LIBERATING BLUETOOTH ON THE ESP32

INTRODUCTION

- Who am I?
- Why?
- Why the ESP32?

WHO AM I?

- Antonio Vázquez Blanco (Antón)
- @antoniovazquezblanco@mastodon.social
- @antonvblanco
- antonio.vazquez@tarlogic.com

- R&D @ 

WHY?

- Ongoing Bluetooth security research line

WHY?

- Ongoing Bluetooth security research line
- First steps: Isolated security findings (BlueTrust, BlueSpy…)

WHY?

- Ongoing Bluetooth security research line
- First steps: Isolated security findings (BlueTrust, BlueSpy…)
- Standardization: Bluetooth Security Assessment Methodology

WHY?

- Ongoing Bluetooth security research line
- First steps: Isolated security findings (BlueTrust, BlueSpy…)
- Standardization: Bluetooth Security Assessment Methodology
- Now: Tooling! (UsbBluetooth, Scapy…)

WHY?

- Ongoing Bluetooth security research line
- First steps: Isolated security findings (BlueTrust, BlueSpy…)
- Standardization: Bluetooth Security Assessment Methodology
- Now: Tooling! (UsbBluetooth, Scapy…)
- But still lack affordable low-level access capabilities…

WHY?

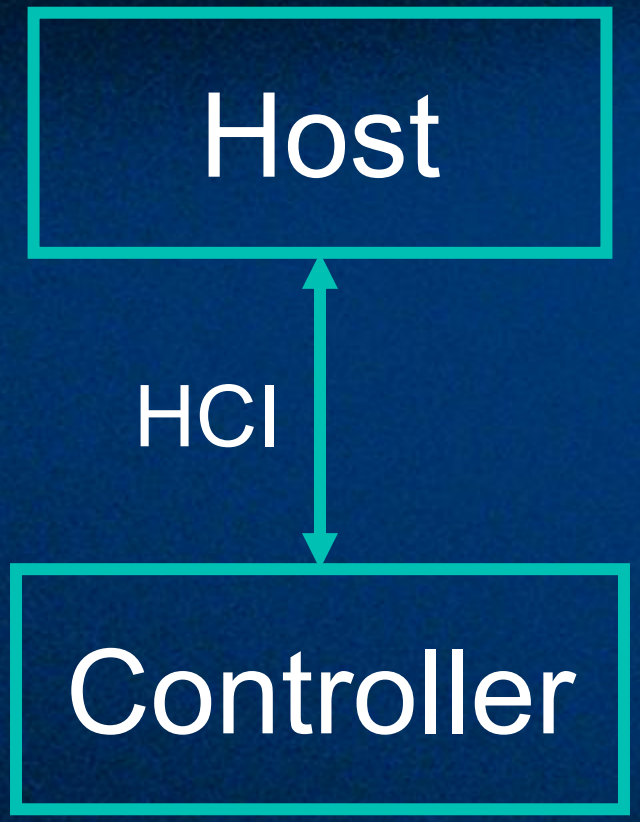
General purpose

Host

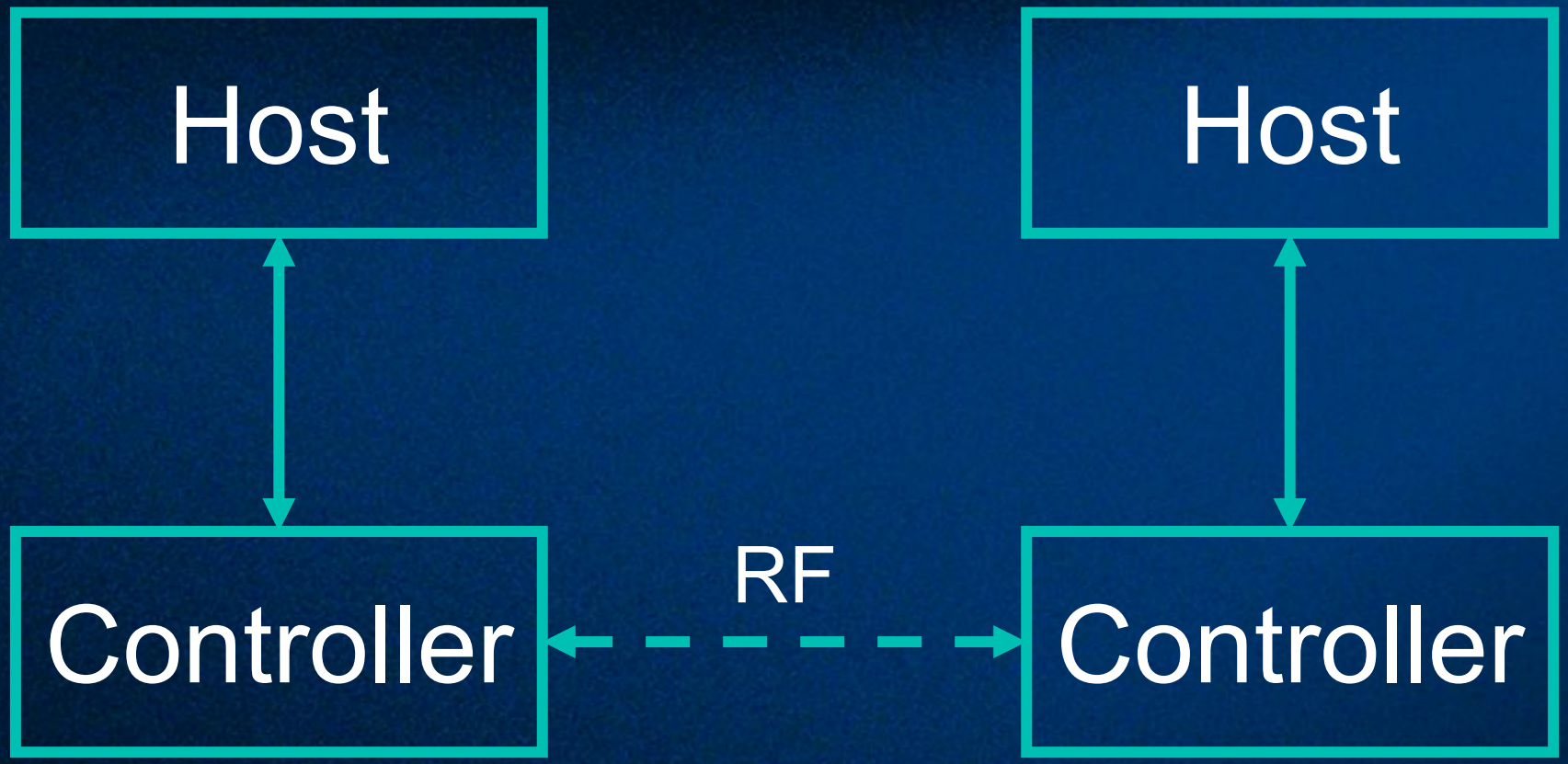
Specialized

Controller

WHY?



WHY?



WHY THE ESP32?

- Market penetration & availability
- Very low cost!
- Supports both BR/EDR and LE!
- Almost all the SDK is already open source
- The remaining closed source blobs are Apache licensed!

