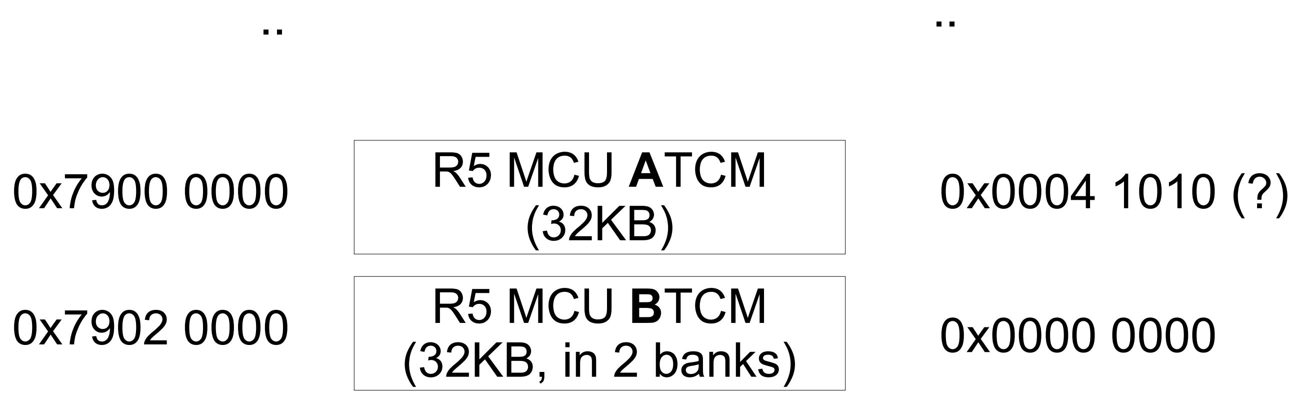
Transferring data between R5 <==> Linux

Topics

1) How we share data between Linux and the R5

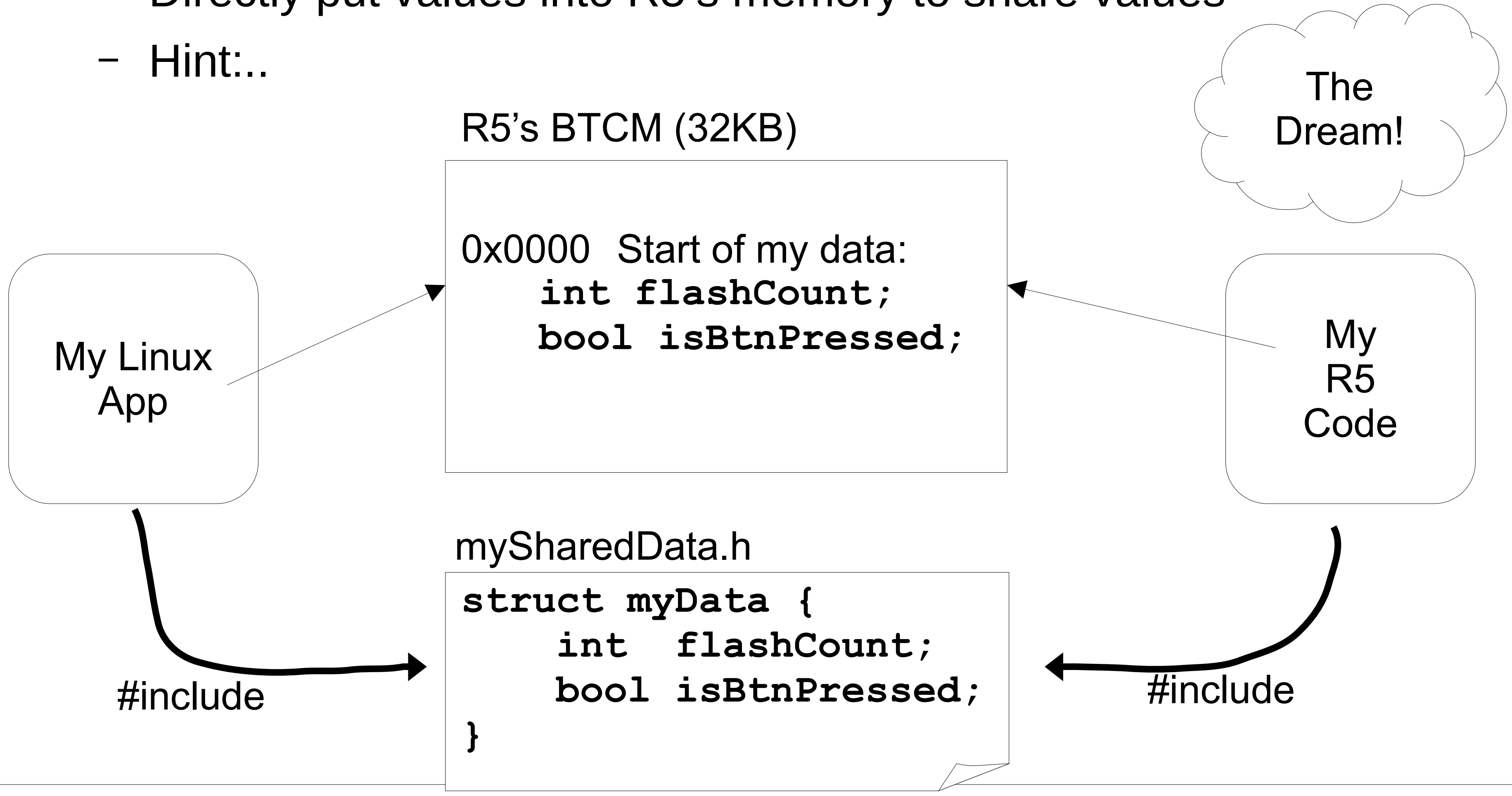
Memory sharing



- To use BTCM, Linux global address 0x79020000
  - Must be mapped into your app’s memory space with mmap()

