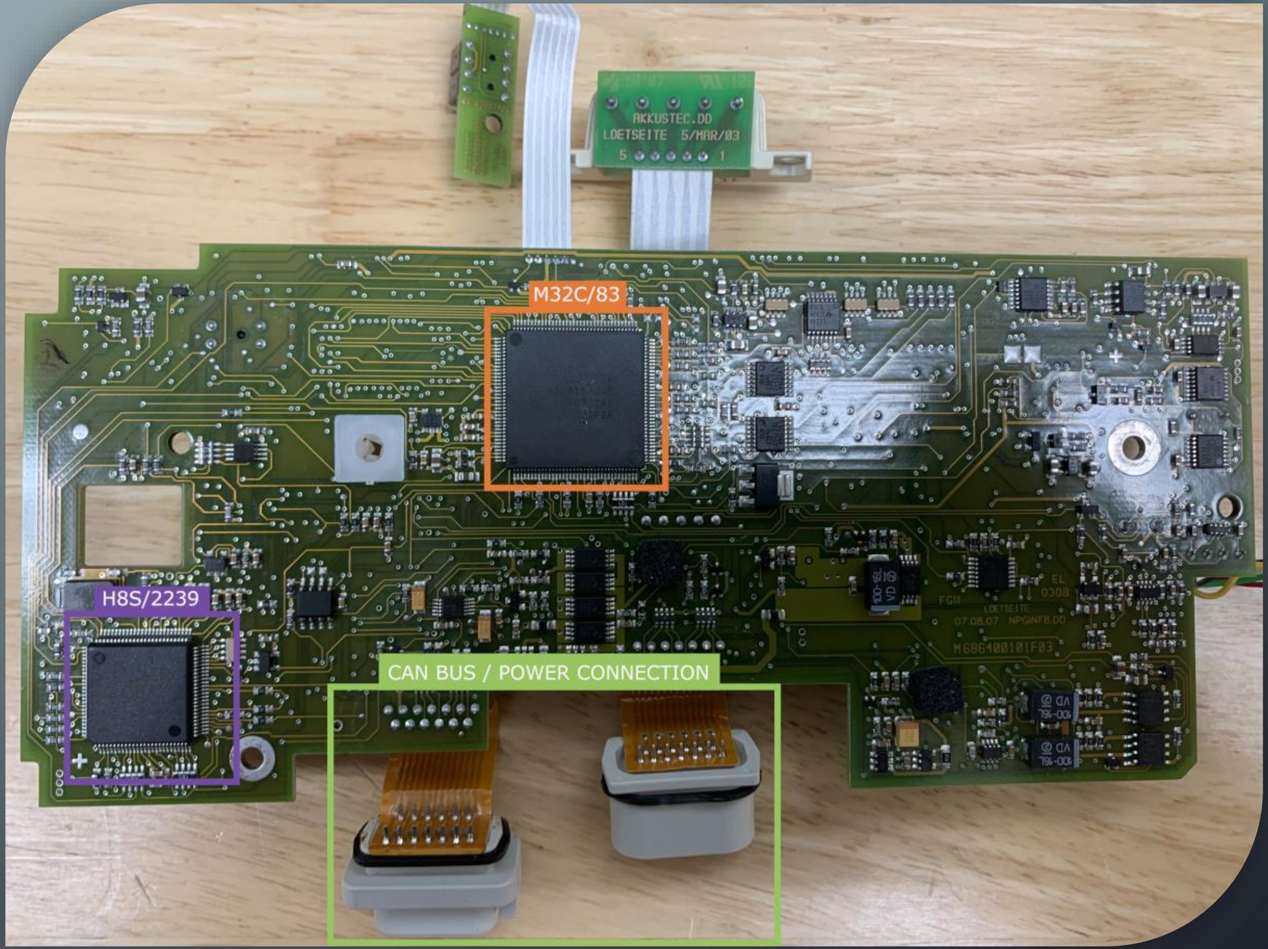
- B. Braun Infusomat Large Volume Pump Model 871305U
- SpaceStation Model 8713142U
- Running - SpaceCom
- Released in 2017
- "An infusion pump is a medical device that delivers fluids, such as nutrients and medications, into a patient's body in controlled amounts." - FDA

PAST RESEARCH

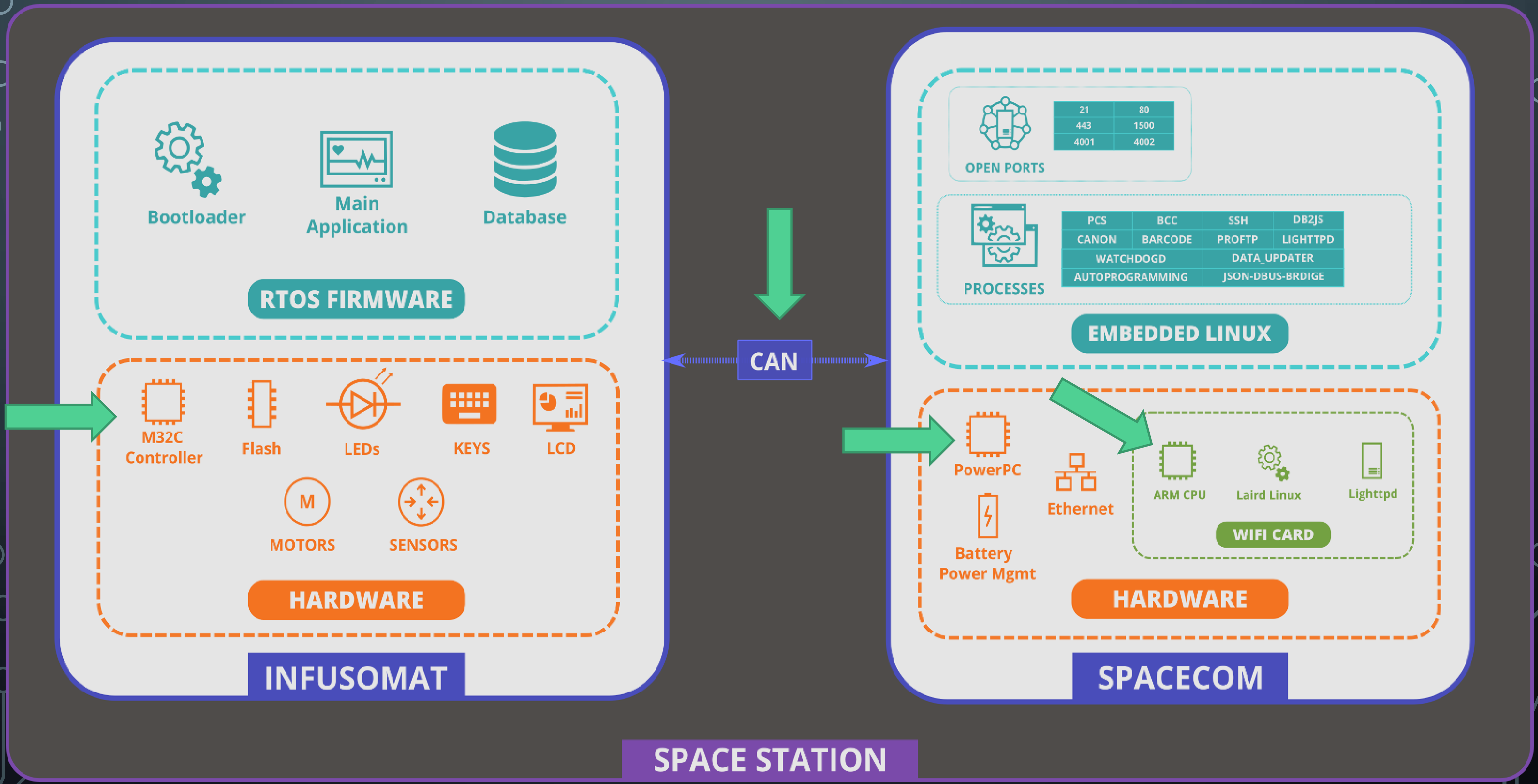
- ManiMed (2020)
  - Overview of medical device security ordered by German BSI
- Pacemaker + Insulin pump hack (Billy Rios + Jonathan Butts @ Blackhat 2018)
  - Announcement of FDA's CYMSAB
  - CYMSAB = CyberMed Safety (Expert) Analysis Board

SYSTEM ARCHITECTURE


Microcontroller
Flash chip


M32C/83
H8S/2239
CAN BUS / POWER CONNECTION

SYSTEM ARCHITECTURE

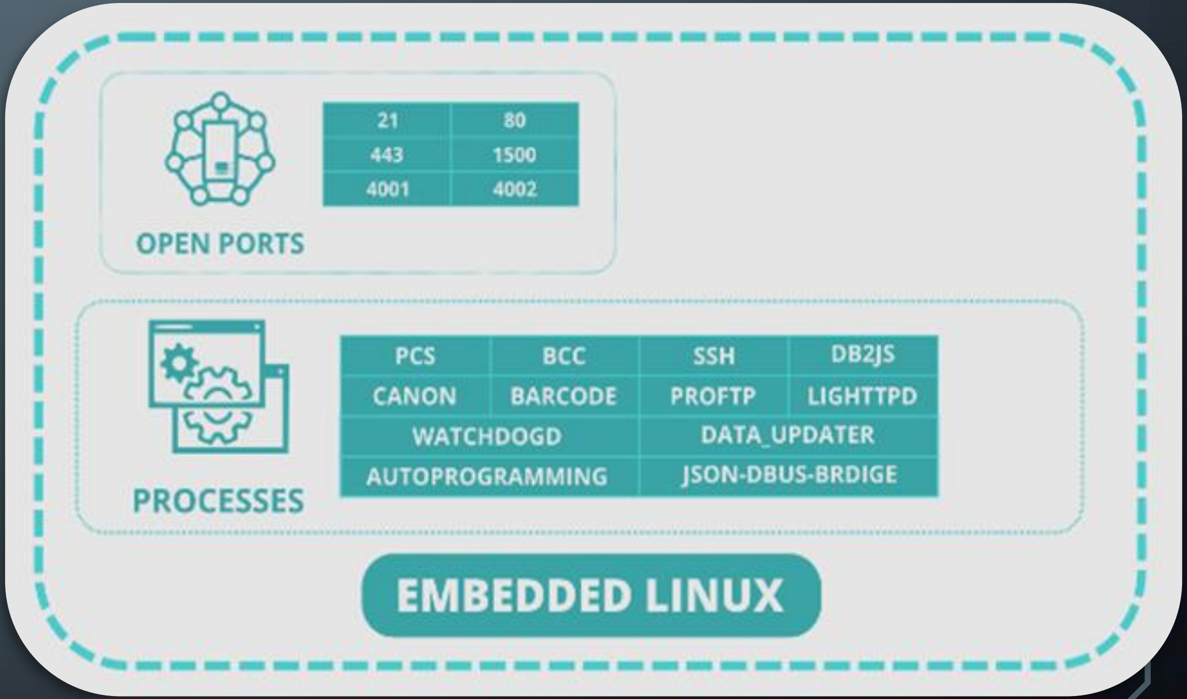


21	80
443	1500
4001	4002

PCS	BCC	SSH	DB2JS
CANON	BARCODE	PROFTP	LIGHTTPD
WATCHDOGD		DATA_UPDATER	
AUTOPROGRAMMING		JSON-DBUS-BRDIGE	

IMPORTANT APPLICATIONS

- Json-dbus-bridge
  - Listens on 80/443 for GET/POST requests
  - Used for Wi-Fi configuration
  - Open source!
- PCS
  - Processes commands from the management software
  - Listens on port 1500 for proprietary protocol/commands
  - Commands sent in cleartext
  - Updates drug library, calibration data and pump settings


OPEN PORTS

21	80
443	1500
4001	4002

PROCESSES

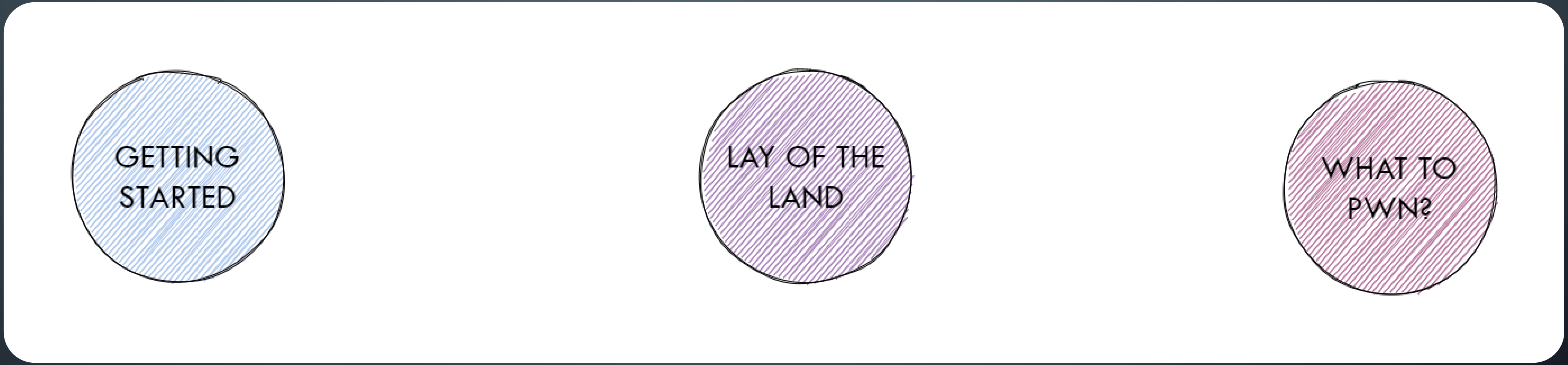
PCS	BCC	SSH	DB2JS
CANON	BARCODE	PROFTP	LIGHTTPD
WATCHDOGD		DATA_UPDATER	
AUTOPROGRAMMING		JSON-DBUS-BRDIGE	

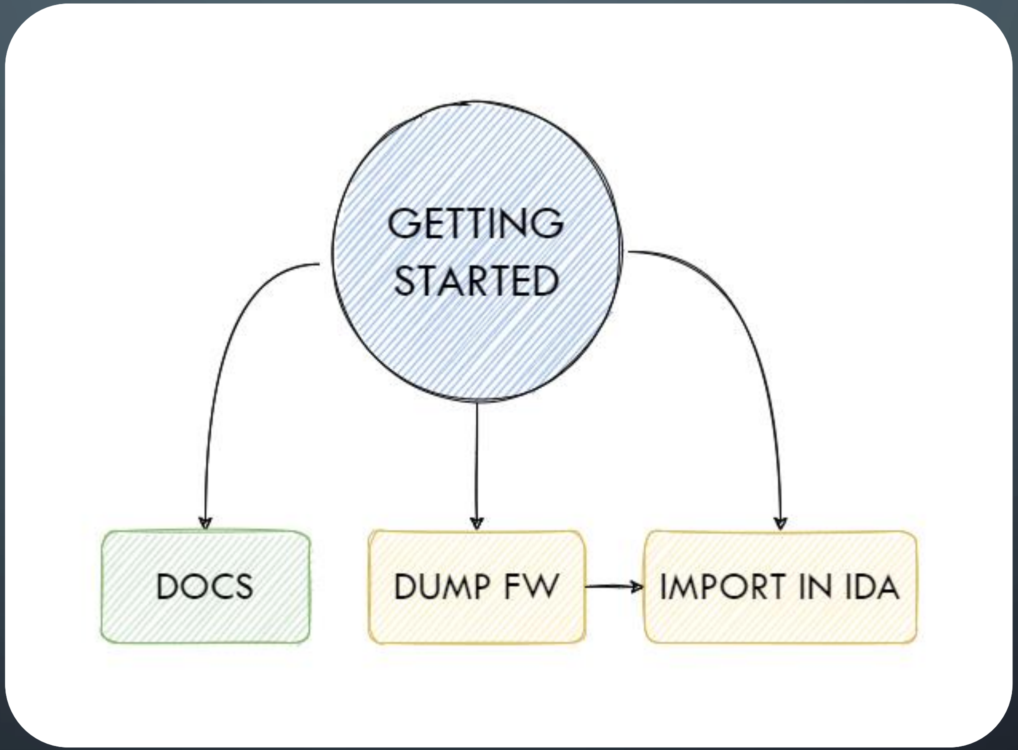
EMBEDDED LINUX

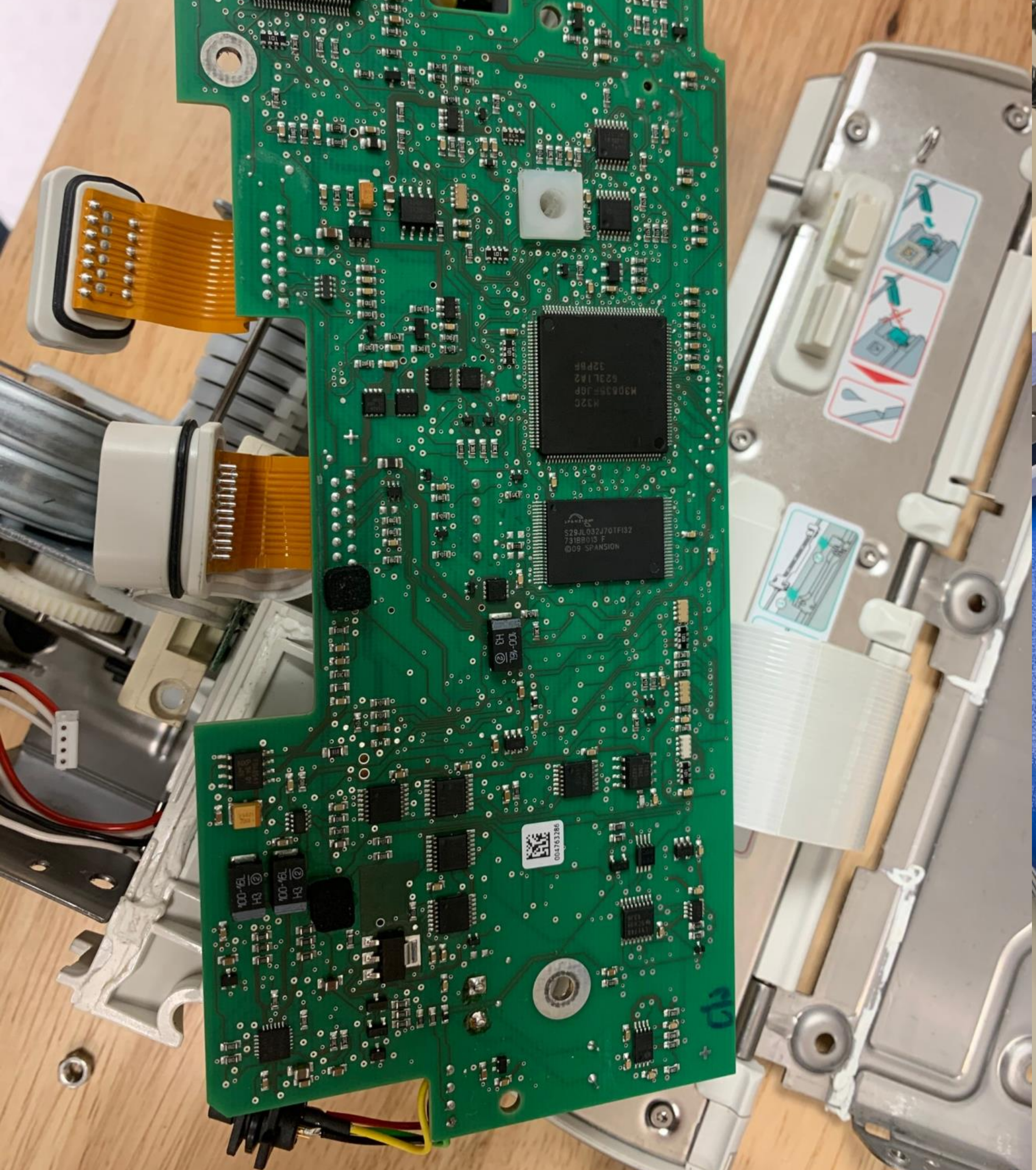
DIVIDE AND CONQUER

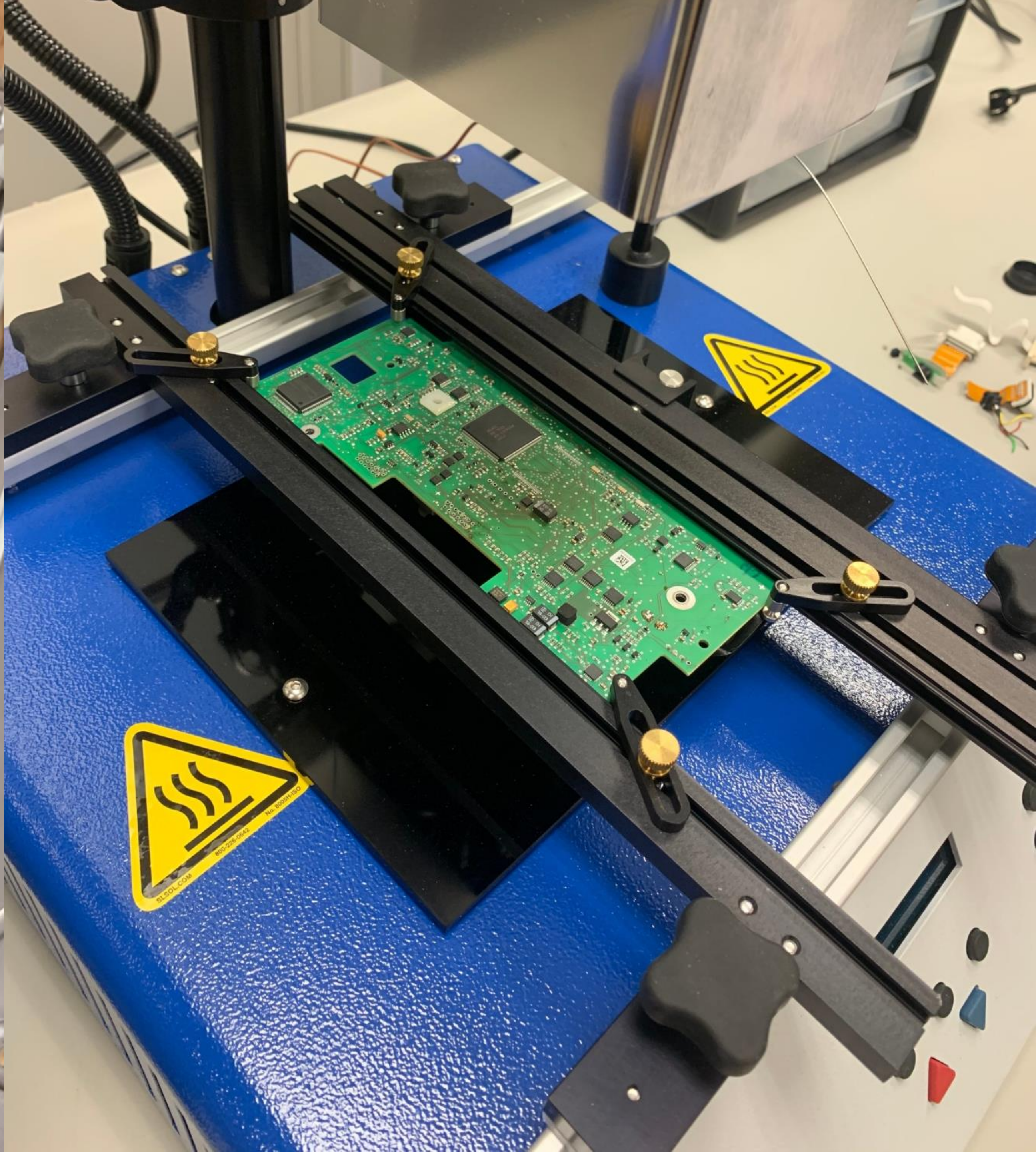
- Constraints
  - Limited lab access
  - Work done in Texas & France (6h time difference)
- Approach
  - Split the two logical components
  - RE the firmware and pwn communication module
  - Meet in the middle with compromised com module and firmware w/ interesting attack vectors uncovered