REVERSING

- Information compilation
- Techniques
- Tooling

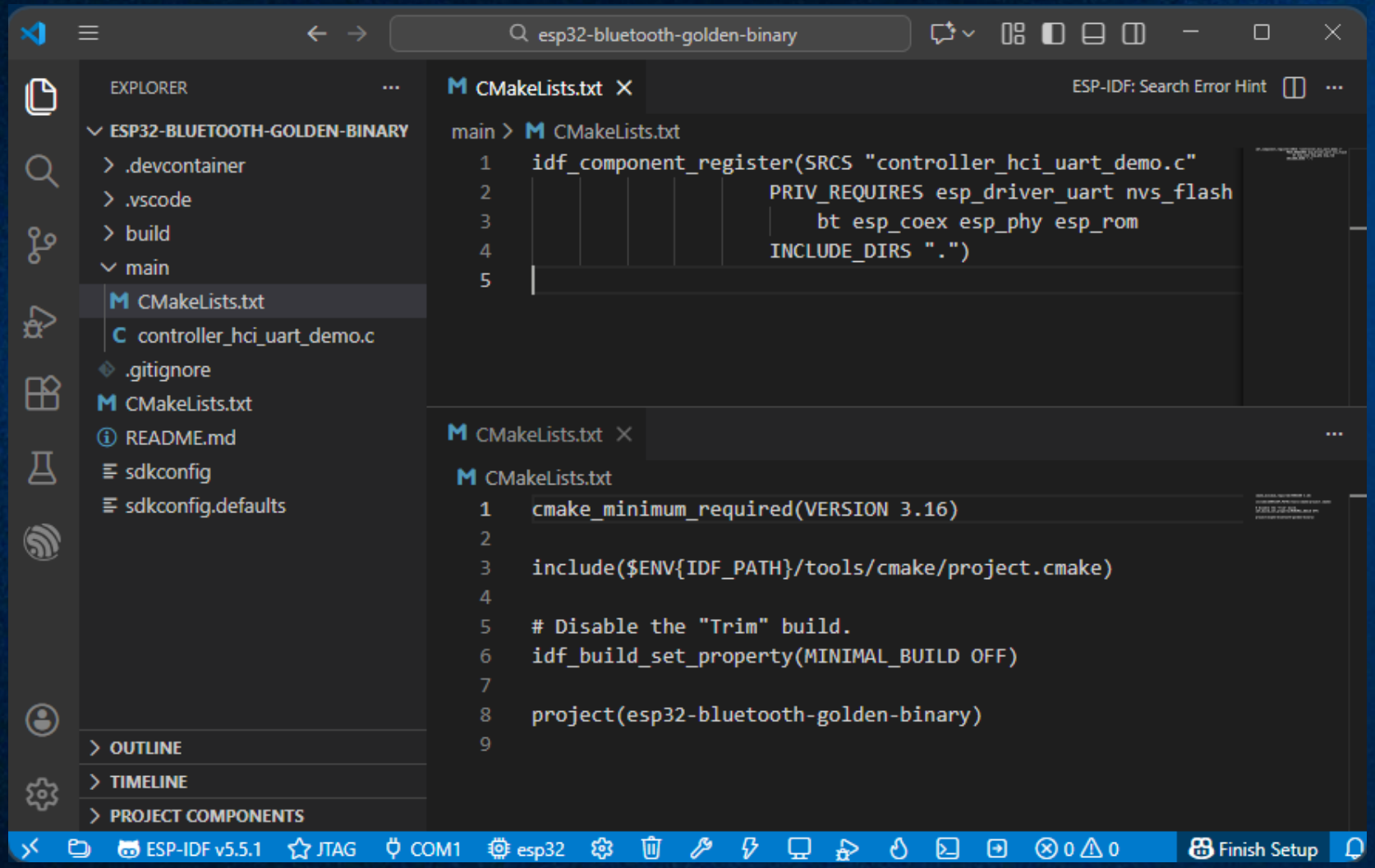
BINARIES

- Espressif publishes a lot of information…
- ESP32 ROMs: https://github.com/espressif/esp-rom-elfs/releases
- Bluetooth libs: https://github.com/espressif/esp32-bt-lib/
- RF PHY libs: https://github.com/espressif/esp-phy-lib/
- Not easy to reverse separately…

GOLDEN BINARY

- Custom ESP-IDF project
- Links components: bt, esp_coex, esp_phy, esp_rom
- Disables “trim”: idf_build_set_property(MINIMAL_BUILD OFF)
- Warning! Results are dependendent on IDF version!
- Initial “ELF” with symbols!

GOLDEN BINARY



LINKER SCRIPTS

- There are many linker scripts with symbol names in the SDK…
- ESP LD: https://github.com/espressif/esp-idf/tree/master/components/esp_rom/esp32/ld
- GhidraLinkerScript: https://github.com/antoniovazquezblanco/GhidraLinkerScript