Memory Use

- Shared Memory Idea
  - Directly put values into R5’s memory to share values
  - Hint:..



mySharedData.h

```
struct myData {
    int  flashCount;
    bool isBtnPressed;
}
```

Sample Program - Shared Struct

- Shared .h file
  - Create one .h file which defines
    ..
    between R5 & Linux
  - Each program #include this same file

```
typedef struct {
  bool isLedOn;
  bool isButtonPressed;
} sharedMemStruct_t;
```

sharedDataStruct.h

Scratch this idea...
It does not work on the BYAI
at the moment!

Reality

- The R5 halts when accessing a struct pointer.
  - Trying to do myStruct->count = 0;
    haults the processor.
  - But, using a separate pointer works:
    int *ptr = &myStruct->count;
    *ptr = 0;
- Why?
  - No clue.
- Solution?
  - Raw memory access, or array access.

Demo

- See sharedMem example
  - R5 code built with r5_mcu_build.sh
  - Linux code built with make
- Load R5 code with load_r5_mcu.sh

- What could we do to improve the code from raw memory pointers?
  - Array?
  - Enum?

Packing Structs

Data Types

- C data types can be of different sizes
  - C spec simply mentions their relative size
  - R5 and Linux use:

1 byte:	char
2 bytes:	short
4 bytes:	int, long, float
8 bytes:	long long, double

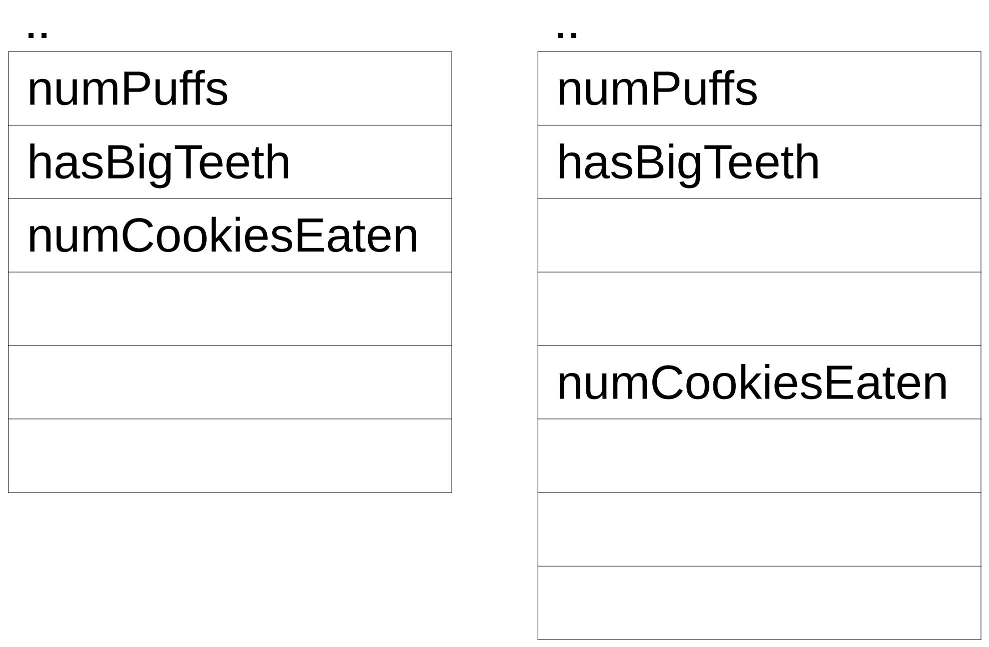
- ..
  - Gives integer data types based on #bits
  - Useful for..
    uint8_t, uint16_t, uint32_t, uint64_t
    int8_t, int16_t, int32_t, int64_t

Structs

- Structs store different types of data in one allocated unit of memory

```
struct bigBadWolfData_t {
   char numPuffs;
   bool hasBigTeeth;
   int numCookiesEaten;
};
```

- How does this layout in memory?



2 Processors

- Incorrect alignment gives a bus error
- Word align int/uint32_t
- Double world align doubles, long long, uint64_t

Padding Structs

```
struct bigBadWolfData_t {
   char numPuffs;
   bool hasBigTeeth;
   char _pad1, _pad2;
   int numCookiesEaten;
};
```

Padded
numPuffs
hasBigTeeth
_pad1
_pad2
numCookiesEaten

Padding bytes

- Add extra bytes to struct..

  char/bool: byte aligned

  int/uint32_t: word aligned

  double/uint64_t: dword aligned

- Once padded correctly, struct is identical as both packed and unpacked processors
  - Incorrect padding means values written to a field by one processor not seen correctly by other.

Troubleshooting

- Hard to debug the R5 because ..
  - Write very little code at a time, then test it.
  - Flash the LED for some visual status
- Common Issues
  - Permission denied on /dev/mem:
    run with sudo
  - Input/output not working:
    check you have run GPIO code on Linux first
  - Data exchange problems:
    R5 halts on struct access; use array.
  - Changes to code not running:
    add compile-time error to check if correct code is compiling

Summary

- R5 Memory
  - 32KB in ATCM and BTCM banks
  - ~~Can use a struct to define which values are in shared memory~~
    - NOPE! Use raw memory / array
- Linux <==> R5 Memory
  - Linux app calls mmap() to request access to R5 memory
- Alignment / Packing
  - pad structs to line up data