THE PUMP FIRMWARE

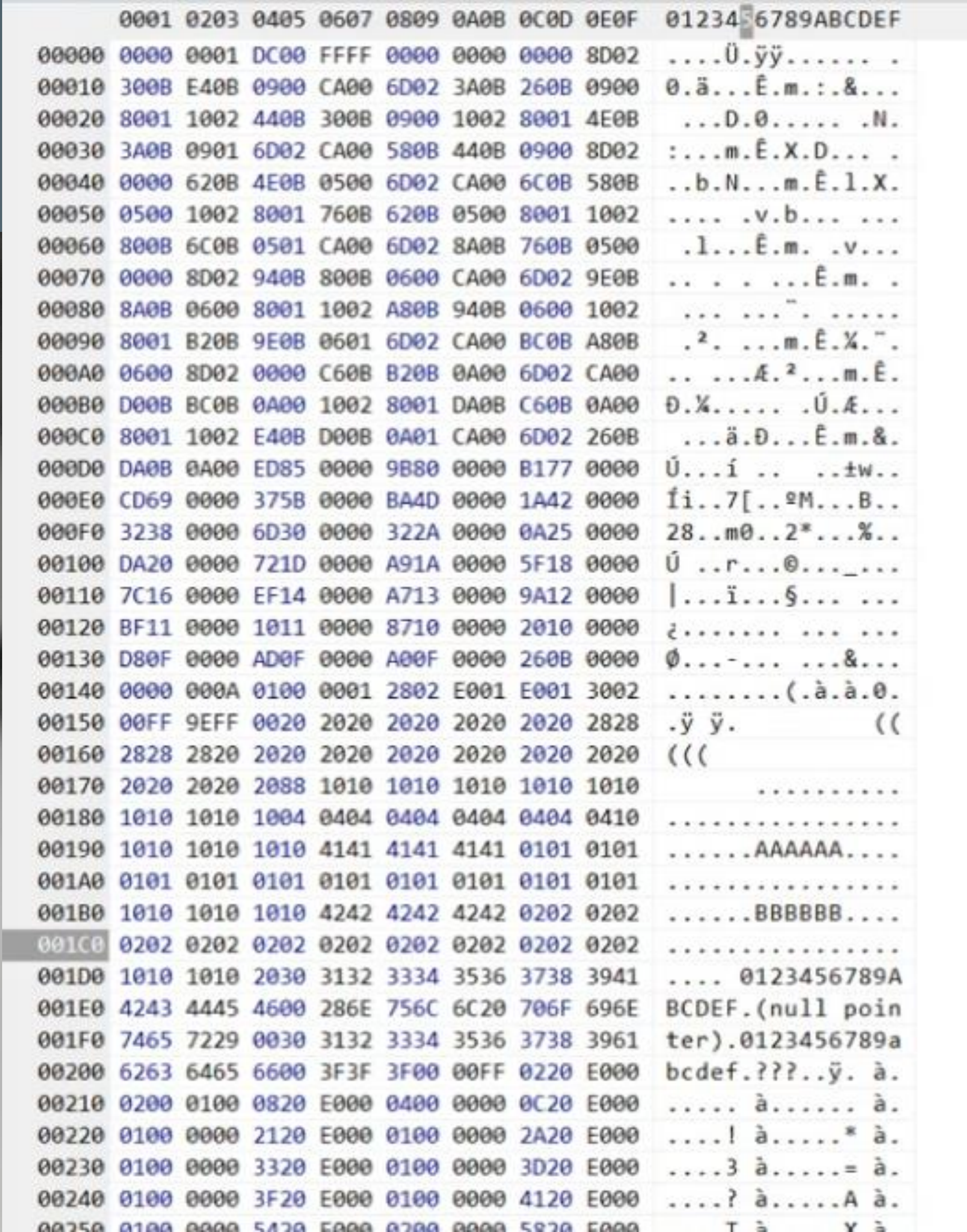

GETTING STARTED
LAY OF THE LAND
WHAT TO PWN?


GETTING STARTED
DOCS
DUMP FW
IMPORT IN IDA





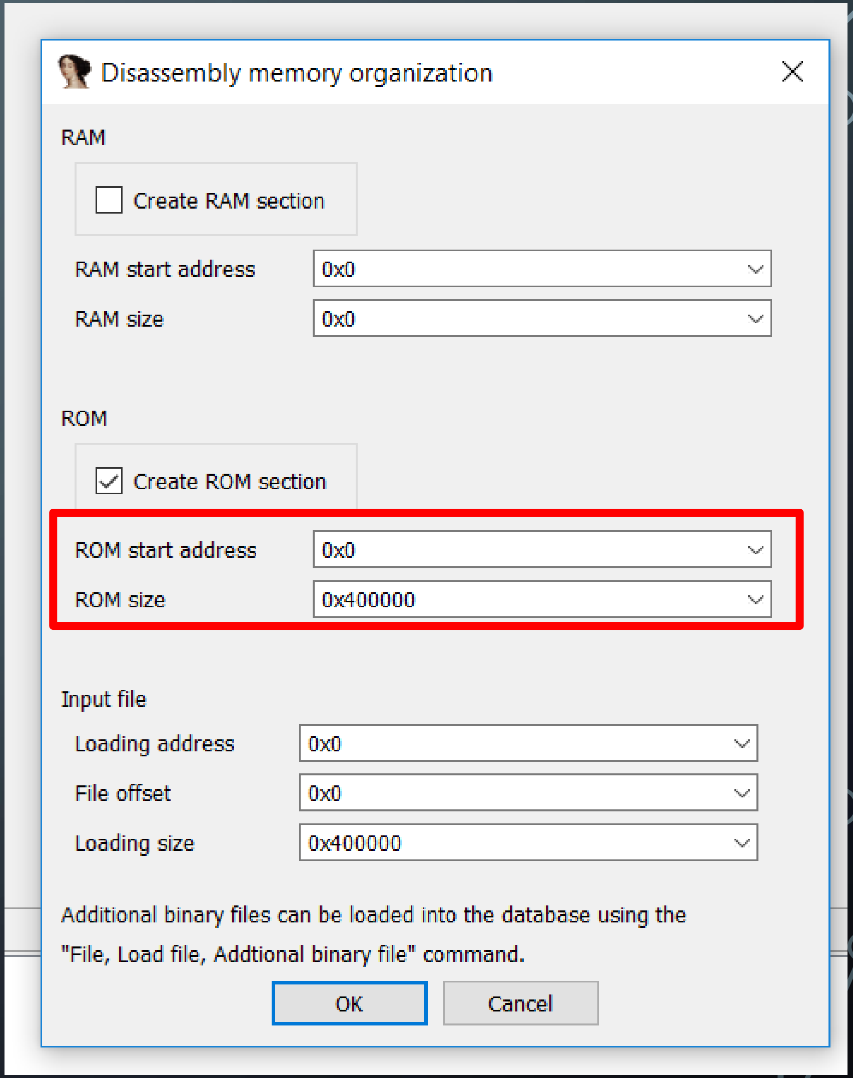
GETTING THE FIRMWARE OUT



```
      0001 0203 0405 0607 0809 0A0B 0C0D 0E0F  0123456789ABCDEF
00000 0000 0001 DC00 FFFF 0000 0000 0000 8D02  ....Ü.ÿÿ...... .
00010 300B E40B 0900 CA00 6D02 3A0B 260B 0900  0.ä...Ê.m.:.&...
00020 8001 1002 440B 300B 0900 1002 8001 4E0B   ...D.0..... .N.
00030 3A0B 0901 6D02 CA00 580B 440B 0900 8D02  :...m.Ê.X.D... .
00040 0000 620B 4E0B 0500 6D02 CA00 6C0B 580B  ..b.N...m.Ê.l.X.
00050 0500 1002 8001 760B 620B 0500 8001 1002  .... .v.b... ...
00060 800B 6C0B 0501 CA00 6D02 8A0B 760B 0500   .l...Ê.m. .v...
00070 0000 8D02 940B 800B 0600 CA00 6D02 9E0B  .. . . ...Ê.m. .
00080 8A0B 0600 8001 1002 A80B 940B 0600 1002   ... ...¨. .....
00090 8001 B20B 9E0B 0601 6D02 CA00 BC0B A80B   .². ...m.Ê.¼.¨.
000A0 0600 8D02 0000 C60B B20B 0A00 6D02 CA00  .. ...Æ.²...m.Ê.
000B0 D00B BC0B 0A00 1002 8001 DA0B C60B 0A00  Ð.¼..... .Ú.Æ...
000C0 8001 1002 E40B D00B 0A01 CA00 6D02 260B   ...ä.Ð...Ê.m.&.
000D0 DA0B 0A00 ED85 0000 9B80 0000 B177 0000  Ú...í ..  ..±w..
000E0 CD69 0000 375B 0000 BA4D 0000 1A42 0000  Íi..7[..ºM...B..
000F0 3238 0000 6D30 0000 322A 0000 0A25 0000  28..m0..2*...%..
00100 DA20 0000 721D 0000 A91A 0000 5F18 0000  Ú ..r...©..._...
00110 7C16 0000 EF14 0000 A713 0000 9A12 0000  |...ï...§... ...
00120 BF11 0000 1011 0000 8710 0000 2010 0000  ¿....... ... ...
00130 D80F 0000 AD0F 0000 A00F 0000 260B 0000  Ø...-... ...&...
00140 0000 000A 0100 0001 2802 E001 E001 3002  ........(.à.à.0.
00150 00FF 9EFF 0020 2020 2020 2020 2020 2828  .ÿ ÿ.          ((
00160 2828 2820 2020 2020 2020 2020 2020 2020  (((
00170 2020 2020 2088 1010 1010 1010 1010 1010       ..........
00180 1010 1010 1004 0404 0404 0404 0404 0410  ................
00190 1010 1010 1010 4141 4141 4141 0101 0101  ......AAAAAA....
001A0 0101 0101 0101 0101 0101 0101 0101 0101  ................
001B0 1010 1010 1010 4242 4242 4242 0202 0202  ......BBBBBB....
001C0 0202 0202 0202 0202 0202 0202 0202 0202  ................
001D0 1010 1010 2030 3132 3334 3536 3738 3941  .... 0123456789A
001E0 4243 4445 4600 286E 756C 6C20 706F 696E  BCDEF.(null poin
001F0 7465 7229 0030 3132 3334 3536 3738 3961  ter).0123456789a
00200 6263 6465 6600 3F3F 3F00 00FF 0220 E000  bcdef.???..ÿ. à.
00210 0200 0100 0820 E000 0400 0000 0C20 E000  ..... à...... à.
00220 0100 0000 2120 E000 0100 0000 2A20 E000  ....! à.....* à.
00230 0100 0000 3320 E000 0100 0000 3D20 E000  ....3 à.....= à.
00240 0100 0000 3F20 E000 0100 0000 4120 E000  ....? à.....A à.
00250 0100 0000 5420 E000 0200 0000 5820 E000  ....T à.....X à.
```

IMPORTING IN IDA?

- Support for M32C/80
  - We have a M32C/83, close enough….
- Finding the load address?
  - Common way
    - Strings, pointers, interrupt table ⇒ guess best candidate
  - Datasheet!



Disassembly memory organization

RAM

☐ Create RAM section

RAM start address: 0x0
RAM size: 0x0

ROM

☑ Create ROM section

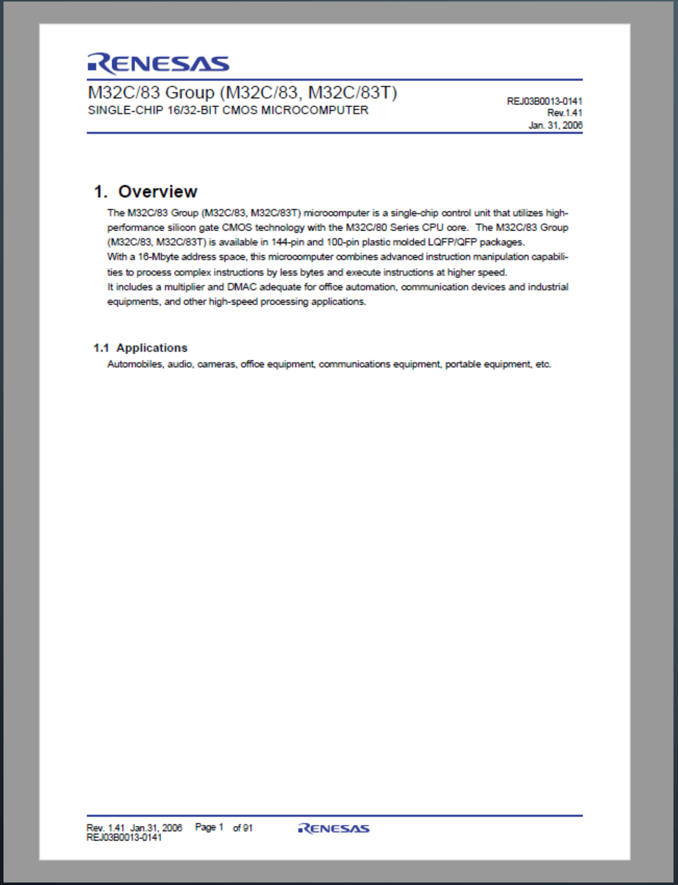
ROM start address: 0x0
ROM size: 0x400000

Input file

Loading address: 0x0
File offset: 0x0
Loading size: 0x400000

Additional binary files can be loaded into the database using the "File, Load file, Addtional binary file" command.

DATASHEET + MANUALS


M32C/83 Group (M32C/83, M32C/83T)
SINGLE-CHIP 16/32-BIT CMOS MICROCOMPUTER

REJ03B0013-0141
Rev.1.41
Jan. 31, 2006

1. Overview

The M32C/83 Group (M32C/83, M32C/83T) microcomputer is a single-chip control unit that utilizes high-performance silicon gate CMOS technology with the M32C/80 Series CPU core. The M32C/83 Group (M32C/83, M32C/83T) is available in 144-pin and 100-pin plastic molded LQFP/QFP packages.
With a 16-Mbyte address space, this microcomputer combines advanced instruction manipulation capabilities to process complex instructions by less bytes and execute instructions at higher speed.
It includes a multiplier and DMAC adequate for office automation, communication devices and industrial equipments, and other high-speed processing applications.

1.1 Applications

Automobiles, audio, cameras, office equipment, communications equipment, portable equipment, etc.

Rev. 1.41 Jan.31, 2006 Page 1 of 91
REJ03B0013-0141

REJ09B0319-0100

Everywhere you imagine.

16/32

M32C/80 Series
Software Manual

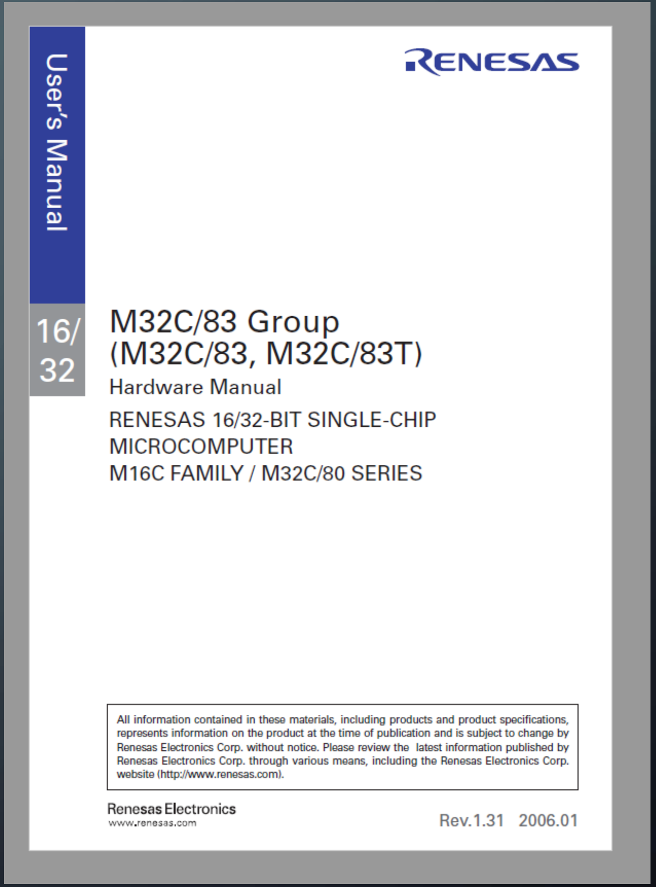
RENESAS 16/32-BIT SINGLE-CHIP MICROCOMPUTER
M16C FAMILY / M32C/80 SERIES

Software Manual

Before using this material, please visit our website to verify that this is the most updated document available.

Rev. 1.00
Revision date: May 31, 2006

Renesas Technology
www.renesas.com


User's Manual

16/32

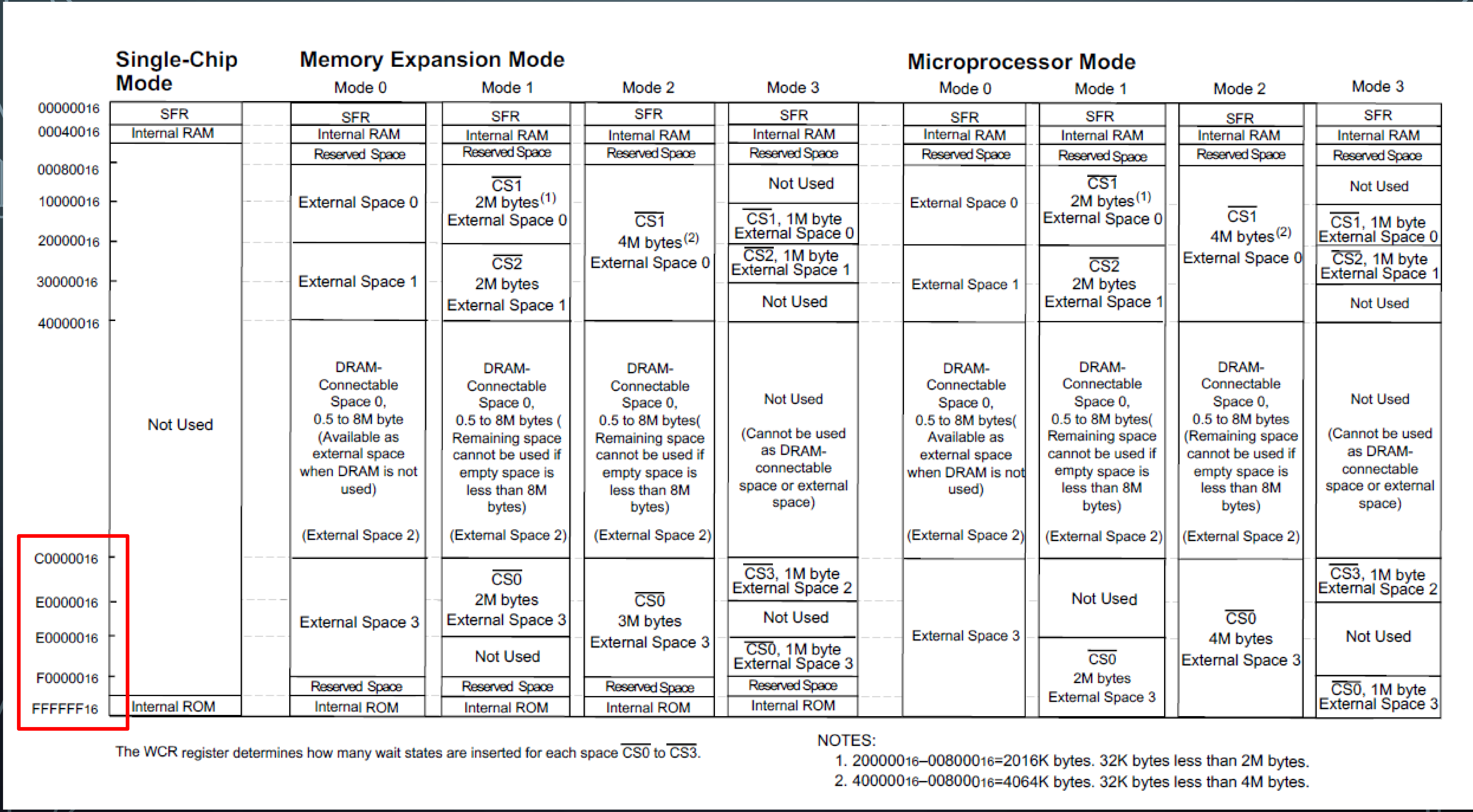
M32C/83 Group
(M32C/83, M32C/83T)
Hardware Manual

RENESAS 16/32-BIT SINGLE-CHIP MICROCOMPUTER
M16C FAMILY / M32C/80 SERIES

All information contained in these materials, including products and product specifications, represents information on the product at the time of publication and is subject to change by Renesas Electronics Corp. without notice. Please review the latest information published by Renesas Electronics Corp. through various means, including the Renesas Electronics Corp. website (http://www.renesas.com).