LINKER SCRIPTS

master

esp-idf / components / esp_rom / esp32 / ld / esp32.rom.ld

Go to file t

BetterJincheng and boblane1 fix(bt/controller): Fixed some controller bugs

b607198 · last month

1658 lines (1651 loc) · 74.3 KB

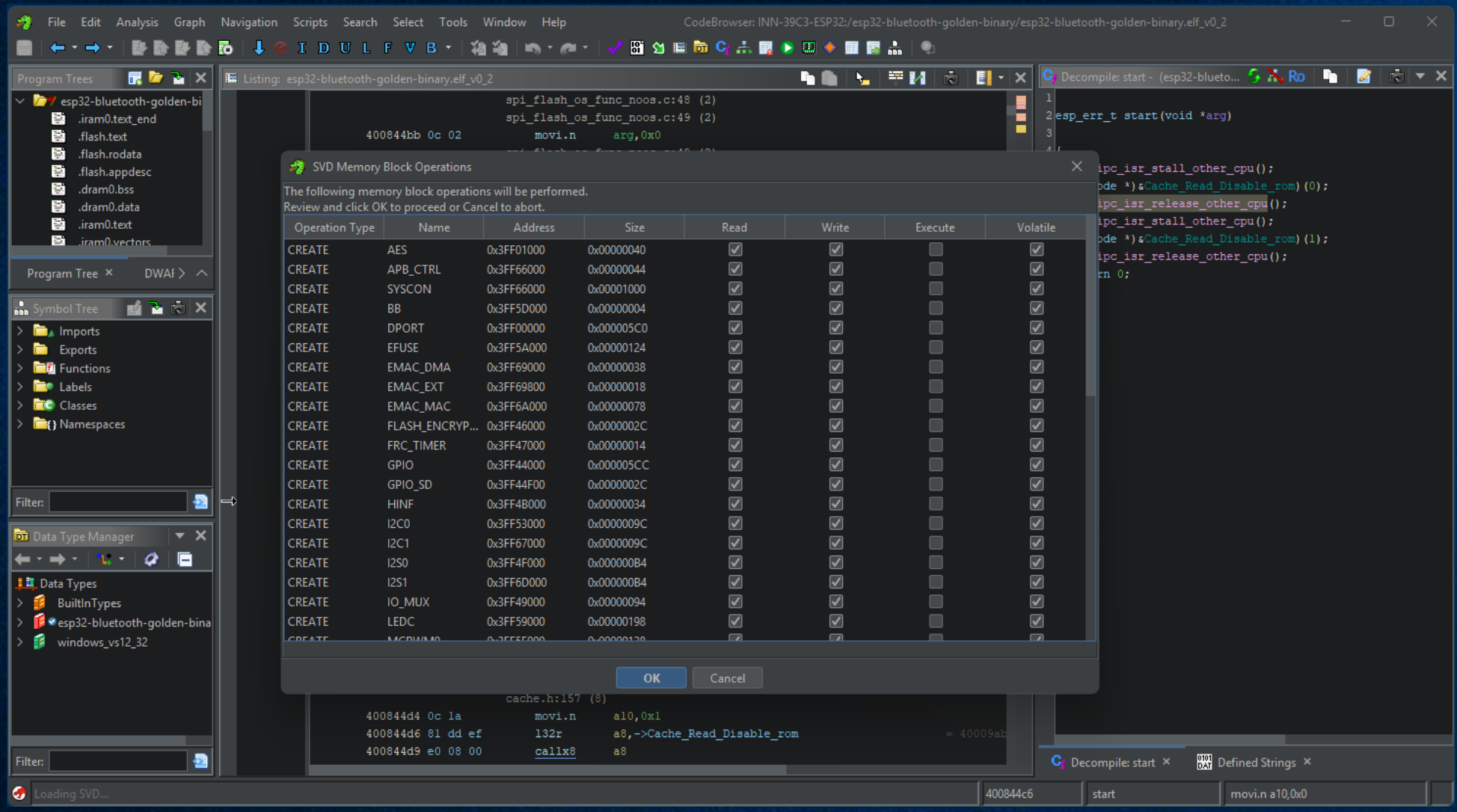
Code Blame Raw

```
1   /*
2   ESP32 ROM address table
3   Generated for ROM with MD5sum:
4   ab8282ae908fe9e7a63fb2a4ac2df013  ../../rom_image/prorom.elf
5   */
6   PROVIDE ( Add2SelfBigHex256 = 0x40015b7c );
7   PROVIDE ( AddBigHex256 = 0x40015b28 );
8   PROVIDE ( AddBigHexModP256 = 0x40015c98 );
9   PROVIDE ( AddP256 = 0x40015c74 );
10  PROVIDE ( AddPdiv2_256 = 0x40015ce0 );
11  PROVIDE ( app_gpio_arg = 0x3ffe003c );
12  PROVIDE ( app_gpio_handler = 0x3ffe0040 );
13  PROVIDE ( BasePoint_x_256 = 0x3ff97488 );
14  PROVIDE ( BasePoint_y_256 = 0x3ff97468 );
```

SVDs

- The “ELF” only contains the memory map of the code/some RAM…
- Code interacts directly with peripheral addresses…
- ESP SVDs (Outdated): https://github.com/espressif/svd
- ESP RS Pacs: https://github.com/esp-rs/esp-pacs
- GhidraSVD: https://github.com/antoniovazquezblanco/GhidraSVD