Renesas Electronics
www.renesas.com

Rev.1.31 2006.01



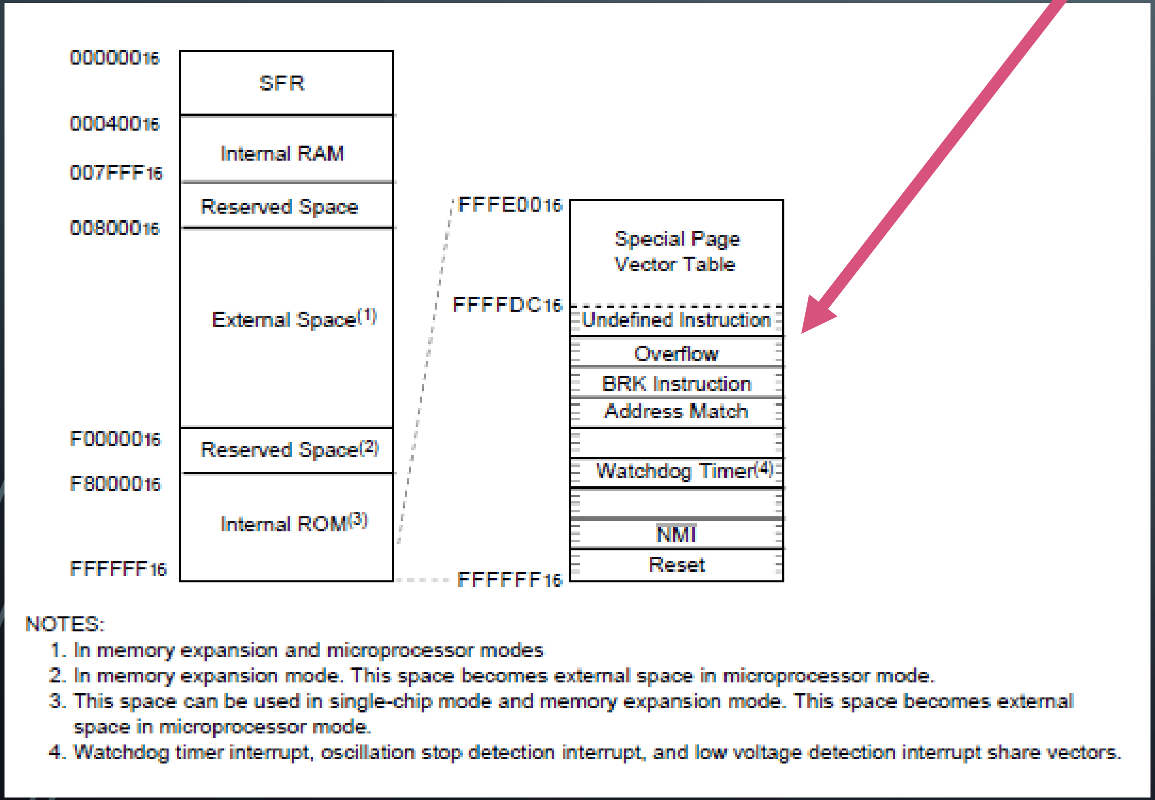
	Single-Chip Mode	Memory Expansion Mode				Microprocessor Mode			
		Mode 0	Mode 1	Mode 2	Mode 3	Mode 0	Mode 1	Mode 2	Mode 3
000000_{16}	SFR	SFR	SFR	SFR	SFR	SFR	SFR	SFR	SFR
000400_{16}	Internal RAM	Internal RAM	Internal RAM	Internal RAM	Internal RAM	Internal RAM	Internal RAM	Internal RAM	Internal RAM
	Not Used	Reserved Space	Reserved Space	Reserved Space	Reserved Space	Reserved Space	Reserved Space	Reserved Space	Reserved Space
008000_{16}		External Space 0	$\overline{CS1}$ 2M bytes(1) External Space 0	$\overline{CS1}$ 4M bytes(2) External Space 0	Not Used	External Space 0	$\overline{CS1}$ 2M bytes(1) External Space 0	$\overline{CS1}$ 4M bytes(2) External Space 0	Not Used
100000_{16}					$\overline{CS1}$, 1M byte External Space 0				$\overline{CS1}$, 1M byte External Space 0
200000_{16}		External Space 1	$\overline{CS2}$ 2M bytes External Space 1		$\overline{CS2}$, 1M byte External Space 1	External Space 1	$\overline{CS2}$ 2M bytes External Space 1		$\overline{CS2}$, 1M byte External Space 1
300000_{16}					Not Used				Not Used
400000_{16}		DRAM-Connectable Space 0, 0.5 to 8M byte (Available as external space when DRAM is not used) (External Space 2)	DRAM-Connectable Space 0, 0.5 to 8M bytes (Remaining space cannot be used if empty space is less than 8M bytes) (External Space 2)	DRAM-Connectable Space 0, 0.5 to 8M bytes(Remaining space cannot be used if empty space is less than 8M bytes) (External Space 2)	Not Used (Cannot be used as DRAM-connectable space or external space)	DRAM-Connectable Space 0, 0.5 to 8M bytes(Available as external space when DRAM is not used) (External Space 2)	DRAM-Connectable Space 0, 0.5 to 8M bytes(Remaining space cannot be used if empty space is less than 8M bytes) (External Space 2)	DRAM-Connectable Space 0, 0.5 to 8M bytes (Remaining space cannot be used if empty space is less than 8M bytes) (External Space 2)	Not Used (Cannot be used as DRAM-connectable space or external space)
$C00000_{16}$		External Space 3	$\overline{CS0}$ 2M bytes External Space 3	$\overline{CS0}$ 3M bytes External Space 3	$\overline{CS3}$, 1M byte External Space 2	External Space 3	Not Used	$\overline{CS0}$ 4M bytes External Space 3	$\overline{CS3}$, 1M byte External Space 2
$E00000_{16}$					Not Used				Not Used
$E00000_{16}$			Not Used		$\overline{CS0}$, 1M byte External Space 3		$\overline{CS0}$ 2M bytes External Space 3		
$F00000_{16}$		Reserved Space	Reserved Space	Reserved Space	Reserved Space				$\overline{CS0}$, 1M byte External Space 3
$FFFFFF_{16}$	Internal ROM	Internal ROM	Internal ROM	Internal ROM	Internal ROM				

The WCR register determines how many wait states are inserted for each space $\overline{CS0}$ to $\overline{CS3}$.

NOTES:
1. 200000_{16}–008000_{16}=2016K bytes. 32K bytes less than 2M bytes.
2. 400000_{16}–008000_{16}=4064K bytes. 32K bytes less than 4M bytes.

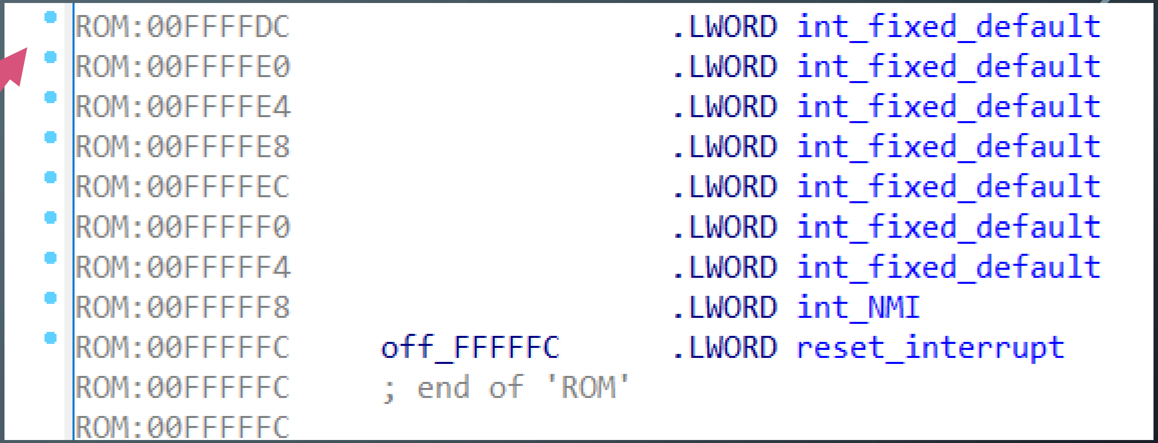
MEMORY MAP

Datasheet: There are function pointers at
0xFFFFDC - 0xFFFFFF

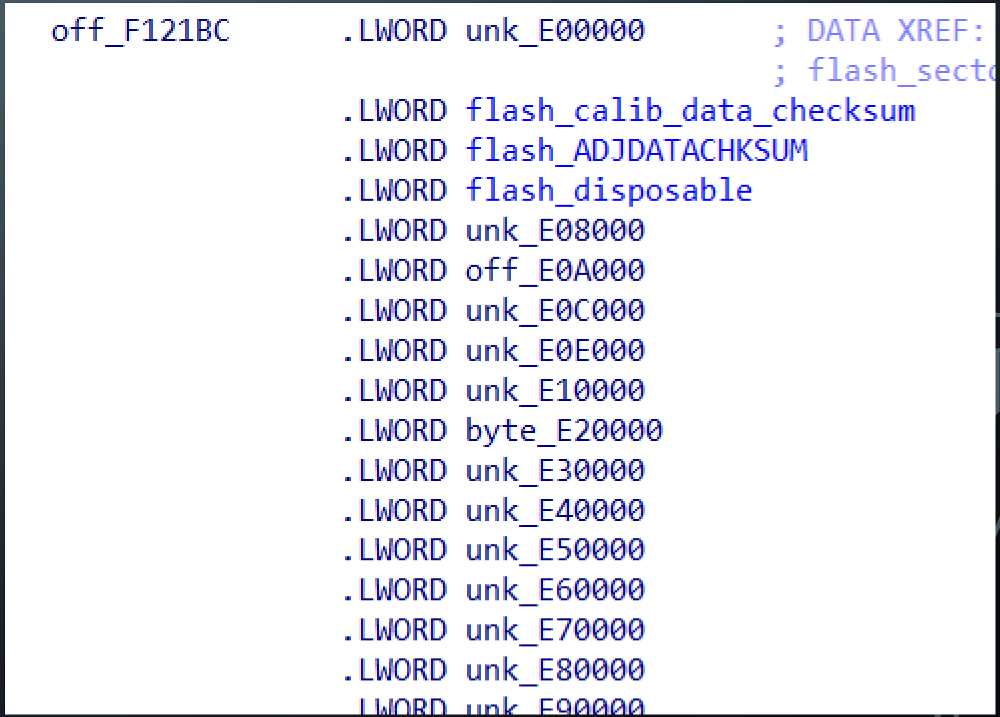


NOTES:
1. In memory expansion and microprocessor modes
2. In memory expansion mode. This space becomes external space in microprocessor mode.
3. This space can be used in single-chip mode and memory expansion mode. This space becomes external space in microprocessor mode.
4. Watchdog timer interrupt, oscillation stop detection interrupt, and low voltage detection interrupt share vectors.

Result when we load the FW at 0xE00000

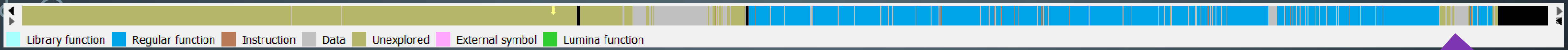


```
ROM:00FFFFDC                    .LWORD int_fixed_default
ROM:00FFFFE0                    .LWORD int_fixed_default
ROM:00FFFFE4                    .LWORD int_fixed_default
ROM:00FFFFE8                    .LWORD int_fixed_default
ROM:00FFFFEC                    .LWORD int_fixed_default
ROM:00FFFFF0                    .LWORD int_fixed_default
ROM:00FFFFF4                    .LWORD int_fixed_default
ROM:00FFFFF8                    .LWORD int_NMI
ROM:00FFFFFC     off_FFFFFC     .LWORD reset_interrupt
ROM:00FFFFFC     ; end of 'ROM'
ROM:00FFFFFC
```

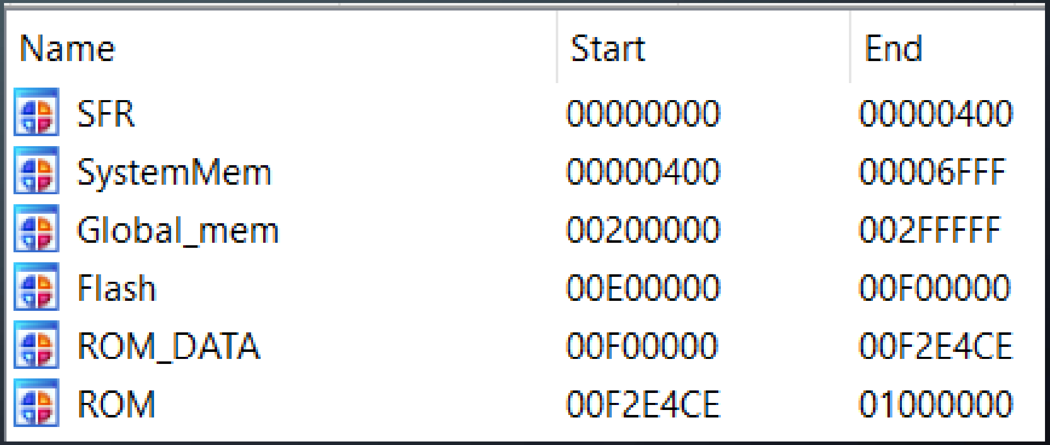


```
off_F121BC     .LWORD unk_E00000          ; DATA XREF:
                                          ; flash_secto
               .LWORD flash_calib_data_checksum
               .LWORD flash_ADJDATACHKSUM
               .LWORD flash_disposable
               .LWORD unk_E08000
               .LWORD off_E0A000
               .LWORD unk_E0C000
               .LWORD unk_E0E000
               .LWORD unk_E10000
               .LWORD byte_E20000
               .LWORD unk_E30000
               .LWORD unk_E40000
               .LWORD unk_E50000
               .LWORD unk_E60000
               .LWORD unk_E70000
               .LWORD unk_E80000
               .LWORD unk_E90000
```

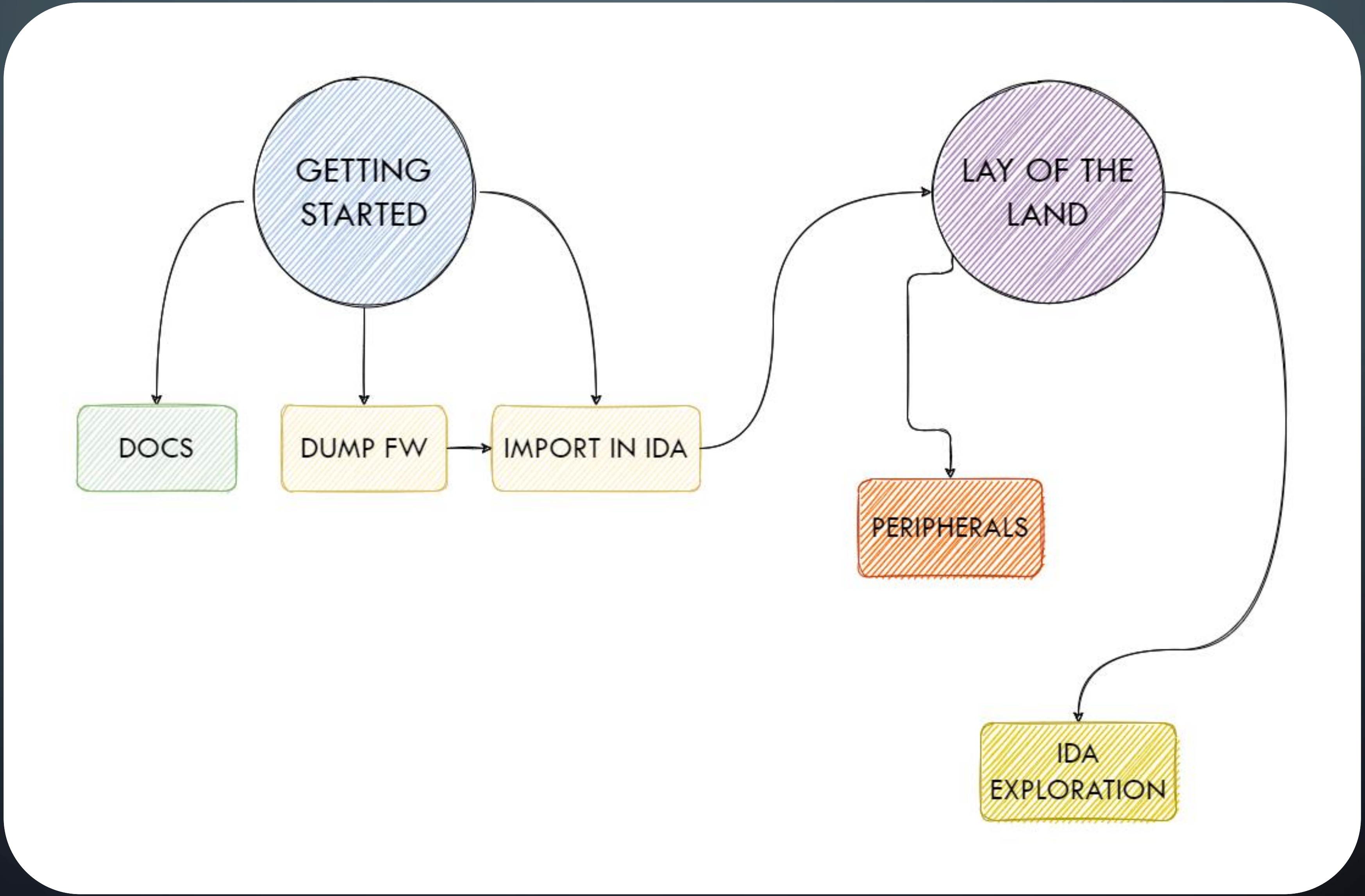
MEMORY MAP – IDA



- Two sections of code
  - Main application (0xF2E4CE – 0xFEFC00)
  - Bootloader / Monitor code (0xFF9000 – 0x0FFE6E4)
- Segments as defined in IDA
  - Special Function Register
  - Internal RAM (used by RTOS)
  - Application RAM
  - Flash: Configuration, Calibration
  - Flash: Code



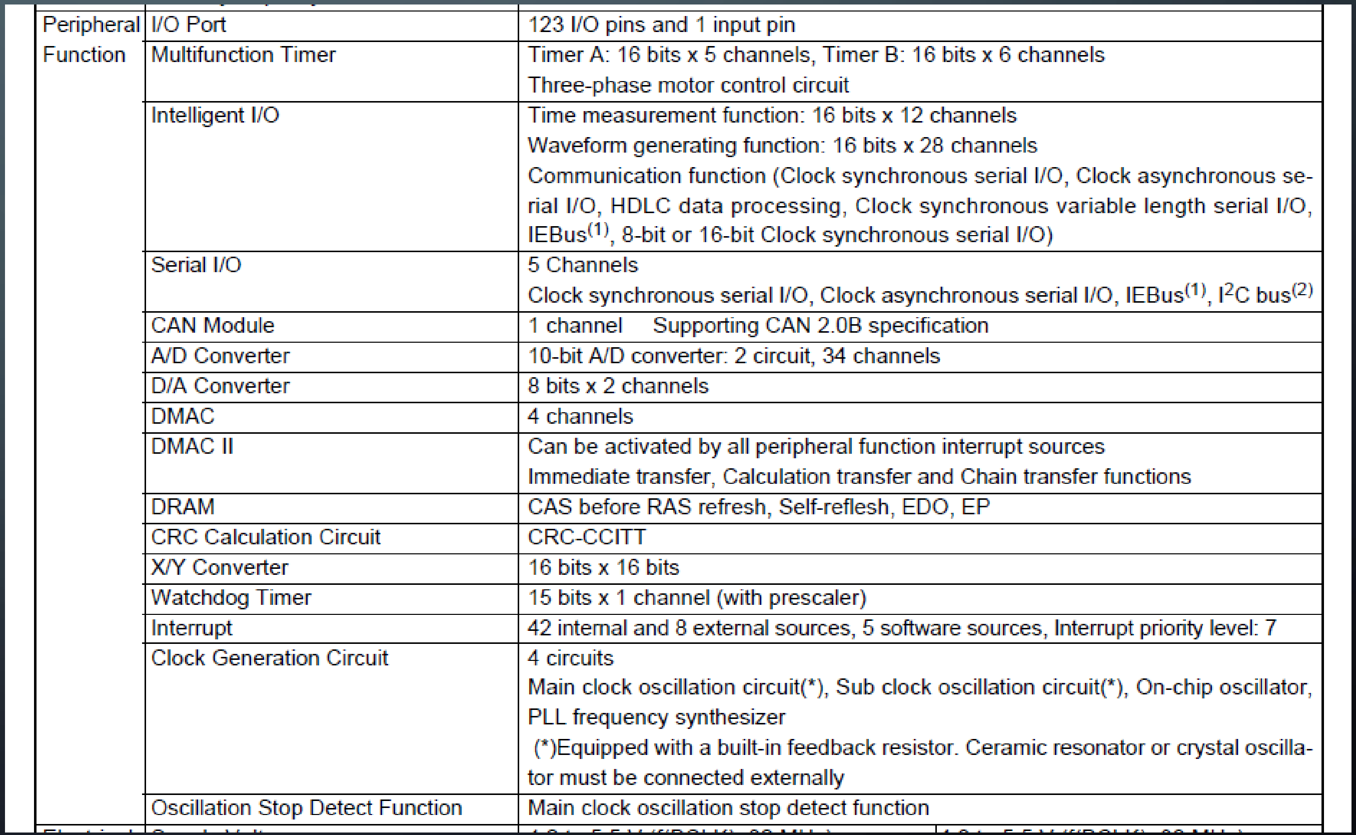
Name	Start	End
SFR	00000000	00000400
SystemMem	00000400	00006FFF
Global_mem	00200000	002FFFFF
Flash	00E00000	00F00000
ROM_DATA	00F00000	00F2E4CE
ROM	00F2E4CE	01000000


GETTING STARTED
LAY OF THE LAND
DOCS
DUMP FW
IMPORT IN IDA
PERIPHERALS
IDA EXPLORATION

PERIPHERALS OF INTEREST

- How the firmware communicates with the rest of the world:
  - UART
  - CAN bus
  - Tons of I/O pins
  - A/D and D/A converters