SVDs

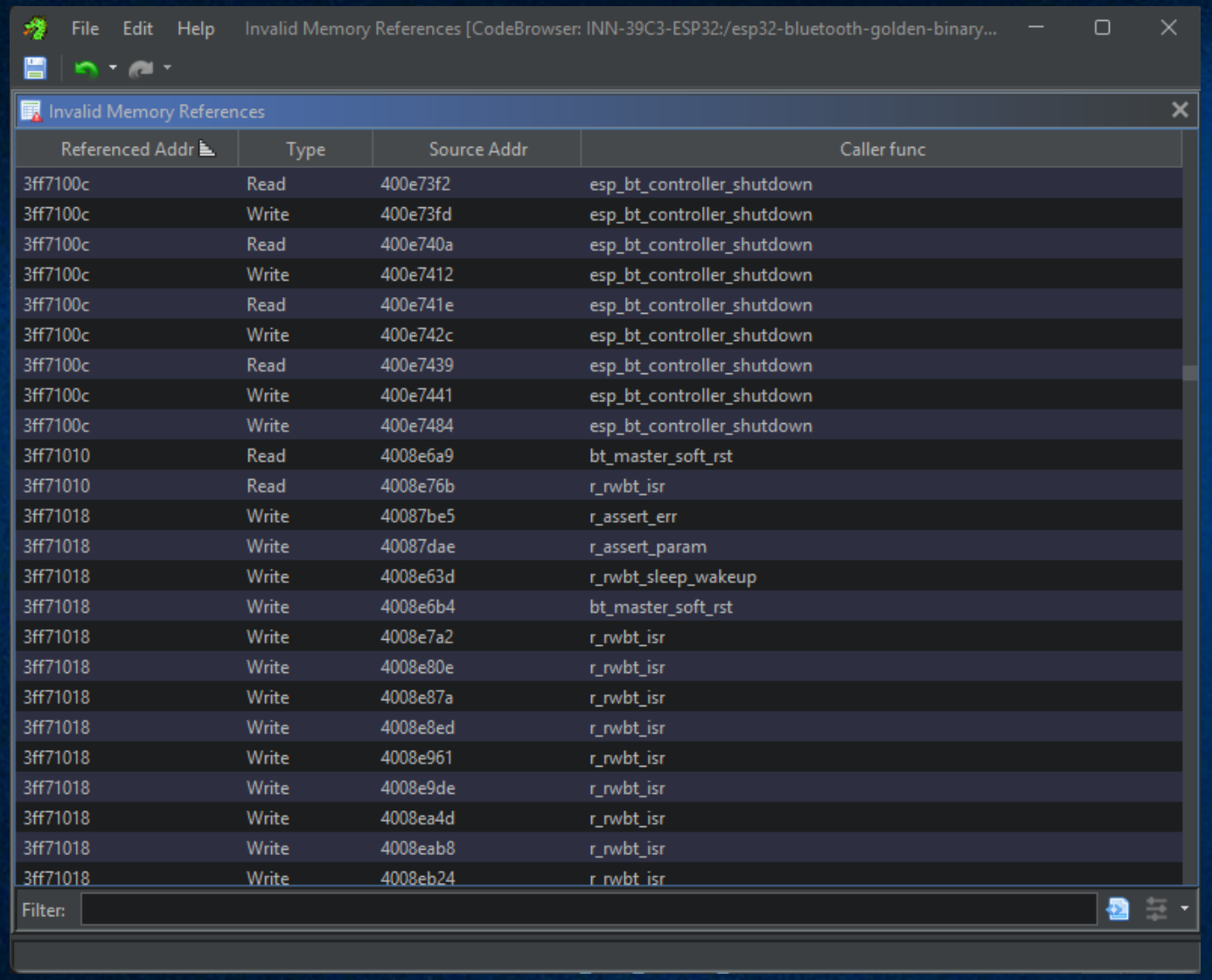

File Edit Analysis Graph Navigation Scripts Search Select Tools Window Help
CodeBrowser: INN-39C3-ESP32:/esp32-bluetooth-golden-binary/esp32-bluetooth-golden-binary.elf_v0_2
Program Trees
esp32-bluetooth-golden-bi
.iram0.text_end
.flash.text
.flash.rodata
.flash.appdesc
.dram0.bss
.dram0.data
.iram0.text
.iram0.vectors
Program Tree
DWAf
Symbol Tree
Imports
Exports
Functions
Labels
Classes
Namespaces
Filter:
Data Type Manager
Data Types
BuiltInTypes
esp32-bluetooth-golden-bina
windows_vs12_32
Filter:
Listing: esp32-bluetooth-golden-binary.elf_v0_2
spi_flash_os_func_noos.c:48 (2)
spi_flash_os_func_noos.c:49 (2)
400844bb 0c 02 movi.n arg,0x0
SVD Memory Block Operations
The following memory block operations will be performed.
Review and click OK to proceed or Cancel to abort.
Operation Type | Name | Address | Size | Read | Write | Execute | Volatile
CREATE AES 0x3FF01000 0x00000040
CREATE APB_CTRL 0x3FF66000 0x00000044
CREATE SYSCON 0x3FF66000 0x00001000
CREATE BB 0x3FF5D000 0x00000004
CREATE DPORT 0x3FF00000 0x000005C0
CREATE EFUSE 0x3FF5A000 0x00000124
CREATE EMAC_DMA 0x3FF69000 0x00000038
CREATE EMAC_EXT 0x3FF69800 0x00000018
CREATE EMAC_MAC 0x3FF6A000 0x00000078
CREATE FLASH_ENCRYP... 0x3FF46000 0x0000002C
CREATE FRC_TIMER 0x3FF47000 0x00000014
CREATE GPIO 0x3FF44000 0x000005CC
CREATE GPIO_SD 0x3FF44F00 0x0000002C
CREATE HINF 0x3FF4B000 0x00000034
CREATE I2C0 0x3FF53000 0x0000009C
CREATE I2C1 0x3FF67000 0x0000009C
CREATE I2S0 0x3FF4F000 0x000000B4
CREATE I2S1 0x3FF6D000 0x000000B4
CREATE IO_MUX 0x3FF49000 0x00000094
CREATE LEDC 0x3FF59000 0x00000198
OK
Cancel
cache.h:157 (8)
400844d4 0c 1a movi.n a10,0x1
400844d6 81 dd ef l32r a8,->Cache_Read_Disable_rom
400844d9 e0 08 00 callx8 a8
Decompile: start - (esp32-blueto...
esp_err_t start(void *arg)
ipc_isr_stall_other_cpu();
ipc_isr_release_other_cpu();
ipc_isr_stall_other_cpu();
ipc_isr_release_other_cpu();
rn 0;
Decompile: start
Defined Strings
Loading SVD...
400844c6
start
movi.n a10,0x0