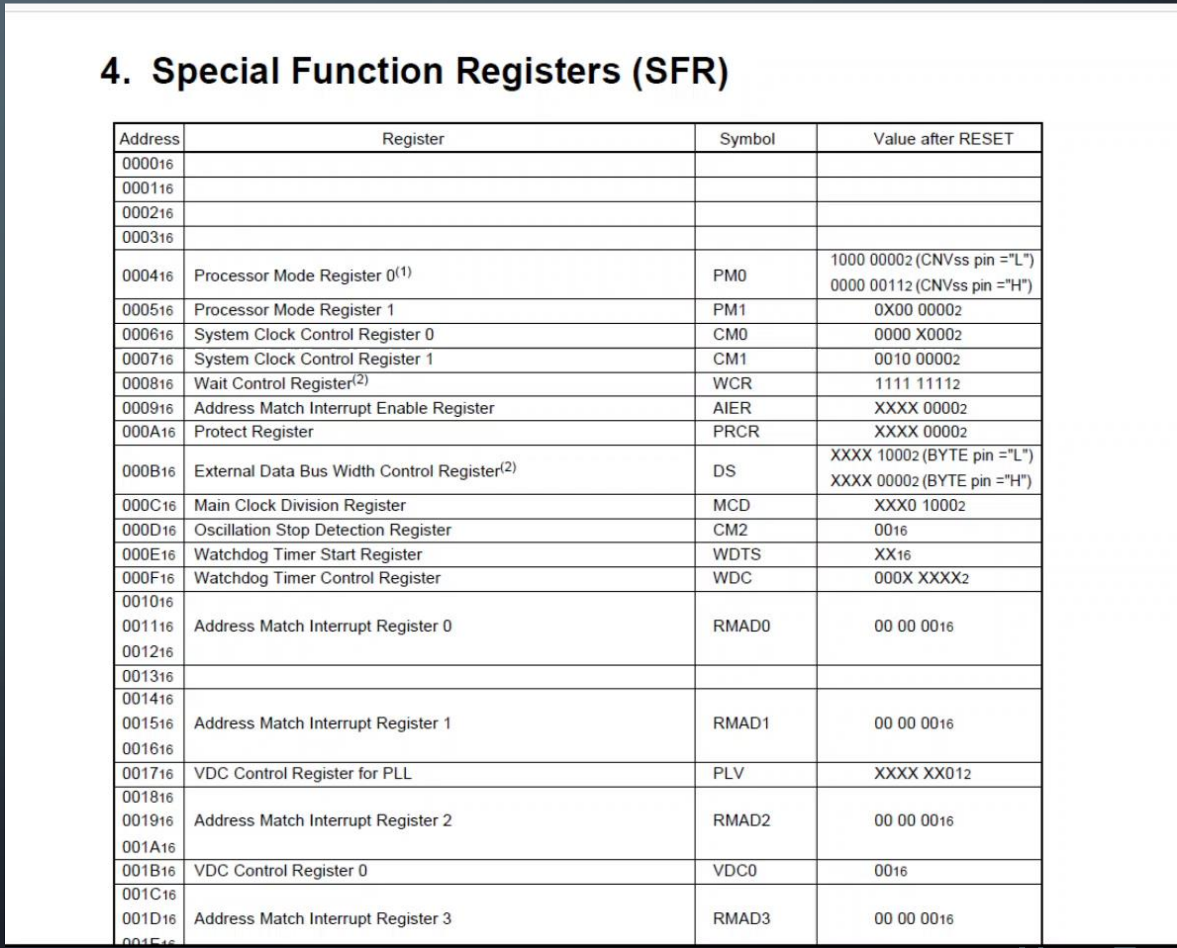
    → useful for physical processes
- Two ways to find them
  - SFR
  - Interrupt handlers



Peripheral Function	I/O Port	123 I/O pins and 1 input pin
	Multifunction Timer	Timer A: 16 bits x 5 channels, Timer B: 16 bits x 6 channels Three-phase motor control circuit
	Intelligent I/O	Time measurement function: 16 bits x 12 channels Waveform generating function: 16 bits x 28 channels Communication function (Clock synchronous serial I/O, Clock asynchronous serial I/O, HDLC data processing, Clock synchronous variable length serial I/O, IEBus(1), 8-bit or 16-bit Clock synchronous serial I/O)
	Serial I/O	5 Channels Clock synchronous serial I/O, Clock asynchronous serial I/O, IEBus(1), I^{2}C bus(2)
	CAN Module	1 channel Supporting CAN 2.0B specification
	A/D Converter	10-bit A/D converter: 2 circuit, 34 channels
	D/A Converter	8 bits x 2 channels
	DMAC	4 channels
	DMAC II	Can be activated by all peripheral function interrupt sources Immediate transfer, Calculation transfer and Chain transfer functions
	DRAM	CAS before RAS refresh, Self-reflesh, EDO, EP
	CRC Calculation Circuit	CRC-CCITT
	X/Y Converter	16 bits x 16 bits
	Watchdog Timer	15 bits x 1 channel (with prescaler)
	Interrupt	42 internal and 8 external sources, 5 software sources, Interrupt priority level: 7
	Clock Generation Circuit	4 circuits Main clock oscillation circuit(*), Sub clock oscillation circuit(*), On-chip oscillator, PLL frequency synthesizer (*)Equipped with a built-in feedback resistor. Ceramic resonator or crystal oscillator must be connected externally
	Oscillation Stop Detect Function	Main clock oscillation stop detect function

FINDING THE FUN VECTORS

- SFR (Special Function Registers)
  - Special Memory [0x0, 0x400]
  - Read/Write/Configure devices
  - Can be XREF-ed
- Interrupt handler
  - Addr in INTB register
  - Function ptr triggered when something interesting happens
  - UART, CAN, Timers, …



4. Special Function Registers (SFR)

Address	Register	Symbol	Value after RESET
0000_{16}			
0001_{16}			
0002_{16}			
0003_{16}			
0004_{16}	Processor Mode Register 0(1)	PM0	$1000\ 0000_2$ (CNVss pin ="L") $0000\ 0011_2$ (CNVss pin ="H")
0005_{16}	Processor Mode Register 1	PM1	$0X00\ 0000_2$
0006_{16}	System Clock Control Register 0	CM0	$0000\ X000_2$
0007_{16}	System Clock Control Register 1	CM1	$0010\ 0000_2$
0008_{16}	Wait Control Register(2)	WCR	$1111\ 1111_2$
0009_{16}	Address Match Interrupt Enable Register	AIER	$XXXX\ 0000_2$
$000A_{16}$	Protect Register	PRCR	$XXXX\ 0000_2$
$000B_{16}$	External Data Bus Width Control Register(2)	DS	$XXXX\ 1000_2$ (BYTE pin ="L") $XXXX\ 0000_2$ (BYTE pin ="H")
$000C_{16}$	Main Clock Division Register	MCD	$XXX0\ 1000_2$
$000D_{16}$	Oscillation Stop Detection Register	CM2	00_{16}
$000E_{16}$	Watchdog Timer Start Register	WDTS	XX_{16}
$000F_{16}$	Watchdog Timer Control Register	WDC	$000X\ XXXX_2$
0010_{16} 0011_{16} 0012_{16}	Address Match Interrupt Register 0	RMAD0	$00\ 00\ 00_{16}$
0013_{16}			
0014_{16} 0015_{16} 0016_{16}	Address Match Interrupt Register 1	RMAD1	$00\ 00\ 00_{16}$
0017_{16}	VDC Control Register for PLL	PLV	$XXXX\ XX01_2$
0018_{16} 0019_{16} $001A_{16}$	Address Match Interrupt Register 2	RMAD2	$00\ 00\ 00_{16}$
$001B_{16}$	VDC Control Register 0	VDC0	00_{16}
$001C_{16}$ $001D_{16}$	Address Match Interrupt Register 3	RMAD3	$00\ 00\ 00_{16}$

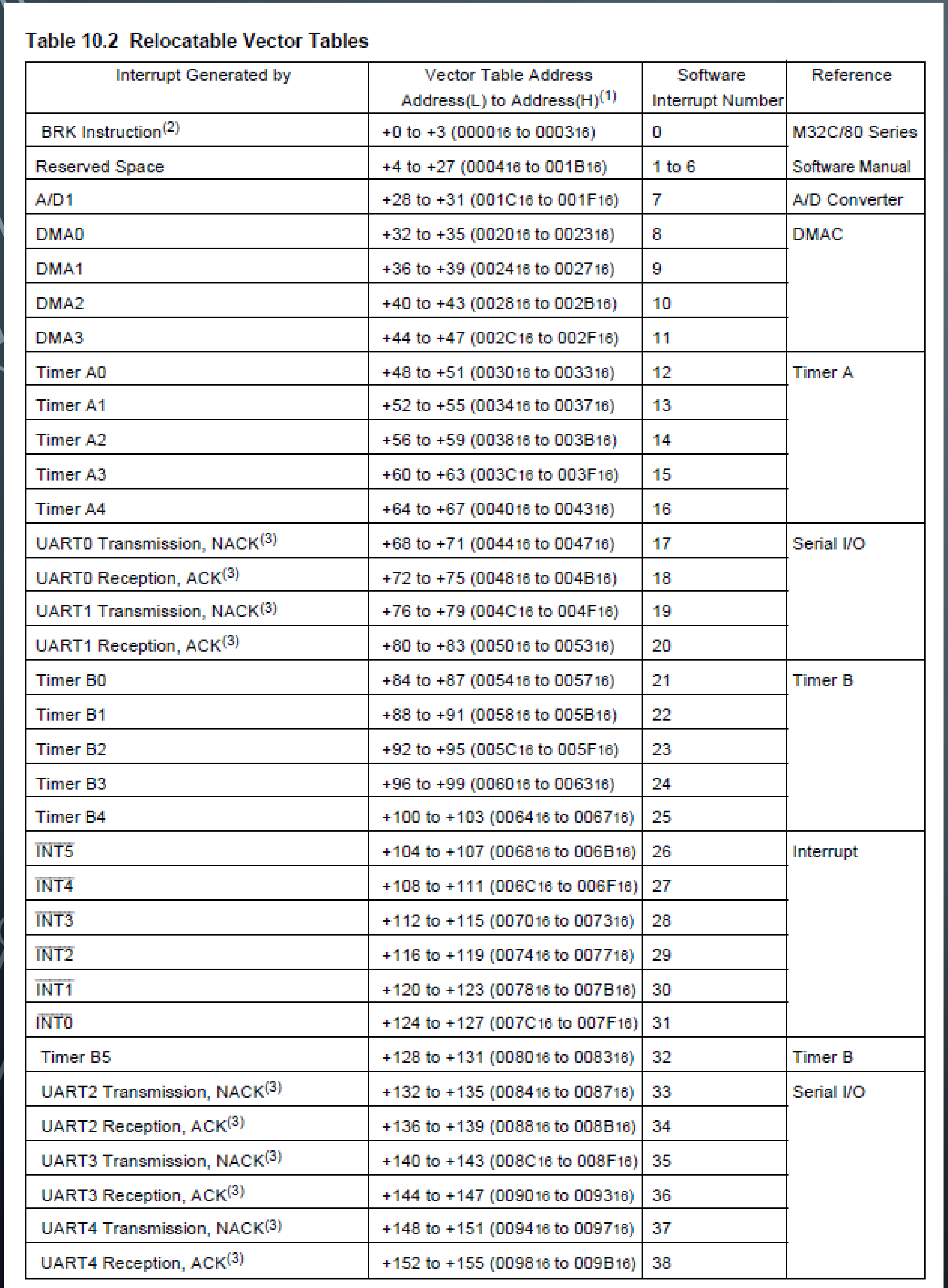


Table 10.2 Relocatable Vector Tables

Interrupt Generated by	Vector Table Address Address(L) to Address(H)[1]	Software Interrupt Number	Reference
BRK Instruction[2]	+0 to +3 (0000_{16} to 0003_{16})	0	M32C/80 Series
Reserved Space	+4 to +27 (0004_{16} to $001B_{16}$)	1 to 6	Software Manual
A/D1	+28 to +31 ($001C_{16}$ to $001F_{16}$)	7	A/D Converter
DMA0	+32 to +35 (0020_{16} to 0023_{16})	8	DMAC
DMA1	+36 to +39 (0024_{16} to 0027_{16})	9	
DMA2	+40 to +43 (0028_{16} to $002B_{16}$)	10	
DMA3	+44 to +47 ($002C_{16}$ to $002F_{16}$)	11	
Timer A0	+48 to +51 (0030_{16} to 0033_{16})	12	Timer A
Timer A1	+52 to +55 (0034_{16} to 0037_{16})	13	
Timer A2	+56 to +59 (0038_{16} to $003B_{16}$)	14	
Timer A3	+60 to +63 ($003C_{16}$ to $003F_{16}$)	15	
Timer A4	+64 to +67 (0040_{16} to 0043_{16})	16	
UART0 Transmission, NACK[3]	+68 to +71 (0044_{16} to 0047_{16})	17	Serial I/O
UART0 Reception, ACK[3]	+72 to +75 (0048_{16} to $004B_{16}$)	18	
UART1 Transmission, NACK[3]	+76 to +79 ($004C_{16}$ to $004F_{16}$)	19	
UART1 Reception, ACK[3]	+80 to +83 (0050_{16} to 0053_{16})	20	
Timer B0	+84 to +87 (0054_{16} to 0057_{16})	21	Timer B
Timer B1	+88 to +91 (0058_{16} to $005B_{16}$)	22	
Timer B2	+92 to +95 ($005C_{16}$ to $005F_{16}$)	23	
Timer B3	+96 to +99 (0060_{16} to 0063_{16})	24	
Timer B4	+100 to +103 (0064_{16} to 0067_{16})	25	
$\overline{\text{INT5}}$	+104 to +107 (0068_{16} to $006B_{16}$)	26	Interrupt
$\overline{\text{INT4}}$	+108 to +111 ($006C_{16}$ to $006F_{16}$)	27	
$\overline{\text{INT3}}$	+112 to +115 (0070_{16} to 0073_{16})	28	
$\overline{\text{INT2}}$	+116 to +119 (0074_{16} to 0077_{16})	29	
$\overline{\text{INT1}}$	+120 to +123 (0078_{16} to $007B_{16}$)	30	
$\overline{\text{INT0}}$	+124 to +127 ($007C_{16}$ to $007F_{16}$)	31	
Timer B5	+128 to +131 (0080_{16} to 0083_{16})	32	Timer B
UART2 Transmission, NACK[3]	+132 to +135 (0084_{16} to 0087_{16})	33	Serial I/O
UART2 Reception, ACK[3]	+136 to +139 (0088_{16} to $008B_{16}$)	34	
UART3 Transmission, NACK[3]	+140 to +143 ($008C_{16}$ to $008F_{16}$)	35	
UART3 Reception, ACK[3]	+144 to +147 (0090_{16} to 0093_{16})	36	
UART4 Transmission, NACK[3]	+148 to +151 (0094_{16} to 0097_{16})	37	
UART4 Reception, ACK[3]	+152 to +155 (0098_{16} to $009B_{16}$)	38	

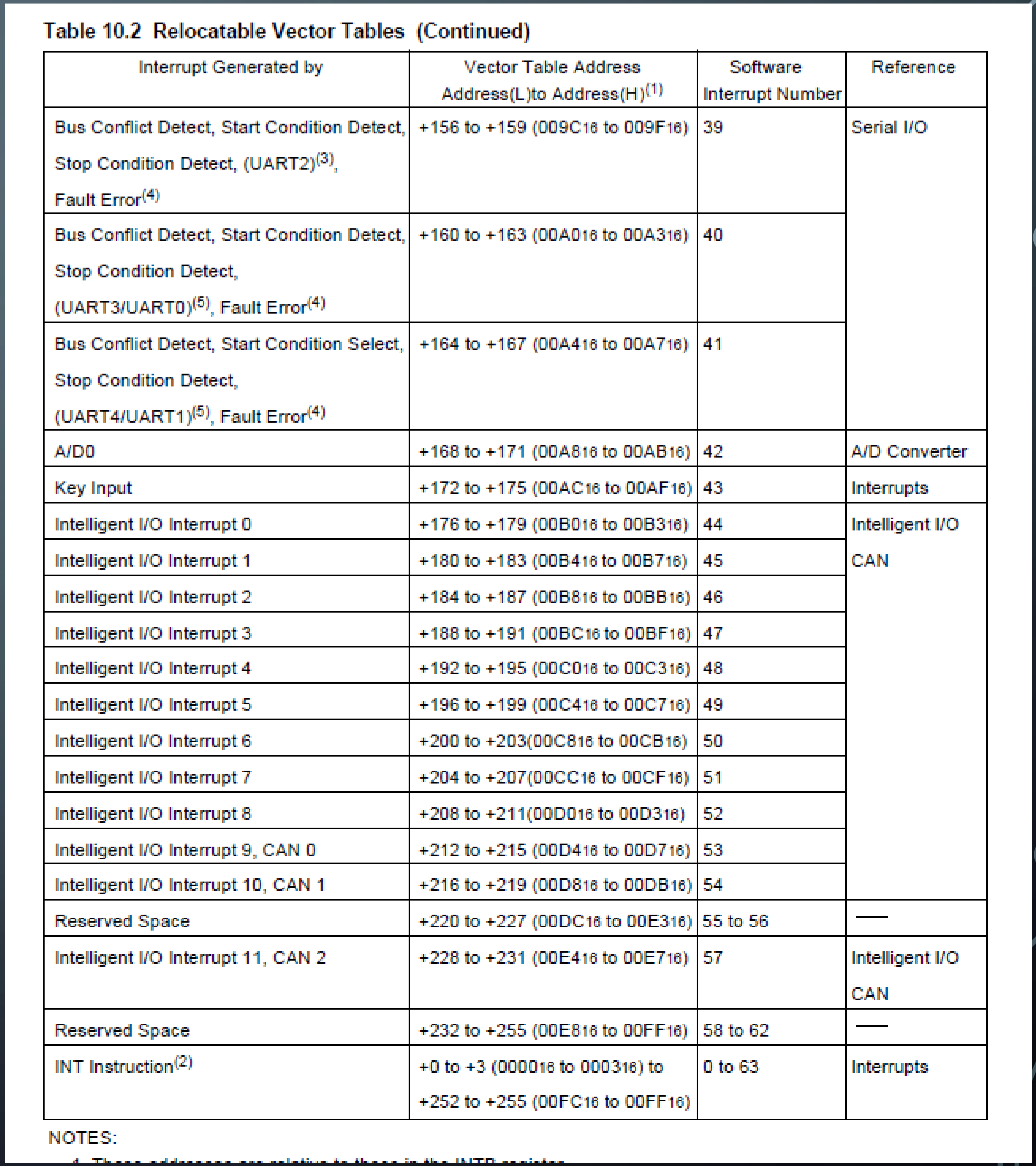
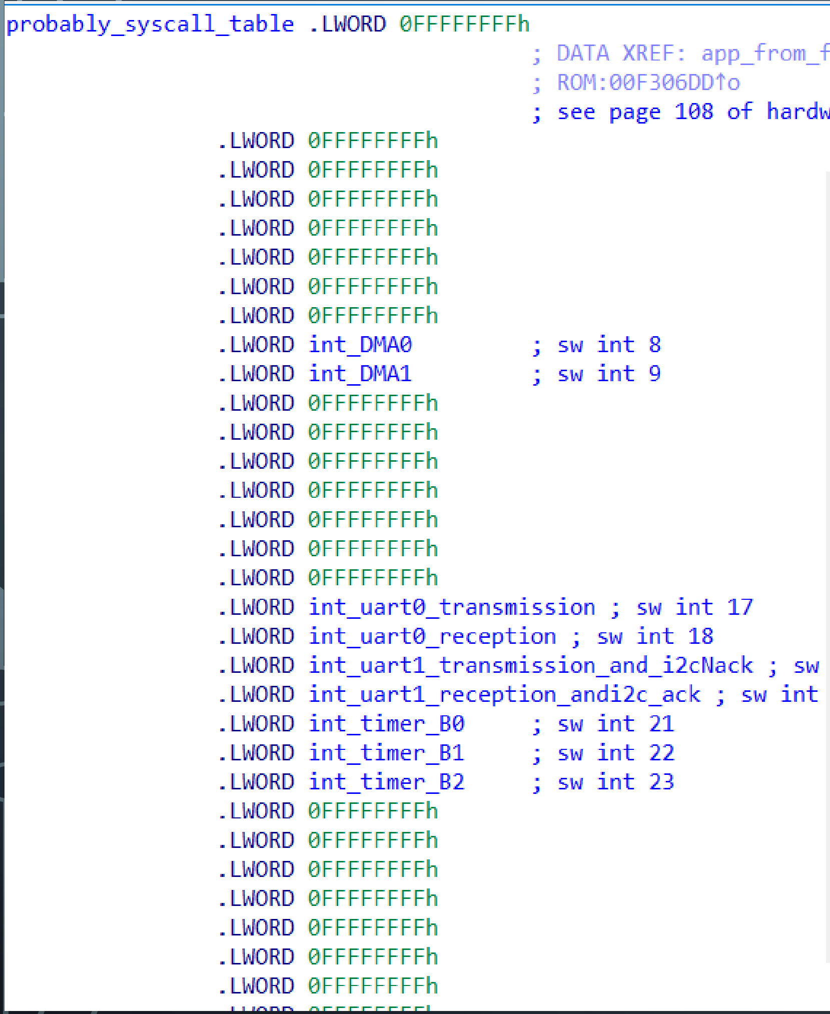


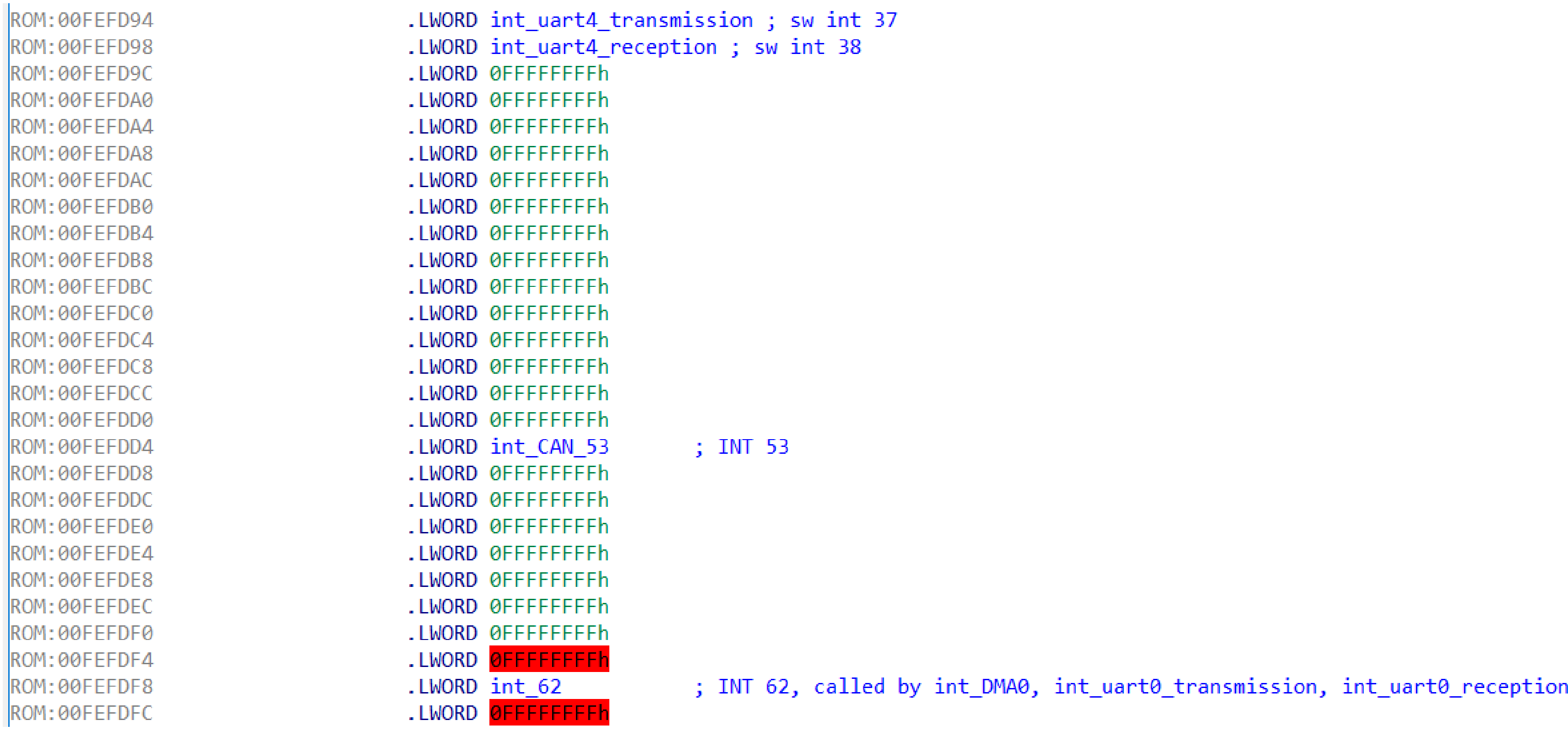
Table 10.2 Relocatable Vector Tables (Continued)

Interrupt Generated by	Vector Table Address Address(L)to Address(H)[1]	Software Interrupt Number	Reference
Bus Conflict Detect, Start Condition Detect, Stop Condition Detect, (UART2)[3], Fault Error[4]	+156 to +159 ($009C_{16}$ to $009F_{16}$)	39	Serial I/O
Bus Conflict Detect, Start Condition Detect, Stop Condition Detect, (UART3/UART0)[5], Fault Error[4]	+160 to +163 ($00A0_{16}$ to $00A3_{16}$)	40	
Bus Conflict Detect, Start Condition Select, Stop Condition Detect, (UART4/UART1)[5], Fault Error[4]	+164 to +167 ($00A4_{16}$ to $00A7_{16}$)	41	
A/D0	+168 to +171 ($00A8_{16}$ to $00AB_{16}$)	42	A/D Converter
Key Input	+172 to +175 ($00AC_{16}$ to $00AF_{16}$)	43	Interrupts
Intelligent I/O Interrupt 0	+176 to +179 ($00B0_{16}$ to $00B3_{16}$)	44	Intelligent I/O CAN
Intelligent I/O Interrupt 1	+180 to +183 ($00B4_{16}$ to $00B7_{16}$)	45	
Intelligent I/O Interrupt 2	+184 to +187 ($00B8_{16}$ to $00BB_{16}$)	46	
Intelligent I/O Interrupt 3	+188 to +191 ($00BC_{16}$ to $00BF_{16}$)	47	
Intelligent I/O Interrupt 4	+192 to +195 ($00C0_{16}$ to $00C3_{16}$)	48	
Intelligent I/O Interrupt 5	+196 to +199 ($00C4_{16}$ to $00C7_{16}$)	49	
Intelligent I/O Interrupt 6	+200 to +203($00C8_{16}$ to $00CB_{16}$)	50	
Intelligent I/O Interrupt 7	+204 to +207($00CC_{16}$ to $00CF_{16}$)	51	
Intelligent I/O Interrupt 8	+208 to +211($00D0_{16}$ to $00D3_{16}$)	52	
Intelligent I/O Interrupt 9, CAN 0	+212 to +215 ($00D4_{16}$ to $00D7_{16}$)	53	
Intelligent I/O Interrupt 10, CAN 1	+216 to +219 ($00D8_{16}$ to $00DB_{16}$)	54	
Reserved Space	+220 to +227 ($00DC_{16}$ to $00E3_{16}$)	55 to 56	—
Intelligent I/O Interrupt 11, CAN 2	+228 to +231 ($00E4_{16}$ to $00E7_{16}$)	57	Intelligent I/O CAN
Reserved Space	+232 to +255 ($00E8_{16}$ to $00FF_{16}$)	58 to 62	—
INT Instruction[2]	+0 to +3 (0000_{16} to 0003_{16}) to +252 to +255 ($00FC_{16}$ to $00FF_{16}$)	0 to 63	Interrupts

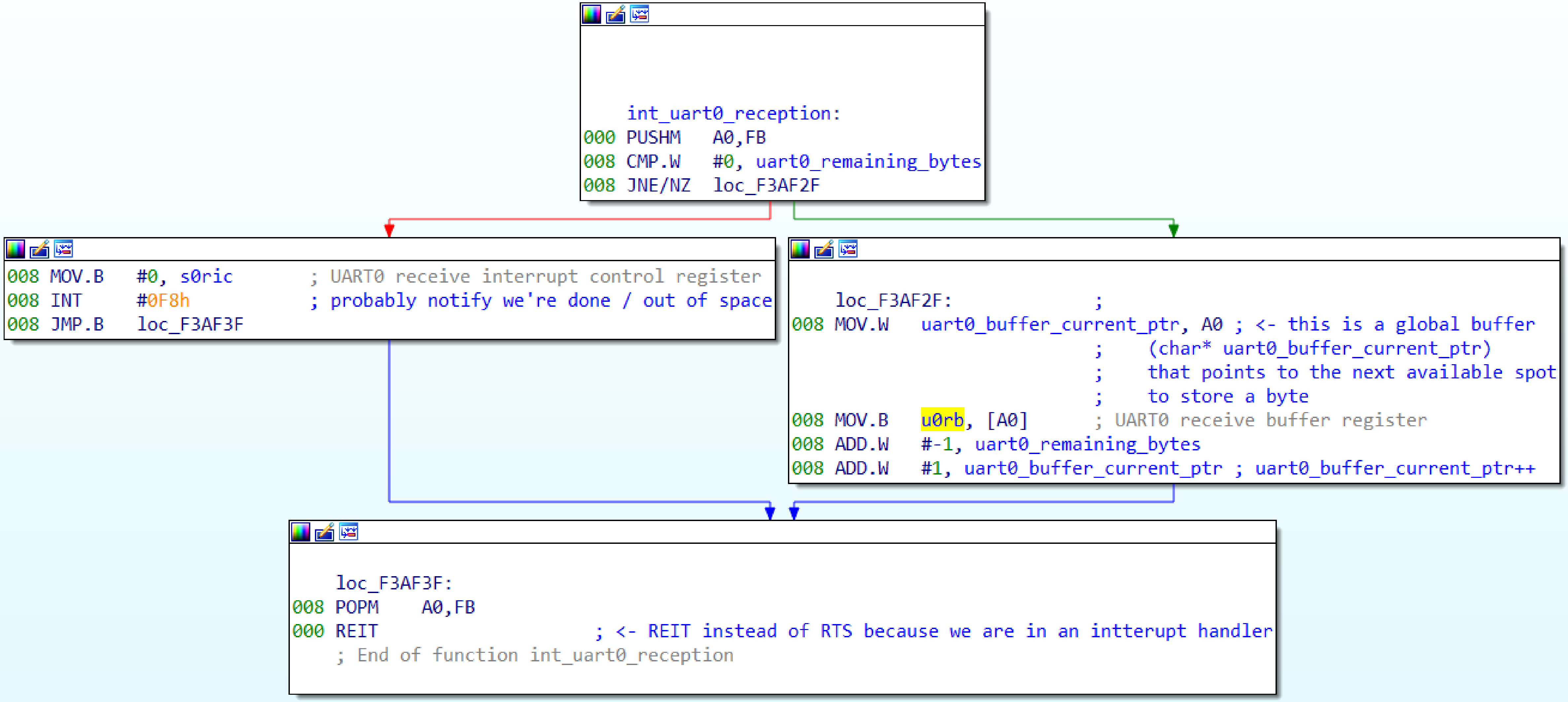
NOTES:



```
probably_syscall_table .LWORD 0FFFFFFFFh
                                         ; DATA XREF: app_from_flash_init+C'        ↑o
                                         ; ROM:00F306DD↑o
                                         ; see page 108 of hardware manual
                .LWORD 0FFFFFFFFh
                .LWORD 0FFFFFFFFh
                .LWORD 0FFFFFFFFh
                .LWORD 0FFFFFFFFh
                .LWORD 0FFFFFFFFh
                .LWORD 0FFFFFFFFh
                .LWORD 0FFFFFFFFh
                .LWORD int_DMA0          ; sw int 8
                .LWORD int_DMA1          ; sw int 9
                .LWORD 0FFFFFFFFh
                .LWORD 0FFFFFFFFh
                .LWORD 0FFFFFFFFh
                .LWORD 0FFFFFFFFh
                .LWORD 0FFFFFFFFh
                .LWORD 0FFFFFFFFh
                .LWORD 0FFFFFFFFh
                .LWORD int_uart0_transmission ; sw int 17
                .LWORD int_uart0_reception ; sw int 18
                .LWORD int_uart1_transmission_and_i2cNack ; sw
                .LWORD int_uart1_reception_andi2c_ack ; sw int
                .LWORD int_timer_B0      ; sw int 21
                .LWORD int_timer_B1      ; sw int 22
                .LWORD int_timer_B2      ; sw int 23
                .LWORD 0FFFFFFFFh
                .LWORD 0FFFFFFFFh
                .LWORD 0FFFFFFFFh
                .LWORD 0FFFFFFFFh
                .LWORD 0FFFFFFFFh
                .LWORD 0FFFFFFFFh
                .LWORD 0FFFFFFFFh
```



```
ROM:00FEFD94                .LWORD int_uart4_transmission ; sw int 37
ROM:00FEFD98                .LWORD int_uart4_reception ; sw int 38
ROM:00FEFD9C                .LWORD 0FFFFFFFFh
ROM:00FEFDA0                .LWORD 0FFFFFFFFh
ROM:00FEFDA4                .LWORD 0FFFFFFFFh
ROM:00FEFDA8                .LWORD 0FFFFFFFFh
ROM:00FEFDAC                .LWORD 0FFFFFFFFh
ROM:00FEFDB0                .LWORD 0FFFFFFFFh
ROM:00FEFDB4                .LWORD 0FFFFFFFFh
ROM:00FEFDB8                .LWORD 0FFFFFFFFh
ROM:00FEFDBC                .LWORD 0FFFFFFFFh
ROM:00FEFDC0                .LWORD 0FFFFFFFFh
ROM:00FEFDC4                .LWORD 0FFFFFFFFh
ROM:00FEFDC8                .LWORD 0FFFFFFFFh
ROM:00FEFDCC                .LWORD 0FFFFFFFFh
ROM:00FEFDD0                .LWORD 0FFFFFFFFh
ROM:00FEFDD4                .LWORD int_CAN_53        ; INT 53
ROM:00FEFDD8                .LWORD 0FFFFFFFFh
ROM:00FEFDDC                .LWORD 0FFFFFFFFh
ROM:00FEFDE0                .LWORD 0FFFFFFFFh
ROM:00FEFDE4                .LWORD 0FFFFFFFFh
ROM:00FEFDE8                .LWORD 0FFFFFFFFh
ROM:00FEFDEC                .LWORD 0FFFFFFFFh
ROM:00FEFDF0                .LWORD 0FFFFFFFFh
ROM:00FEFDF4                .LWORD 0FFFFFFFFh
ROM:00FEFDF8                .LWORD int_62            ; INT 62, called by int_DMA0, int_uart0_transmission, int_uart0_reception
ROM:00FEFDFC                .LWORD 0FFFFFFFFh
```

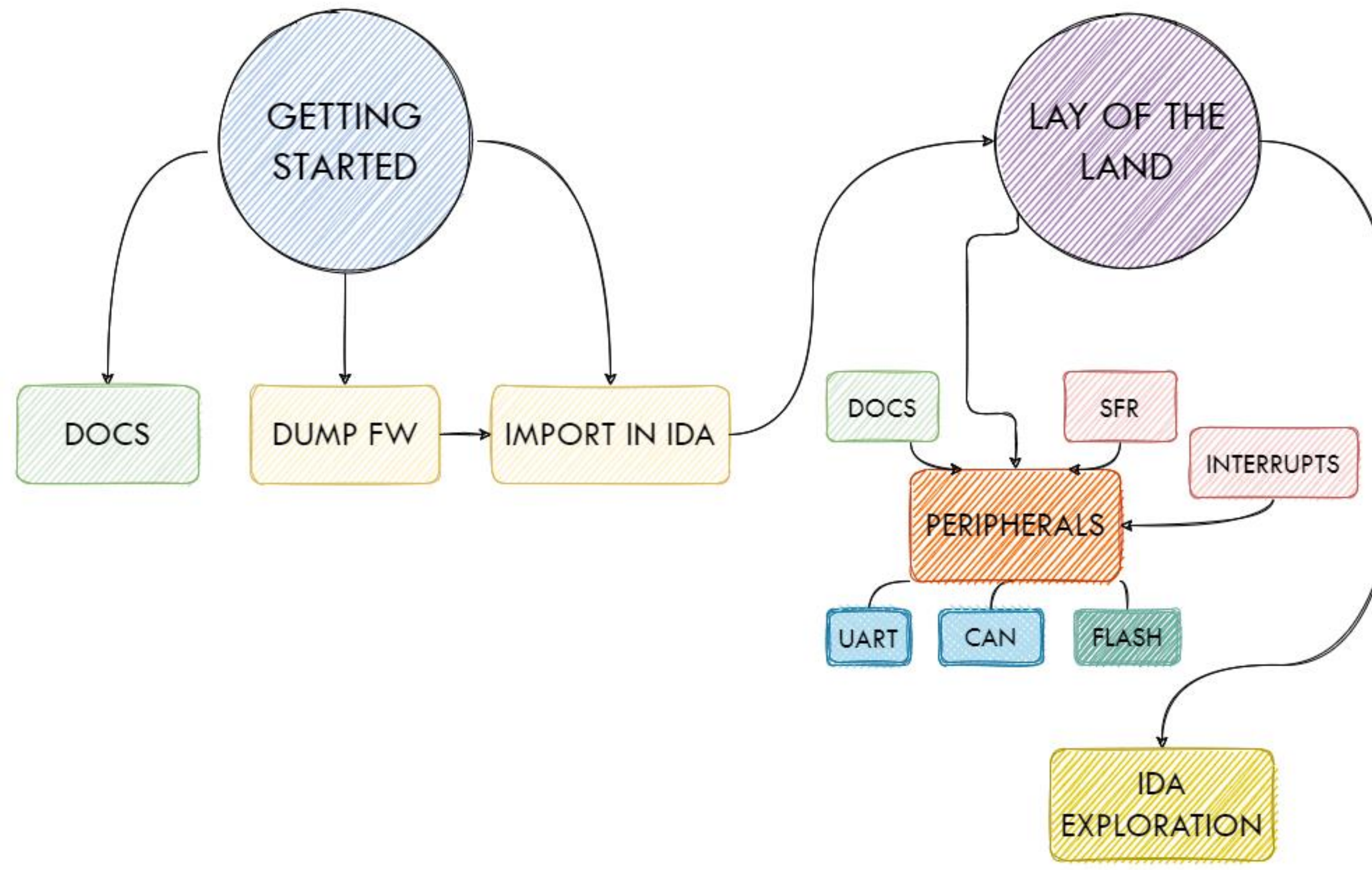


```
    int_uart0_reception:
000 PUSHM   A0,FB
008 CMP.W   #0, uart0_remaining_bytes
008 JNE/NZ  loc_F3AF2F
```

```
008 MOV.B    #0, s0ric            ; UART0 receive interrupt control register
008 INT      #0F8h                ; probably notify we're done / out of space
008 JMP.B    loc_F3AF3F
```

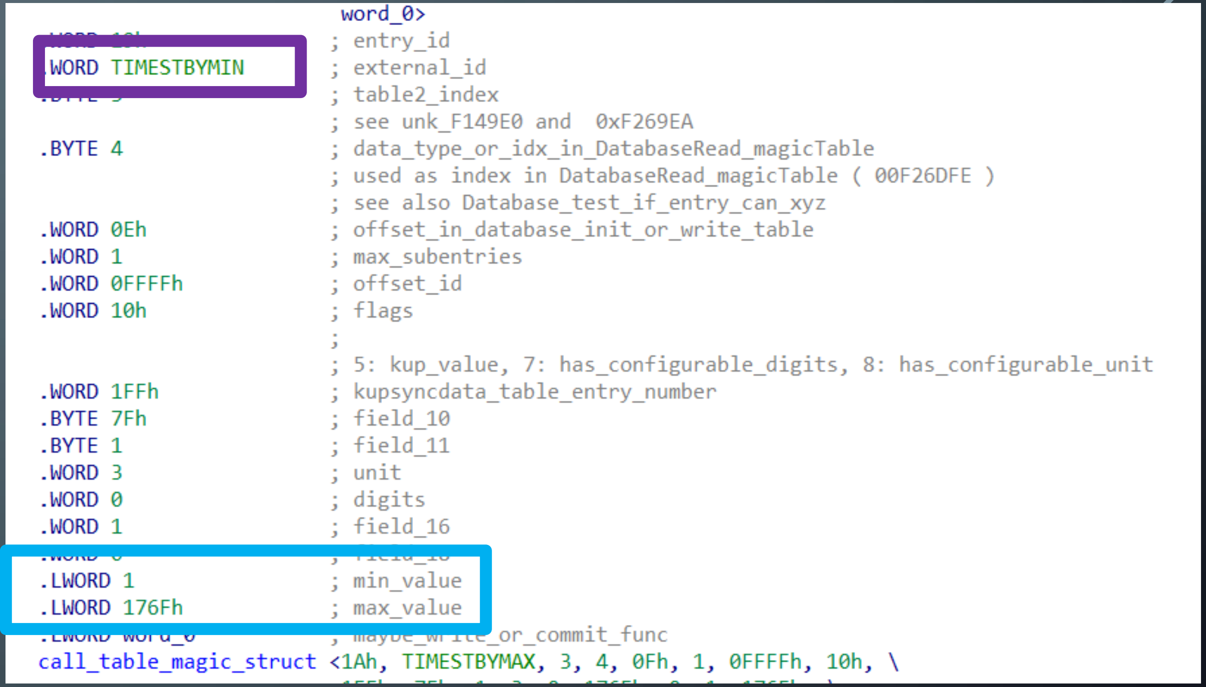
```
    loc_F3AF2F:                          ;
008 MOV.W    uart0_buffer_current_ptr, A0 ; <- this is a global buffer
                                         ;    (char* uart0_buffer_current_ptr)
                                         ;    that points to the next available spot
                                         ;    to store a byte
008 MOV.B    u0rb, [A0]                  ; UART0 receive buffer register
008 ADD.W    #-1, uart0_remaining_bytes
008 ADD.W    #1, uart0_buffer_current_ptr ; uart0_buffer_current_ptr++
```

```
    loc_F3AF3F:
008 POPM     A0,FB
000 REIT                         ; <- REIT instead of RTS because we are in an intterupt handler
    ; End of function int_uart0_reception
```

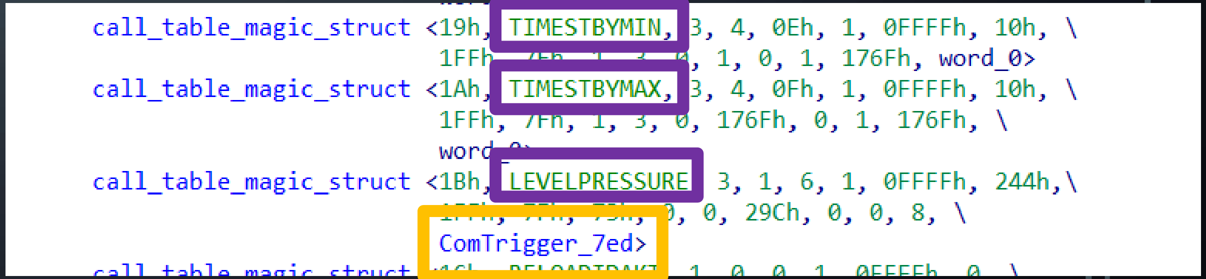

GETTING STARTED
DOCS
DUMP FW
IMPORT IN IDA
LAY OF THE LAND
DOCS
SFR
INTERRUPTS
PERIPHERALS
UART
CAN
FLASH
IDA EXPLORATION

INTERNAL DATABASE

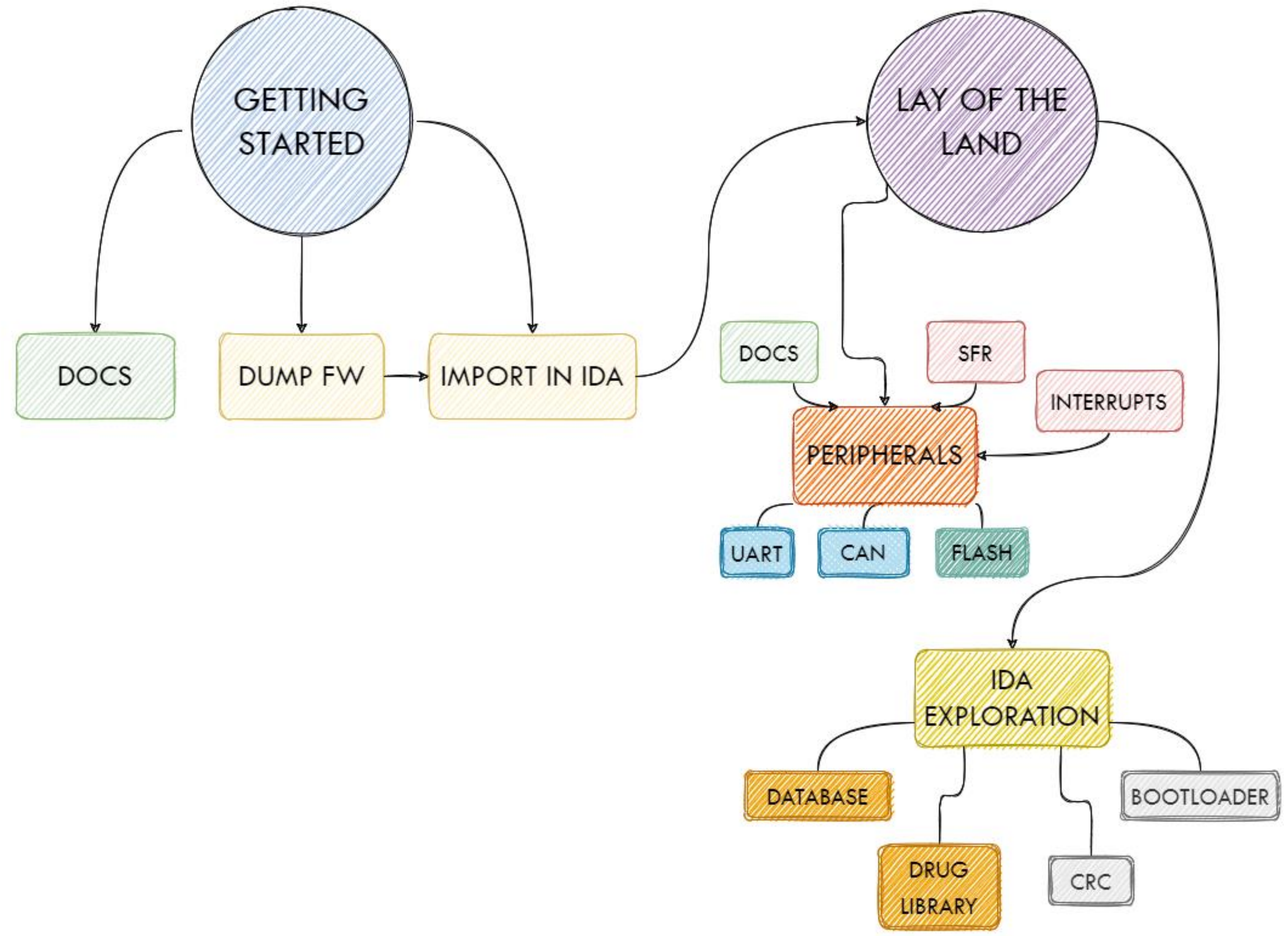
- Used for storing data and/or triggering actions
- Higher-level interface exposed over the CAN bus.
- Key names can be retrieved from auxiliary software
  → PCS binary
- Entry contains:
  - Types
  - Allowed Range
  - Unit
  - Precision
  - Callback Function
  - Etc.