THE MISSING BITS

- How do we map Bluetooth peripheral related memory?
- How can we know where related code is trying to read and write?
- GhidraInvalidMemoryRefs:

  https://github.com/antoniovazquezblanco/GhidraInvalidMemoryRefs

THE MISSING BITS



Referenced Addr	Type	Source Addr	Caller func
3ff7100c	Read	400e73f2	esp_bt_controller_shutdown
3ff7100c	Write	400e73fd	esp_bt_controller_shutdown
3ff7100c	Read	400e740a	esp_bt_controller_shutdown
3ff7100c	Write	400e7412	esp_bt_controller_shutdown
3ff7100c	Read	400e741e	esp_bt_controller_shutdown
3ff7100c	Write	400e742c	esp_bt_controller_shutdown
3ff7100c	Read	400e7439	esp_bt_controller_shutdown
3ff7100c	Write	400e7441	esp_bt_controller_shutdown
3ff7100c	Write	400e7484	esp_bt_controller_shutdown
3ff71010	Read	4008e6a9	bt_master_soft_rst
3ff71010	Read	4008e76b	r_rwbt_isr
3ff71018	Write	40087be5	r_assert_err
3ff71018	Write	40087dae	r_assert_param
3ff71018	Write	4008e63d	r_rwbt_sleep_wakeup
3ff71018	Write	4008e6b4	bt_master_soft_rst
3ff71018	Write	4008e7a2	r_rwbt_isr
3ff71018	Write	4008e80e	r_rwbt_isr
3ff71018	Write	4008e87a	r_rwbt_isr
3ff71018	Write	4008e8ed	r_rwbt_isr
3ff71018	Write	4008e961	r_rwbt_isr
3ff71018	Write	4008e9de	r_rwbt_isr
3ff71018	Write	4008ea4d	r_rwbt_isr
3ff71018	Write	4008eab8	r_rwbt_isr
3ff71018	Write	4008eb24	r_rwbt_isr