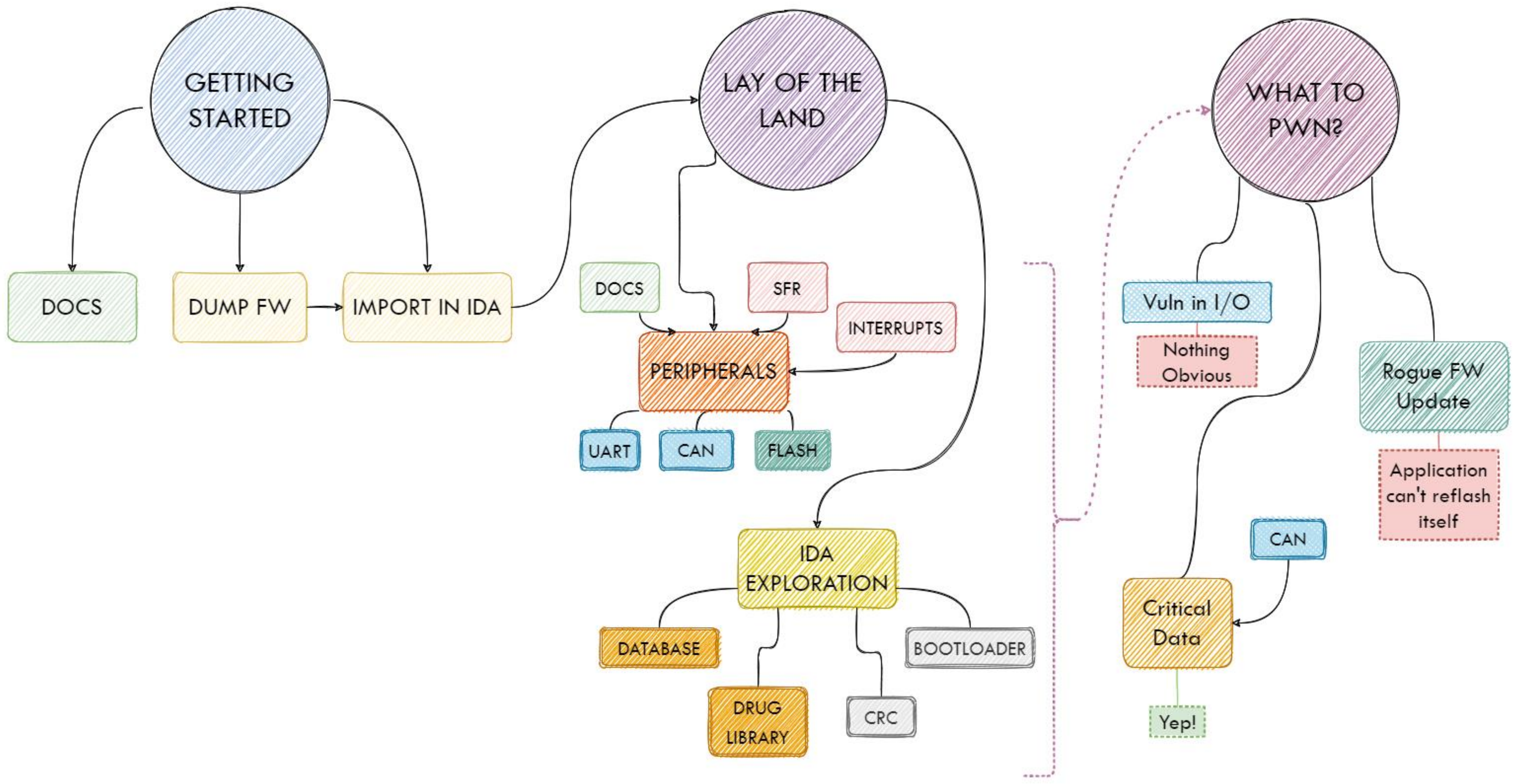


```
                          word_0>
[illegible]               ; entry_id
.WORD TIMESTBYMIN         ; external_id
[illegible]               ; table2_index
                          ; see unk_F149E0 and  0xF269EA
.BYTE 4                   ; data_type_or_idx_in_DatabaseRead_magicTable
                          ; used as index in DatabaseRead_magicTable ( 00F26DFE )
                          ; see also Database_test_if_entry_can_xyz
.WORD 0Eh                 ; offset_in_database_init_or_write_table
.WORD 1                   ; max_subentries
.WORD 0FFFFh              ; offset_id
.WORD 10h                 ; flags
                          ;
                          ; 5: kup_value, 7: has_configurable_digits, 8: has_configurable_unit
.WORD 1FFh                ; kupsyncdata_table_entry_number
.BYTE 7Fh                 ; field_10
.BYTE 1                   ; field_11
.WORD 3                   ; unit
.WORD 0                   ; digits
.WORD 1                   ; field_16
[illegible]               ; [illegible]
.LWORD 1                  ; min_value
.LWORD 176Fh              ; max_value
.LWORD word_0             ; maybe_write_or_commit_func
call_table_magic_struct <1Ah, TIMESTBYMAX, 3, 4, 0Fh, 1, 0FFFFh, 10h, \
```

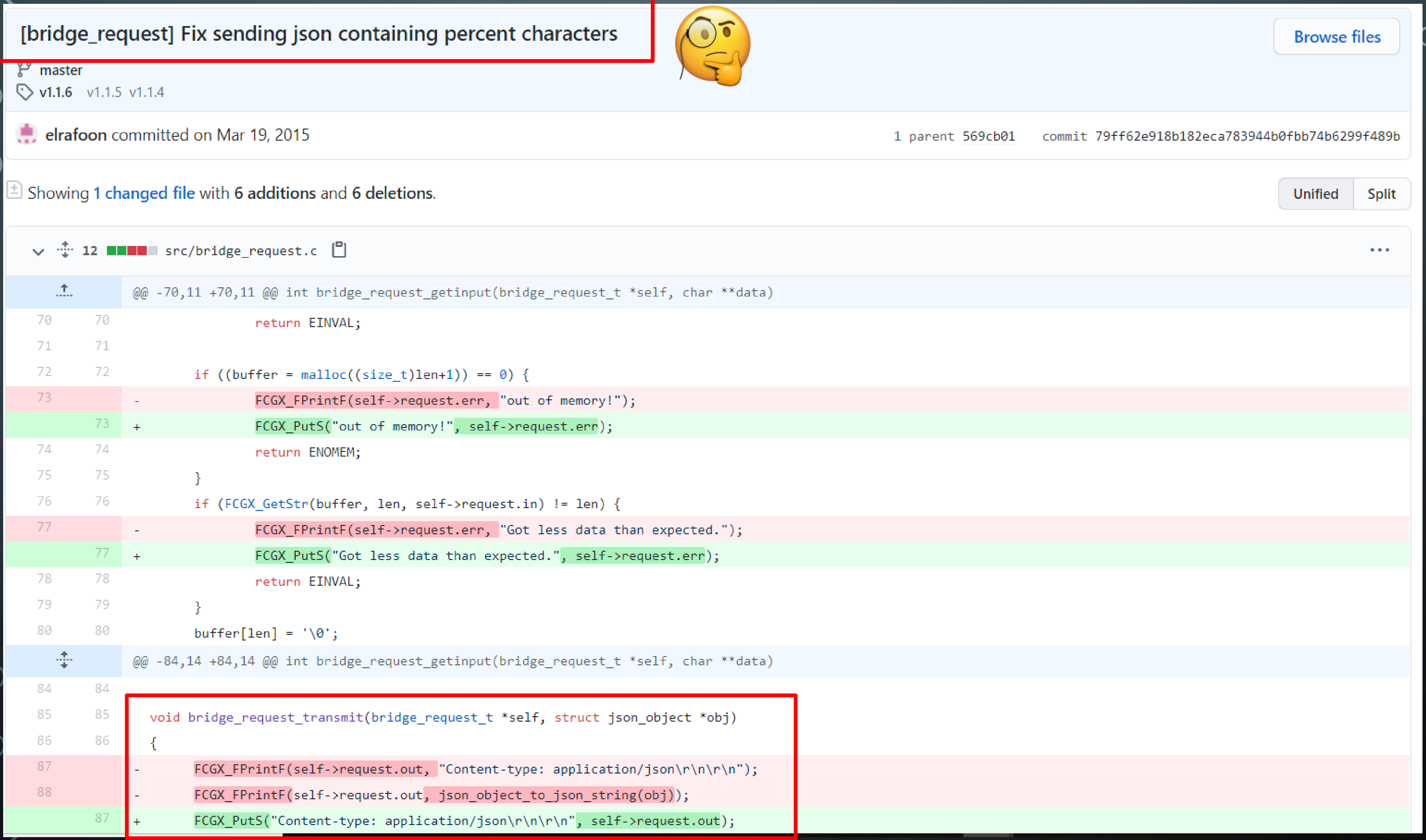


```
    call_table_magic_struct <19h, TIMESTBYMIN, 3, 4, 0Eh, 1, 0FFFFh, 10h, \
                             1FFh, [illegible], 1, 0, 1, 176Fh, word_0>
    call_table_magic_struct <1Ah, TIMESTBYMAX, 3, 4, 0Fh, 1, 0FFFFh, 10h, \
                             1FFh, 7Fh, 1, 3, 0, 176Fh, 0, 1, 176Fh, \
                             word_0>
    call_table_magic_struct <1Bh, LEVELPRESSURE 3, 1, 6, 1, 0FFFFh, 244h,\
                             1FFh, 7Fh, 73h, 0, 0, 29Ch, 0, 0, 8, \
                             ComTrigger_7ed>
    call_table_magic_struct <1Ch, [illegible], 1, 0, 0, 1, 0FFFFh, 0, \
```


GETTING STARTED
DOCS
DUMP FW
IMPORT IN IDA
LAY OF THE LAND
DOCS
SFR
INTERRUPTS
PERIPHERALS
UART
CAN
FLASH
IDA EXPLORATION
DATABASE
DRUG LIBRARY
CRC
BOOTLOADER


GETTING STARTED
DOCS
DUMP FW
IMPORT IN IDA
LAY OF THE LAND
DOCS
SFR
INTERRUPTS
PERIPHERALS
UART
CAN
FLASH
IDA EXPLORATION
DATABASE
DRUG LIBRARY
CRC
BOOTLOADER
WHAT TO PWN?
Vuln in I/O
Nothing Obvious
Rogue FW Update
Application can't reflash itself
CAN
Critical Data
Yep!

HUNTING FOR BUGS 🧐



[bridge_request] Fix sending json containing percent characters

elrafoon committed on Mar 19, 2015

1 parent 569cb01 commit 79ff62e918b182eca783944b0fbb74b6299f489b

Showing 1 changed file with 6 additions and 6 deletions.

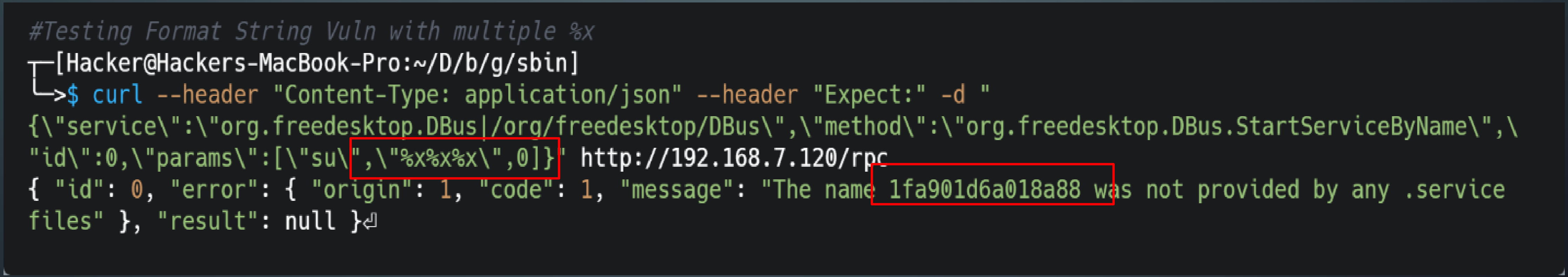
12 src/bridge_request.c

```
@@ -70,11 +70,11 @@ int bridge_request_getinput(bridge_request_t *self, char **data)
                return EINVAL;

        if ((buffer = malloc((size_t)len+1)) == 0) {
-               FCGX_FPrintF(self->request.err, "out of memory!");
+               FCGX_PutS("out of memory!", self->request.err);
                return ENOMEM;
        }
        if (FCGX_GetStr(buffer, len, self->request.in) != len) {
-               FCGX_FPrintF(self->request.err, "Got less data than expected.");
+               FCGX_PutS("Got less data than expected.", self->request.err);
                return EINVAL;
        }
        buffer[len] = '\0';
@@ -84,14 +84,14 @@ int bridge_request_getinput(bridge_request_t *self, char **data)

 void bridge_request_transmit(bridge_request_t *self, struct json_object *obj)
 {
-       FCGX_FPrintF(self->request.out, "Content-type: application/json\r\n\r\n");
-       FCGX_FPrintF(self->request.out, json_object_to_json_string(obj));
+       FCGX_PutS("Content-type: application/json\r\n\r\n", self->request.out);
```

FORMAT STRING EXPLOITATION



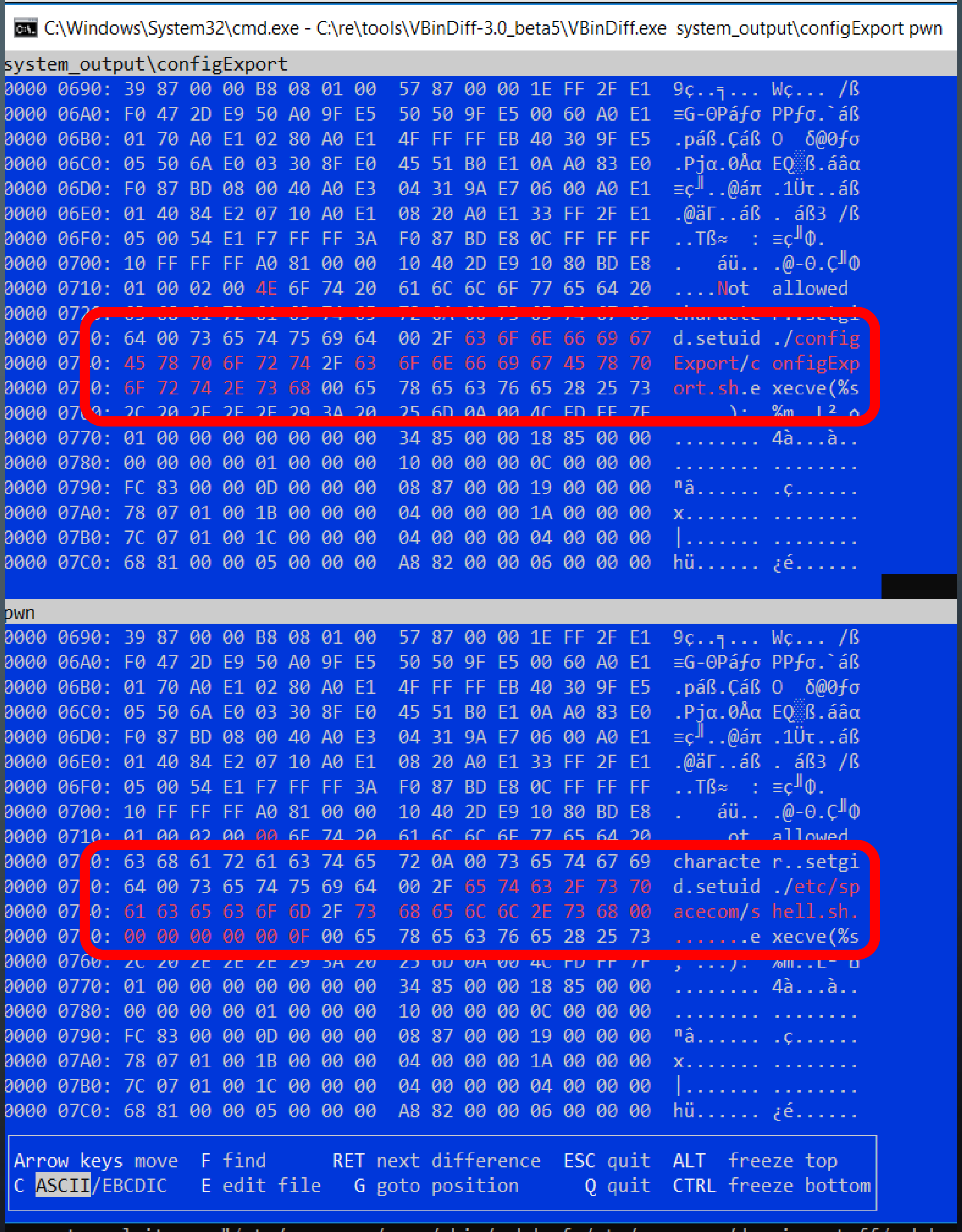
```
#Testing Format String Vuln with multiple %x
┌─[Hacker@Hackers-MacBook-Pro:~/D/b/g/sbin]
└─>$ curl --header "Content-Type: application/json" --header "Expect:" -d "
{\"service\":\"org.freedesktop.DBus|/org/freedesktop/DBus\",\"method\":\"org.freedesktop.DBus.StartServiceByName\",\
"id\":0,\"params\":[\"su\",\"%x%x%x\",0]}" http://192.168.7.120/rpc
{ "id": 0, "error": { "origin": 1, "code": 1, "message": "The name 1fa901d6a018a88 was not provided by any .service
files" }, "result": null }⏎
```

MANY MOONS LATER….

- Leak content of memory layout using %x
- Use %n to
  - Overwrite an address in the PLT with the address of system
  - Return to libc attack
- Gained user level access

PRIVILEGE ESCALATION

- Leveraging insecure .tar handling to create a privilege escalation (ManiMed)
- Binary patch configExport to run attacker script as root