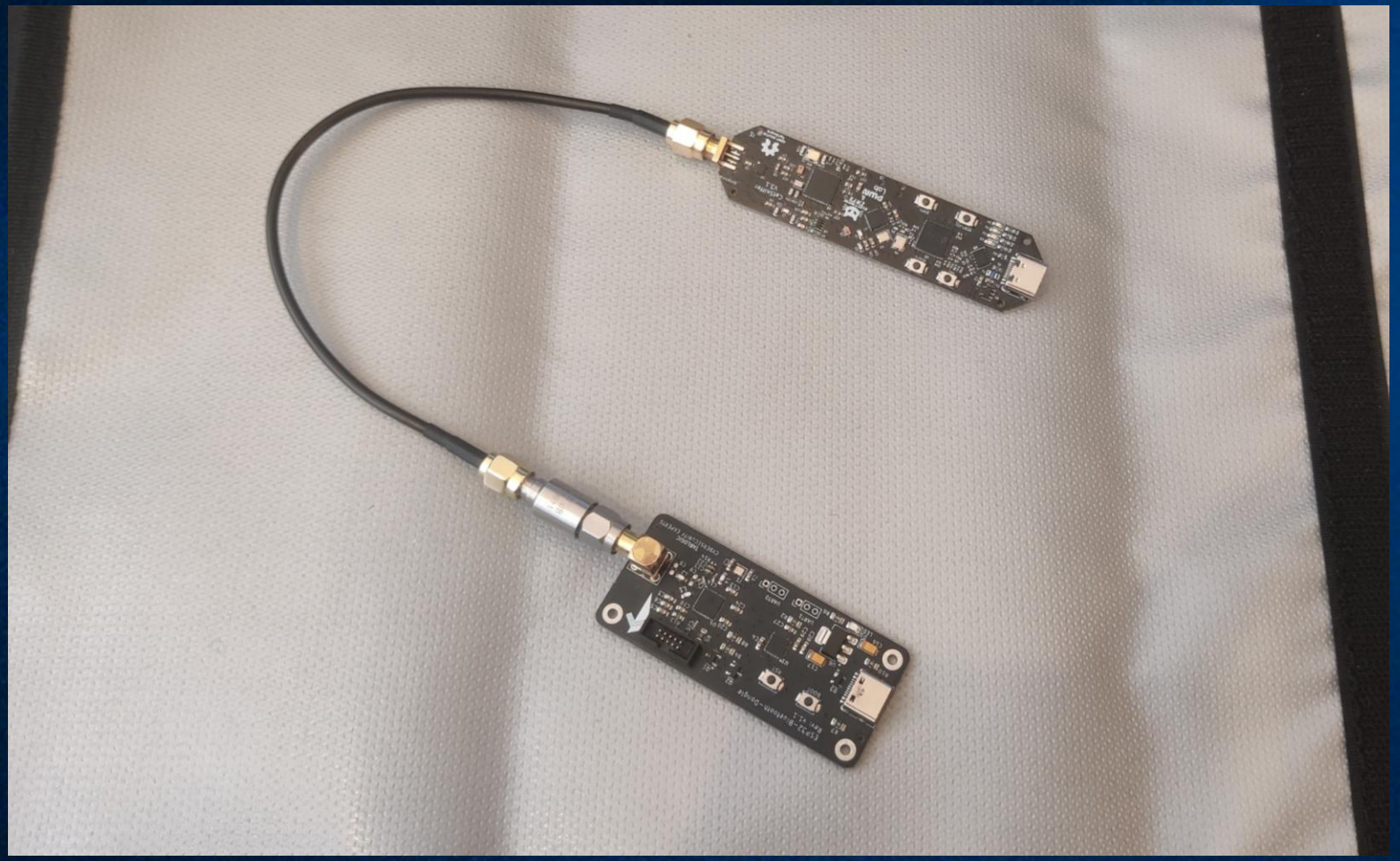
THE PROCESS

- A lot of code reading…
- Context and usage is key (masks, functions…)
- Debug info lists file and line sometimes!
- Asserts leak some register naming! :D

THE SETUP

- When reading is not enough, we need to test!
- Lots of firmware tests to read and write values…
- Designed custom ESP32 boards with JTAG & SMA connectors…
- Additional hardware for specific tests:
  - CatSniffer
  - HackRF