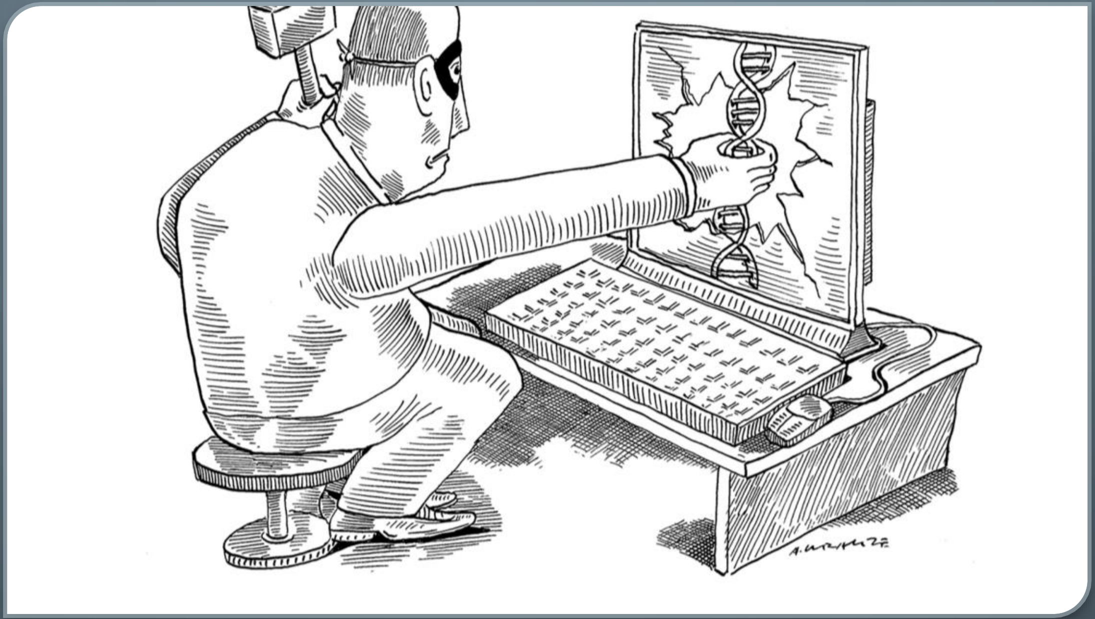


```
C:\Windows\System32\cmd.exe - C:\re\tools\VBinDiff-3.0_beta5\VBinDiff.exe  system_output\configExport pwn
system_output\configExport
0000 0690: 39 87 00 00 B8 08 01 00  57 87 00 00 1E FF 2F E1  9ç..╕... Wç... /ß
0000 06A0: F0 47 2D E9 50 A0 9F E5  50 50 9F E5 00 60 A0 E1  ≡G-ΘPáƒσ PPƒσ.`áß
0000 06B0: 01 70 A0 E1 02 80 A0 E1  4F FF FF EB 40 30 9F E5  .páß.Çáß O  δ@0ƒσ
0000 06C0: 05 50 6A E0 03 30 8F E0  45 51 B0 E1 0A A0 83 E0  .Pjα.0Åα EQ░ß.âα
0000 06D0: F0 87 BD 08 00 40 A0 E3  04 31 9A E7 06 00 A0 E1  ≡ç╜..@áπ .1Üτ..áß
0000 06E0: 01 40 84 E2 07 10 A0 E1  08 20 A0 E1 33 FF 2F E1  .@äΓ..áß . áß3 /ß
0000 06F0: 05 00 54 E1 F7 FF FF 3A  F0 87 BD E8 0C FF FF FF  ..Tß≈  : ≡ç╜Φ.
0000 0700: 10 FF FF FF A0 81 00 00  10 40 2D E9 10 80 BD E8  .   áü.. .@-Θ.Ç╜Φ
0000 0710: 01 00 02 00 4E 6F 74 20  61 6C 6C 6F 77 65 64 20  ....Not  allowed
0000 0720: 63 68 61 72 61 63 74 65  72 0A 00 73 65 74 67 69  characte r..setgi
0000 07 0: 64 00 73 65 74 75 69 64  00 2F 63 6F 6E 66 69 67  d.setuid ./config
0000 07 0: 45 78 70 6F 72 74 2F 63  6F 6E 66 69 67 45 78 70  Export/c onfigExp
0000 07 0: 6F 72 74 2E 73 68 00 65  78 65 63 76 65 28 25 73  ort.sh.e xecve(%s
0000 07 0: 2C 20 2E 2E 2E 29 3A 20  25 6D 0A 00 4C FD FF 7F  , ...):  %m..L² ⌂
0000 0770: 01 00 00 00 00 00 00 00  34 85 00 00 18 85 00 00  ........ 4à...à..
0000 0780: 00 00 00 00 01 00 00 00  10 00 00 00 0C 00 00 00  ........ ........
0000 0790: FC 83 00 00 0D 00 00 00  08 87 00 00 19 00 00 00  ⁿâ...... .ç......
0000 07A0: 78 07 01 00 1B 00 00 00  04 00 00 00 1A 00 00 00  x....... ........
0000 07B0: 7C 07 01 00 1C 00 00 00  04 00 00 00 04 00 00 00  |....... ........
0000 07C0: 68 81 00 00 05 00 00 00  A8 82 00 00 06 00 00 00  hü...... ¿é......
pwn
0000 0690: 39 87 00 00 B8 08 01 00  57 87 00 00 1E FF 2F E1  9ç..╕... Wç... /ß
0000 06A0: F0 47 2D E9 50 A0 9F E5  50 50 9F E5 00 60 A0 E1  ≡G-ΘPáƒσ PPƒσ.`áß
0000 06B0: 01 70 A0 E1 02 80 A0 E1  4F FF FF EB 40 30 9F E5  .páß.Çáß O  δ@0ƒσ
0000 06C0: 05 50 6A E0 03 30 8F E0  45 51 B0 E1 0A A0 83 E0  .Pjα.0Åα EQ░ß.âα
0000 06D0: F0 87 BD 08 00 40 A0 E3  04 31 9A E7 06 00 A0 E1  ≡ç╜..@áπ .1Üτ..áß
0000 06E0: 01 40 84 E2 07 10 A0 E1  08 20 A0 E1 33 FF 2F E1  .@äΓ..áß . áß3 /ß
0000 06F0: 05 00 54 E1 F7 FF FF 3A  F0 87 BD E8 0C FF FF FF  ..Tß≈  : ≡ç╜Φ.
0000 0700: 10 FF FF FF A0 81 00 00  10 40 2D E9 10 80 BD E8  .   áü.. .@-Θ.Ç╜Φ
0000 0710: 01 00 02 00 00 6F 74 20  61 6C 6C 6F 77 65 64 20  .....ot  allowed
0000 07 0: 63 68 61 72 61 63 74 65  72 0A 00 73 65 74 67 69  characte r..setgi
0000 07 0: 64 00 73 65 74 75 69 64  00 2F 65 74 63 2F 73 70  d.setuid ./etc/sp
0000 07 0: 61 63 65 63 6F 6D 2F 73  68 65 6C 6C 2E 73 68 00  acecom/s hell.sh.
0000 07 0: 00 00 00 00 00 0F 00 65  78 65 63 76 65 28 25 73  .......e xecve(%s
0000 0760: 2C 20 2E 2E 2E 29 3A 20  25 6D 0A 00 4C FD FF 7F  , ...):  %m..L² ⌂
0000 0770: 01 00 00 00 00 00 00 00  34 85 00 00 18 85 00 00  ........ 4à...à..
0000 0780: 00 00 00 00 01 00 00 00  10 00 00 00 0C 00 00 00  ........ ........
0000 0790: FC 83 00 00 0D 00 00 00  08 87 00 00 19 00 00 00  ⁿâ...... .ç......
0000 07A0: 78 07 01 00 1B 00 00 00  04 00 00 00 1A 00 00 00  x....... ........
0000 07B0: 7C 07 01 00 1C 00 00 00  04 00 00 00 04 00 00 00  |....... ........
0000 07C0: 68 81 00 00 05 00 00 00  A8 82 00 00 06 00 00 00  hü...... ¿é......

Arrow keys move   F find       RET next difference  ESC quit  ALT  freeze top
C ASCII/EBCDIC    E edit file   G goto position       Q quit  CTRL freeze bottom
```

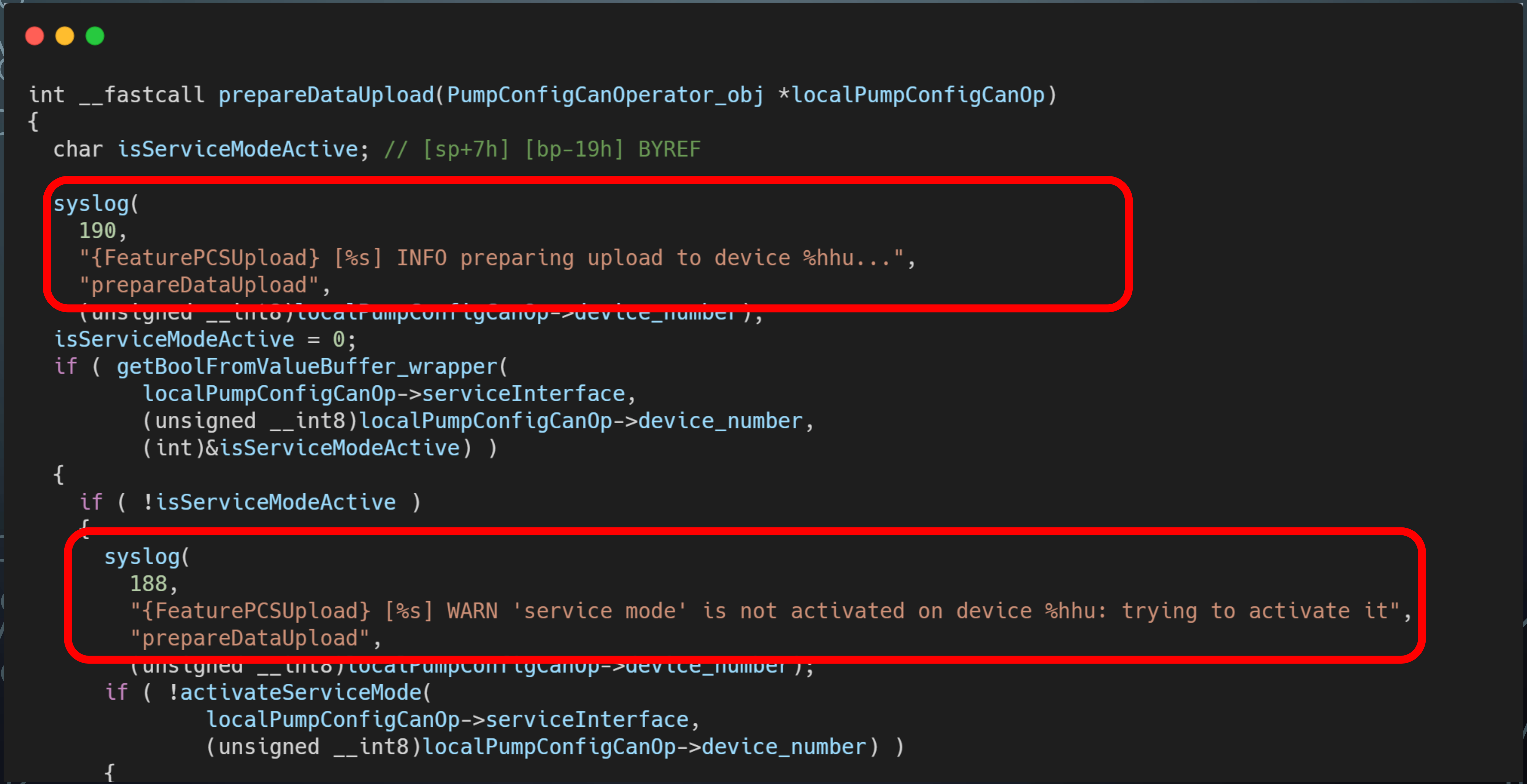
WE GOT ROOT, FINISHED?

- POC || GTFO
- How do we control the pump's critical OS with root access?
- Previous reports indicate root access could not cause "patient harm"
- Realistic attack scenario



HACKING THE PATIENT

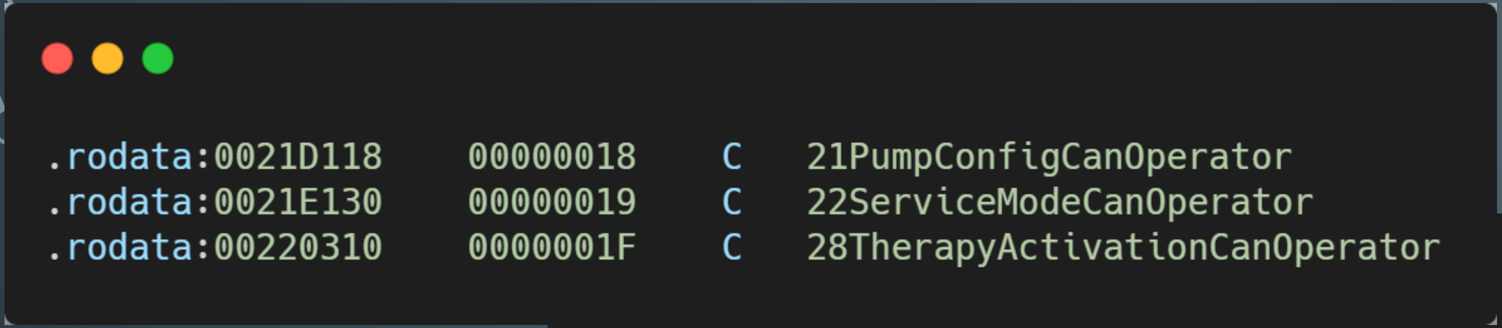
NO SYMBOLS, NO PROBLEM



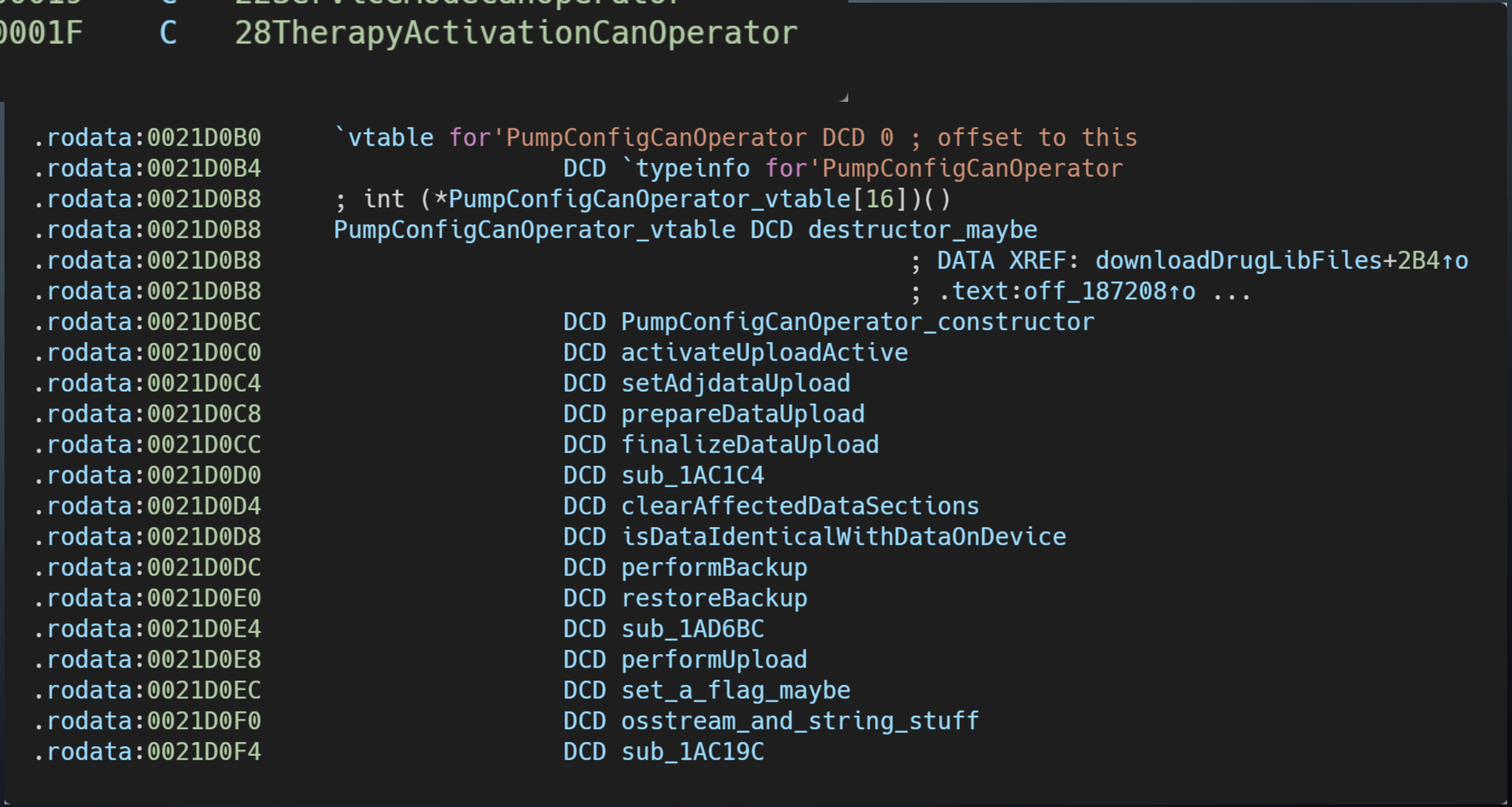
```
int __fastcall prepareDataUpload(PumpConfigCanOperator_obj *localPumpConfigCanOp)
{
  char isServiceModeActive; // [sp+7h] [bp-19h] BYREF

  syslog(
    190,
    "{FeaturePCSUpload} [%s] INFO preparing upload to device %hhu...",
    "prepareDataUpload",
    (unsigned __int8)localPumpConfigCanOp->device_number);
  isServiceModeActive = 0;
  if ( getBoolFromValueBuffer_wrapper(
         localPumpConfigCanOp->serviceInterface,
         (unsigned __int8)localPumpConfigCanOp->device_number,
         (int)&isServiceModeActive) )
  {
    if ( !isServiceModeActive )
    {
      syslog(
        188,
        "{FeaturePCSUpload} [%s] WARN 'service mode' is not activated on device %hhu: trying to activate it",
        "prepareDataUpload",
        (unsigned __int8)localPumpConfigCanOp->device_number);
      if ( !activateServiceMode(
              localPumpConfigCanOp->serviceInterface,
              (unsigned __int8)localPumpConfigCanOp->device_number) )
      {
```

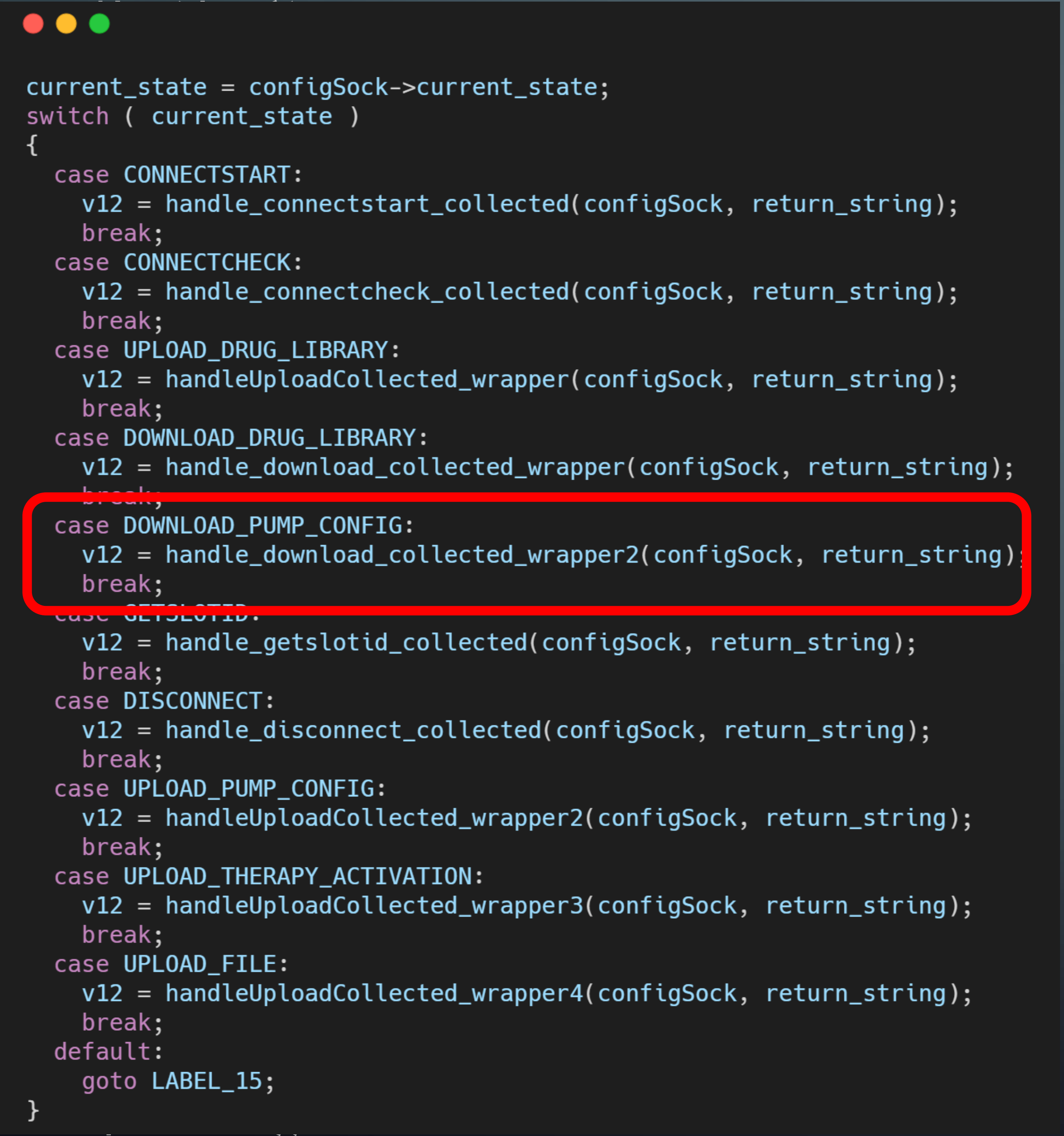
PCS CAN OPERATOR DATATYPES



```
.rodata:0021D118    00000018    C   21PumpConfigCanOperator
.rodata:0021E130    00000019    C   22ServiceModeCanOperator
.rodata:00220310    0000001F    C   28TherapyActivationCanOperator
```



```
.rodata:0021D0B0     `vtable for'PumpConfigCanOperator DCD 0 ; offset to this
.rodata:0021D0B4                     DCD `typeinfo for'PumpConfigCanOperator
.rodata:0021D0B8     ; int (*PumpConfigCanOperator_vtable[16])()
.rodata:0021D0B8     PumpConfigCanOperator_vtable DCD destructor_maybe
.rodata:0021D0B8                                         ; DATA XREF: downloadDrugLibFiles+2B4↑o
.rodata:0021D0B8                                         ; .text:off_187208↑o ...
.rodata:0021D0BC                     DCD PumpConfigCanOperator_constructor
.rodata:0021D0C0                     DCD activateUploadActive
.rodata:0021D0C4                     DCD setAdjdataUpload
.rodata:0021D0C8                     DCD prepareDataUpload
.rodata:0021D0CC                     DCD finalizeDataUpload
.rodata:0021D0D0                     DCD sub_1AC1C4
.rodata:0021D0D4                     DCD clearAffectedDataSections
.rodata:0021D0D8                     DCD isDataIdenticalWithDataOnDevice
.rodata:0021D0DC                     DCD performBackup
.rodata:0021D0E0                     DCD restoreBackup
.rodata:0021D0E4                     DCD sub_1AD6BC
.rodata:0021D0E8                     DCD performUpload
.rodata:0021D0EC                     DCD set_a_flag_maybe
.rodata:0021D0F0                     DCD osstream_and_string_stuff
.rodata:0021D0F4                     DCD sub_1AC19C
```



```
current_state = configSock->current_state;
switch ( current_state )
{
  case CONNECTSTART:
    v12 = handle_connectstart_collected(configSock, return_string);
    break;
  case CONNECTCHECK:
    v12 = handle_connectcheck_collected(configSock, return_string);
    break;
  case UPLOAD_DRUG_LIBRARY:
    v12 = handleUploadCollected_wrapper(configSock, return_string);
    break;
  case DOWNLOAD_DRUG_LIBRARY:
    v12 = handle_download_collected_wrapper(configSock, return_string);
    break;
  case DOWNLOAD_PUMP_CONFIG:
    v12 = handle_download_collected_wrapper2(configSock, return_string);
    break;
  case GETSLOTID:
    v12 = handle_getslotid_collected(configSock, return_string);
    break;
  case DISCONNECT:
    v12 = handle_disconnect_collected(configSock, return_string);
    break;
  case UPLOAD_PUMP_CONFIG:
    v12 = handleUploadCollected_wrapper2(configSock, return_string);
    break;
  case UPLOAD_THERAPY_ACTIVATION:
    v12 = handleUploadCollected_wrapper3(configSock, return_string);
    break;
  case UPLOAD_FILE:
    v12 = handleUploadCollected_wrapper4(configSock, return_string);
    break;
  default:
    goto LABEL_15;
}
```

BUILDING A CAN OPERATOR

- Cause canbus objects to be constructed
- DOWNLOAD commands don't require a file
- UPLOAD commands validate a file
- Pump config commands deal with critical pump settings

UNDERSTANDING THE CALL CHAIN

1. activateServiceMode
2. disableFlashProtection
3. performUpload/restoreBackup
    1. Triggered by sending DOWNLOAD_PUMP_CONFIG command
    2. Results in PumpConfigCanOperator
4. While we have data to write
    1. getPayloadBufferFromValue – helper function to prepare for setSvcData
    2. setSvcData – write data to flash chip
5. enableFlashProtection
6. deactivateServiceMode