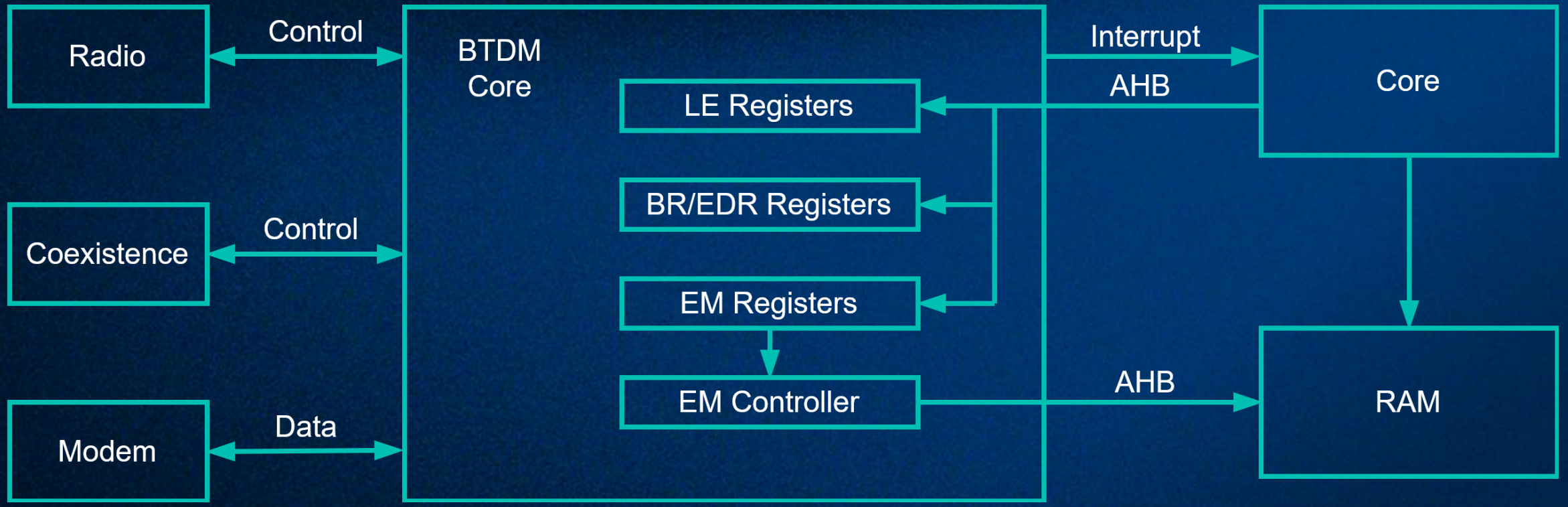
THE SETUP



FINDINGS

- Architecture
- Memory map
- Operation description

ARCHITECTURE



BTDM REGISTERS

- The BTDM block contains registers for BR/EDR, BLE & EM
- Mostly for general peripheral configuration:
  - Enable and disable peripheral, interrupts, error reporting…
  - Options such as encryption, general timing and clocks…
  - Exchange memory configuration…

BTDM REGISTERS

- Documented 40+ registers…
- Documented fields for 20+ registers…
- https://github.com/TarlogicSecurity/esp-pacs
- Hopefully, it will land on https://github.com/esp-rs/esp-pacs

EXCHANGE MEMORY

- Exchange memory contains many linked tables
- Each table handles a part of the Bluetooth information exchange
  - What should a slot do? Scan, Advertise, Connect…
  - Frequency and hop control for that slot…
  - TX and RX buffers…

EXCHANGE MEMORY

Address	Structure
0x3ffb0000	Exchange table
0x3ffb0040	Frequency table
0x3ffb0090	??
0x3ffb0098	BLE encryption
0x3ffb00b8	BLE Control Structures
0x3ffb0480	BLE Whitelist
0x3ffb0510	BLE Resolve Addr List
0x3ffb05ac	BLE TX Descriptors
0x3ffb0934	BLE RX Descriptors
0x3ffb0994	TX Control Buffers
0x3ffb0b82	TX Data Buffers
0x3ffb15aa	RX Buffers
…	BR/EDR Stuff

PREFETCH MECHANISM

- Both user and BTDM Core can modify Exchange Memory…
- How are possible conflicts managed?
- Bluetooth events are very time sensitive…
- How is timing handled?

PREFETCH MECHANISM

- The exchange table has 16 event slots
- Slots are processed synchronized to a 625us clock

PREFETCH MECHANISM

- Before execution of a slot, information is “pre-fetched” from exchange memory…

PREFETCH MECHANISM

- User prepares data in the Exchange Memory
- User sets the event slot as ready to be processed
- BTDM Core processes the slot and notifies the user
- User can read the resulting data…

WHAT NOW?

- What can be done?
- What can’t be done?

WHAT CAN BE DONE

- All standard procedures:
  - Scan
  - Advertise
  - Connect
  - Await connections

WHAT CAN BE DONE

- Testing procedures:
  - RF Test TX
  - RF Test RX

WHAT CAN BE DONE

- But with low level access :D
  - Log low level traffic!
  - Perform per channel scan!
  - Send arbitrary BT RF traffic!
  - Follow & sniff connections!
  - Continuous jamming of a single channel!

WHAT CAN BE DONE

- Modify the existing controller implementation:
  - Port PoCs from other platforms!
  - Low level protocol fuzzing!

WHAT CAN'T BE DONE

- Cannot receive arbitrary frames of unknown "sync word"…
  - No "full monitor mode" :(
  - Partial "sync word" matching is possible :)
- Obtain all Bluetooth packets from this peripheral
  - Maybe reversing some other periph? Modem? RF?

CONCLUSIONS

- Thoughts
- References
- Thanks

FUTURE WORK

- Enough documentation to:
  - Start writing an open-source controller stack!
  - Tinker and implement custom tools & PoCs!
- Maybe reversing remaining peripherals?
  - Modem? RF PHY?

REFERENCES

- GhidraLinkerScript:

  https://github.com/antoniovazquezblanco/GhidraLinkerScript
- GhidraSVD:

  https://github.com/antoniovazquezblanco/GhidraSVD
- GhidraInvalidMemoryRefs:

  https://github.com/antoniovazquezblanco/GhidraInvalidMemoryRefs

REFERENCES

- SVDs are now public, a PR will soon follow:

  https://github.com/TarlogicSecurity/esp-pacs

- Documentation is available:

  https://www.tarlogic.com/blog/

  https://github.com/TarlogicSecurity/talks

  https://github.com/TarlogicSecurity/ESP32-Bluetooth-Reversing

THANKS

- To 

for financing the research
- To Isaac Lleida for his help while reversing!
- To 

for the open SDK & license choices
- To @Frostie314159 and @Jasper Devreker for ESP32 Open WiFi
- To all of you for listening!