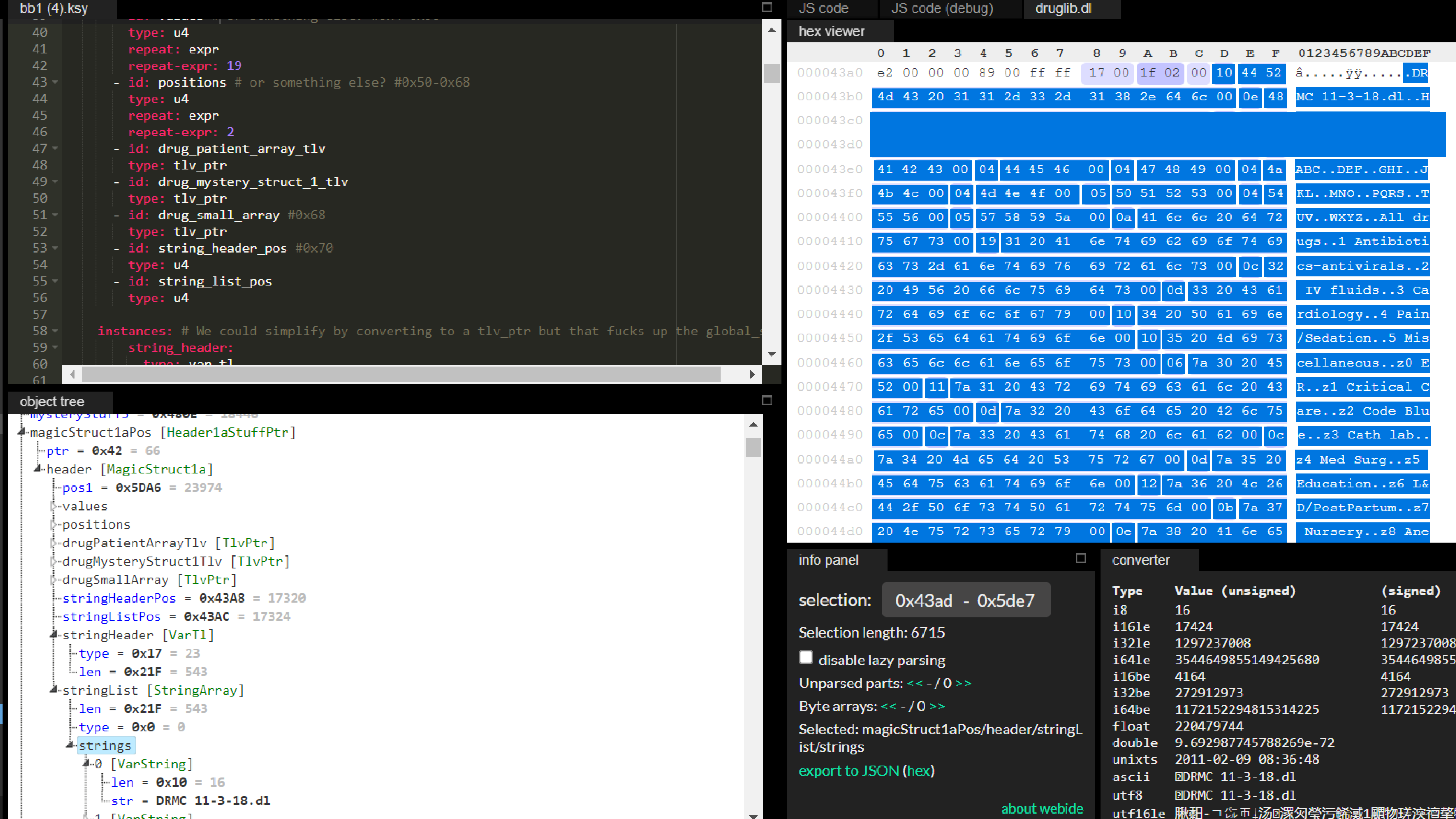
USING GDB FOR FUN AND PROFIT

- Install "malware" → GDB
- Attach to PCS
- Breakpoint to grab PumpConfigCanOperator
- Execute function call chain to modify data
- Profit! But what to write where??

CRITICAL DATA

WHAT DATA CAN WE MESS WITH?

- Drug Library
  - Safety net to avoid improper posology
  - Lots of data structure to reverse
  - Most of what we can tamper with is going to be shown on screen
- Calibration Data + Disposable Data
  - Internal parameters on how the device and disposable (tubes) operate
  - Servicing information → invisible to end user (nurse/doctor)



bb1 (4).ksy

```
        type: u4
        repeat: expr
        repeat-expr: 19
      - id: positions # or something else? #0x50-0x68
        type: u4
        repeat: expr
        repeat-expr: 2
      - id: drug_patient_array_tlv
        type: tlv_ptr
      - id: drug_mystery_struct_1_tlv
        type: tlv_ptr
      - id: drug_small_array #0x68
        type: tlv_ptr
      - id: string_header_pos #0x70
        type: u4
      - id: string_list_pos
        type: u4

  instances: # We could simplify by converting to a tlv_ptr but that fucks up the global_
      string_header:
```

object tree

```
magicStruct1aPos [Header1aStuffPtr]
  ptr = 0x42 = 66
  header [MagicStruct1a]
    pos1 = 0x5DA6 = 23974
    values
    positions
    drugPatientArrayTlv [TlvPtr]
    drugMysteryStruct1Tlv [TlvPtr]
    drugSmallArray [TlvPtr]
    stringHeaderPos = 0x43A8 = 17320
    stringListPos = 0x43AC = 17324
    stringHeader [VarTl]
      type = 0x17 = 23
      len = 0x21F = 543
    stringList [StringArray]
      len = 0x21F = 543
      type = 0x0 = 0
      strings
        0 [VarString]
          len = 0x10 = 16
          str = DRMC 11-3-18.dl
```

JS code | JS code (debug) | druglib.dl

hex viewer

```
          0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F  0123456789ABCDEF
000043a0  e2 00 00 00 89 00 ff ff  17 00 1f 02 00 10 44 52  â.....ÿÿ......DR
000043b0  4d 43 20 31 31 2d 33 2d  31 38 2e 64 6c 00 0e 48  MC 11-3-18.dl..H
000043c0  [redacted]
000043d0
000043e0  41 42 43 00 04 44 45 46  00 04 47 48 49 00 04 4a  ABC..DEF..GHI..J
000043f0  4b 4c 00 04 4d 4e 4f 00  05 50 51 52 53 00 04 54  KL..MNO..PQRS..T
00004400  55 56 00 05 57 58 59 5a  00 0a 41 6c 6c 20 64 72  UV..WXYZ..All dr
00004410  75 67 73 00 19 31 20 41  6e 74 69 62 69 6f 74 69  ugs..1 Antibioti
00004420  63 73 2d 61 6e 74 69 76  69 72 61 6c 73 00 0c 32  cs-antivirals..2
00004430  20 49 56 20 66 6c 75 69  64 73 00 0d 33 20 43 61   IV fluids..3 Ca
00004440  72 64 69 6f 6c 6f 67 79  00 10 34 20 50 61 69 6e  rdiology..4 Pain
00004450  2f 53 65 64 61 74 69 6f  6e 00 10 35 20 4d 69 73  /Sedation..5 Mis
00004460  63 65 6c 6c 61 6e 65 6f  75 73 00 06 7a 30 20 45  cellaneous..z0 E
00004470  52 00 11 7a 31 20 43 72  69 74 69 63 61 6c 20 43  R..z1 Critical C
00004480  61 72 65 00 0d 7a 32 20  43 6f 64 65 20 42 6c 75  are..z2 Code Blu
00004490  65 00 0c 7a 33 20 43 61  74 68 20 6c 61 62 00 0c  e..z3 Cath lab..
000044a0  7a 34 20 4d 65 64 20 53  75 72 67 00 0d 7a 35 20  z4 Med Surg..z5
000044b0  45 64 75 63 61 74 69 6f  6e 00 12 7a 36 20 4c 26  Education..z6 L&
000044c0  44 2f 50 6f 73 74 50 61  72 74 75 6d 00 0b 7a 37  D/PostPartum..z7
000044d0  20 4e 75 72 73 65 72 79  00 0e 7a 38 20 41 6e 65   Nursery..z8 Ane
```

info panel

selection: 0x43ad - 0x5de7

Selection length: 6715

☐ disable lazy parsing

Unparsed parts: << - / 0 >>

Byte arrays: << - / 0 >>

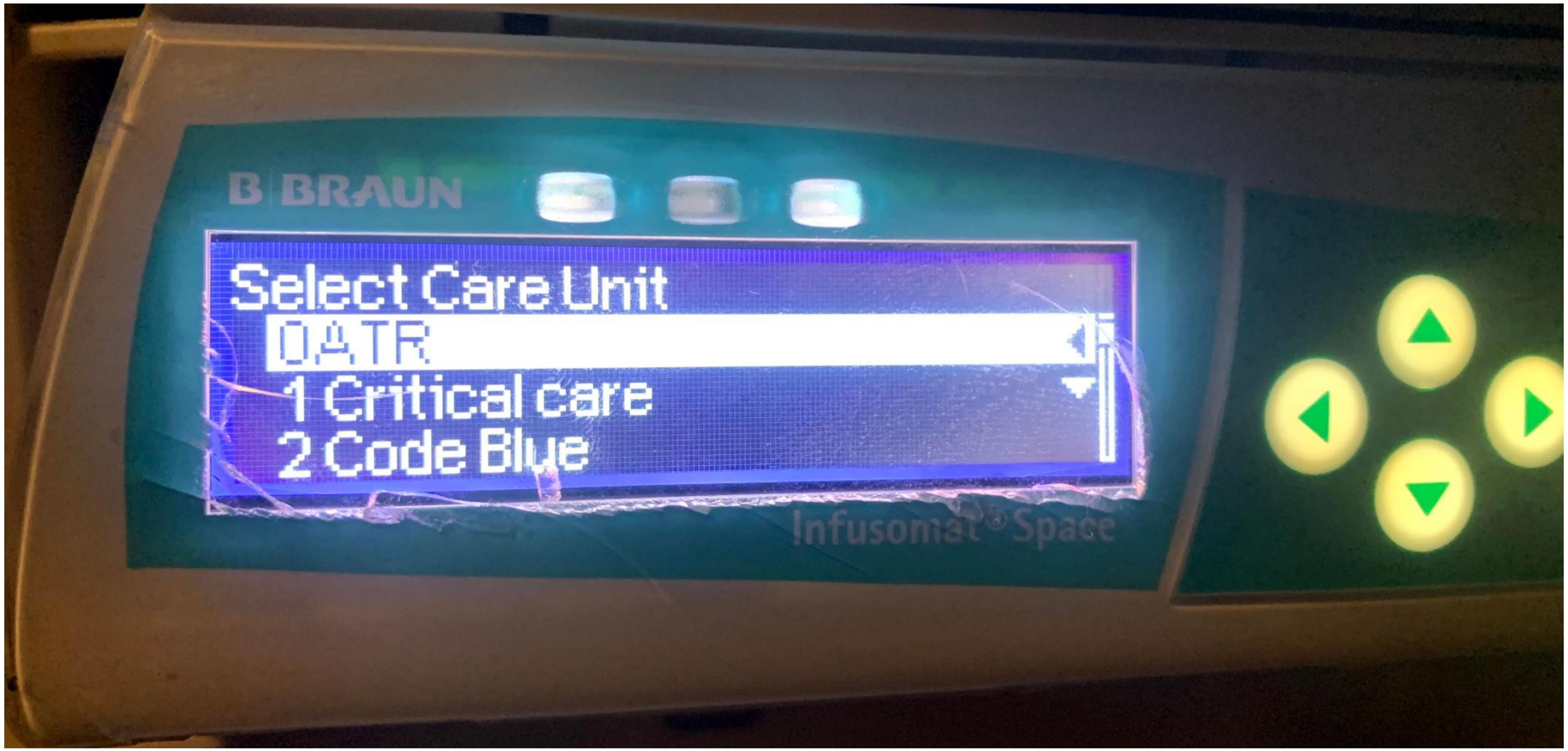
Selected: magicStruct1aPos/header/stringList/strings

export to JSON (hex)

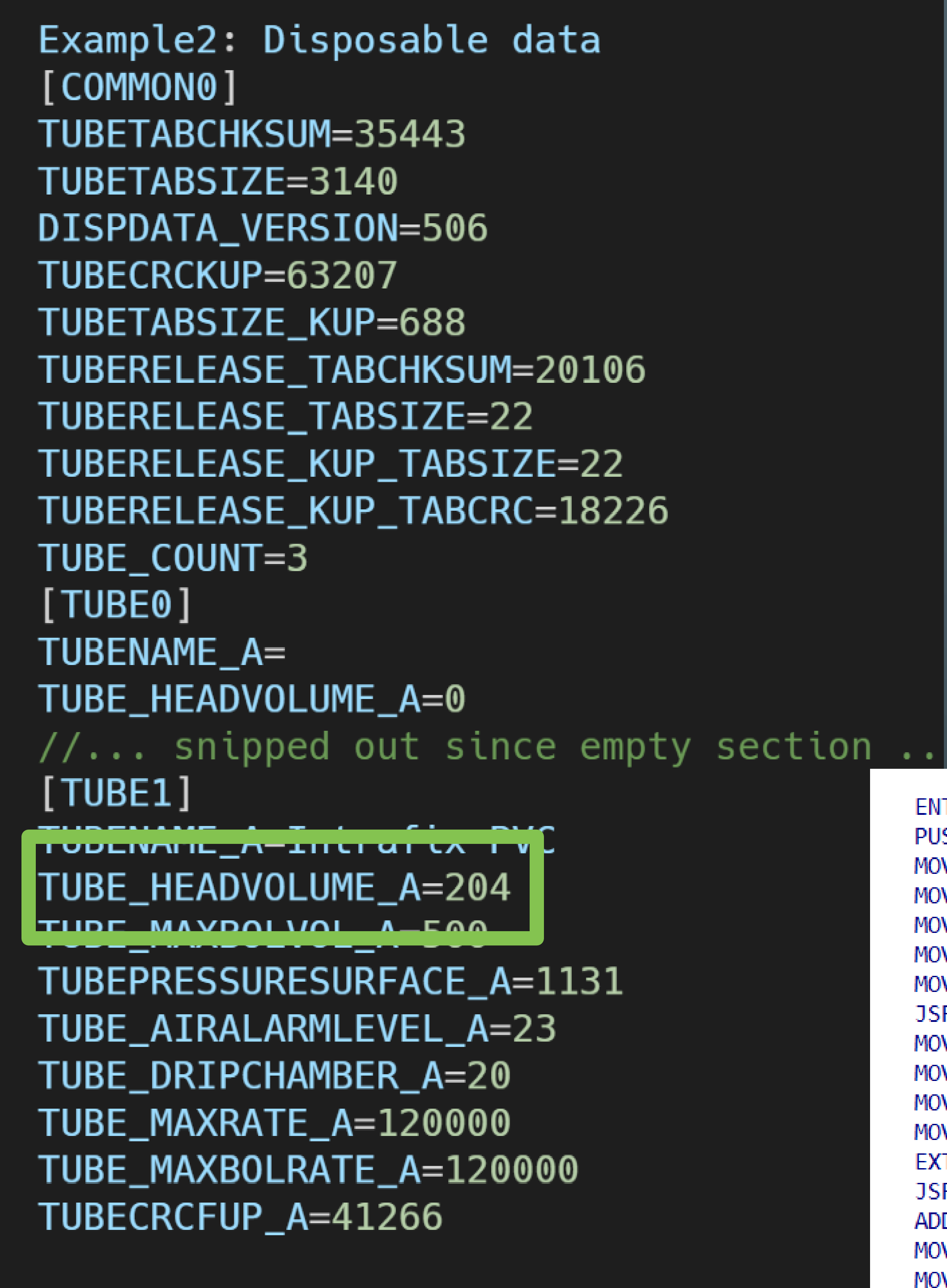
about webide

converter

```
Type     Value (unsigned)          (signed)
i8       16                        16
i16le    17424                     17424
i32le    1297237008                1297237008
i64le    3544649855149425680       3544649855
i16be    4164                      4164
i32be    272912973                 272912973
i64be    1172152294815314225       1172152294
float    220479744
double   9.692987745788269e-72
unixts   2011-02-09 08:36:48
ascii    DRMC 11-3-18.dl
utf8     DRMC 11-3-18.dl
```



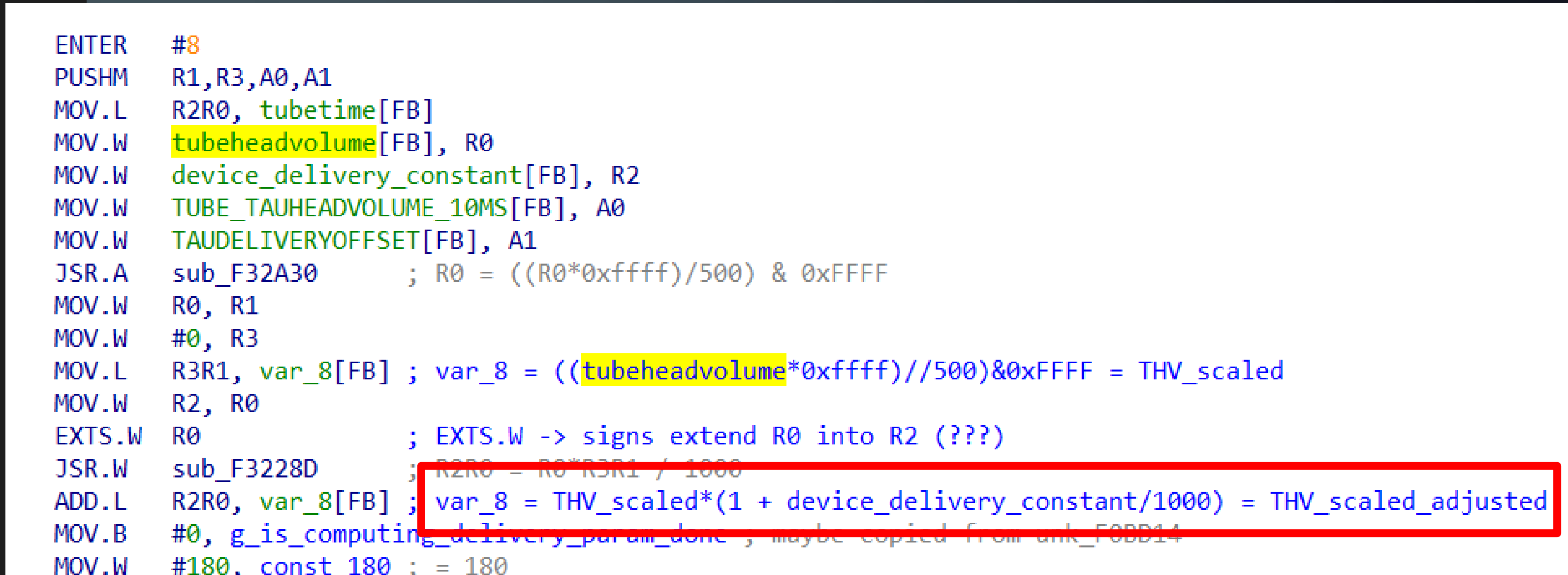
Select Care Unit
0 ATR
1 Critical care
2 Code Blue



```
Example2: Disposable data
[COMMON0]
TUBETABCHKSUM=35443
TUBETABSIZE=3140
DISPDATA_VERSION=506
TUBECRCKUP=63207
TUBETABSIZE_KUP=688
TUBERELEASE_TABCHKSUM=20106
TUBERELEASE_TABSIZE=22
TUBERELEASE_KUP_TABSIZE=22
TUBERELEASE_KUP_TABCRC=18226
TUBE_COUNT=3
[TUBE0]
TUBENAME_A=
TUBE_HEADVOLUME_A=0
//... snipped out since empty section ..
[TUBE1]
TUBENAME_A=Intrafix PVC
TUBE_HEADVOLUME_A=204
TUBE_MAXBOLVOL_A=500
TUBEPRESSURESURFACE_A=1131
TUBE_AIRALARMLEVEL_A=23
TUBE_DRIPCHAMBER_A=20
TUBE_MAXRATE_A=120000
TUBE_MAXBOLRATE_A=120000
TUBECRCFUP_A=41266
```

DISPOSABLE DATA

- Allow the pump to handle different infusion tubes (different volume, rates, …)
- TUBE_HEADVOLUME
  - "Amount" of drug per squeeze of the pump
  - Used for calculations
    - Wrong value → incorrect volume estimation → over/under delivery



```
ENTER    #8
PUSHM    R1,R3,A0,A1
MOV.L    R2R0, tubetime[FB]
MOV.W    tubeheadvolume[FB], R0
MOV.W    device_delivery_constant[FB], R2
MOV.W    TUBE_TAUHEADVOLUME_10MS[FB], A0
MOV.W    TAUDELIVERYOFFSET[FB], A1
JSR.A    sub_F32A30       ; R0 = ((R0*0xffff)/500) & 0xFFFF
MOV.W    R0, R1
MOV.W    #0, R3
MOV.L    R3R1, var_8[FB] ; var_8 = ((tubeheadvolume*0xffff)//500)&0xFFFF = THV_scaled
MOV.W    R2, R0
EXTS.W   R0               ; EXTS.W -> signs extend R0 into R2 (???)
JSR.W    sub_F3228D       ; R2R0 = R0*R3R1 / 1000
ADD.L    R2R0, var_8[FB] ; var_8 = THV_scaled*(1 + device_delivery_constant/1000) = THV_scaled_adjusted
MOV.B    #0, g_is_computing_delivery_param_done ; maybe copied from unk_F0BD14
MOV.W    #180, const_180 ; = 180
```

MODIFYING CRITICAL DATA

- What?

  Disposable Data → TUBE_HEADVOLUME
- Where?

  Flash Memory → Internal Database
- How?
  - gdb + PCS → SetSvcData → CAN messages
- Requirements
  - Must account for multiple CRCs
  - First erase existing disposable data

PUTTING IT ALL TOGETHER

IMPACT

- Manipulating medication dosage can be fatal
- Ransomware
    - FBI reported $61 million earned in 21months
- University of Vermont Medical Center
    - October 28th 2020
    - 75% of active chemotherapy patients being turned

"Something as routine as correcting a person's high blood sugar or sodium level too quickly can cause the brain to swell or damage the nerves which can lead to permanent disability or even death."

Dr. Shaun Nordeck

MEDICAL INDUSTRY COMMON PITFALLS

- Device lifecycle
- Patching is costly
- Designed for safety rather than security
- Everything is trusted
- CAN gets connected to WIFI
- Technical debt

TL;DR

- 5 CVEs discovered (highest a CVSS 9.7)
- Remotely compromised a B.Braun Infusion Pump
- Exploitation can lead to overdosing
- Infusion pumps are popular →$54B in sales worldwide
- Mitigations for medical devices are hard
- Worldwide hacking collaboration is fun!

QUESTIONS?

THANK YOU!!!

@FULMETALPACKETS

@PHLAUL

HTTPS://WWW.MCAFEE.COM/ENTERPRISE/EN-US/THREAT-CENTER/ADVANCED-THREAT-RESEARCH.HTML