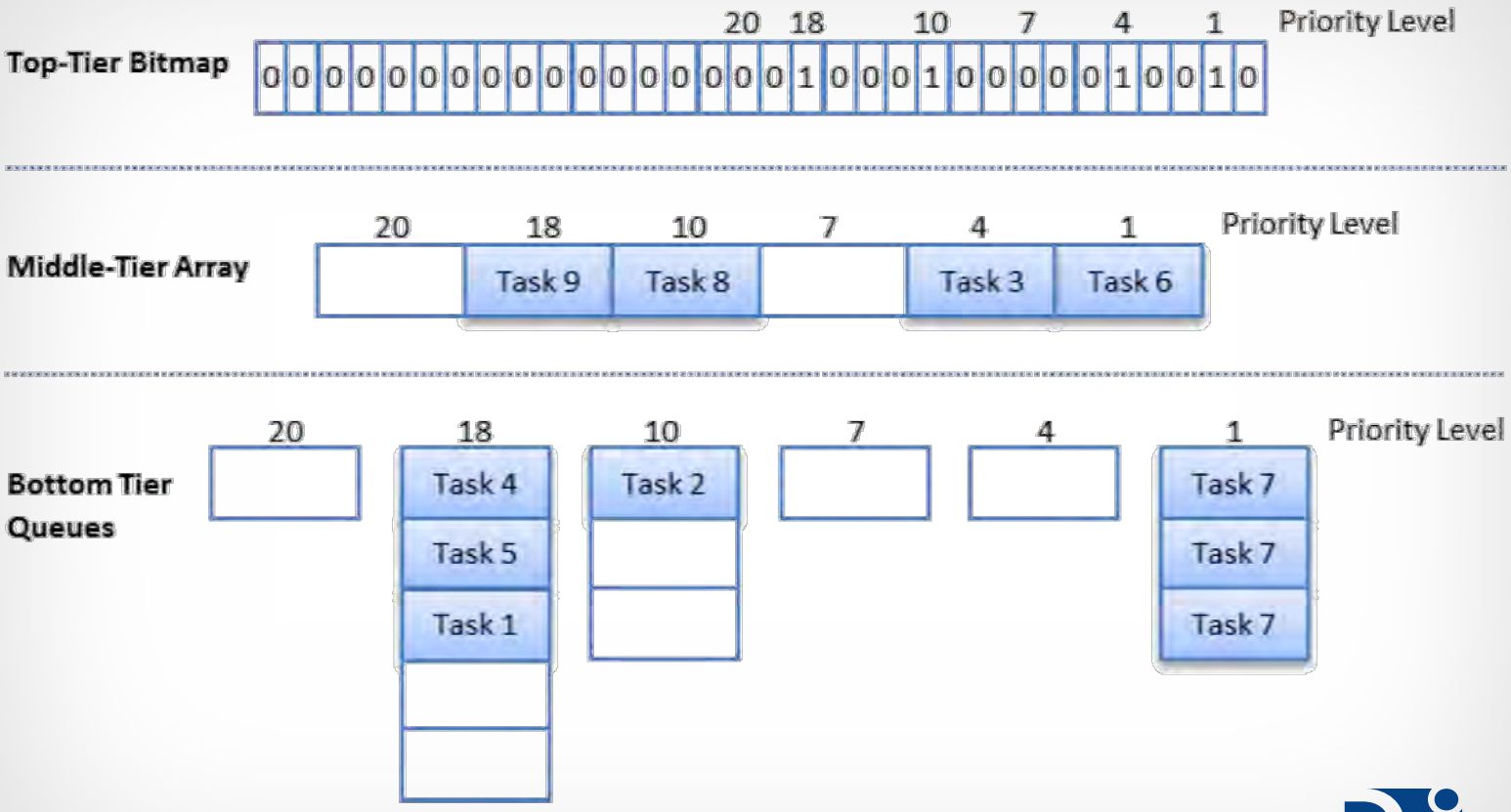
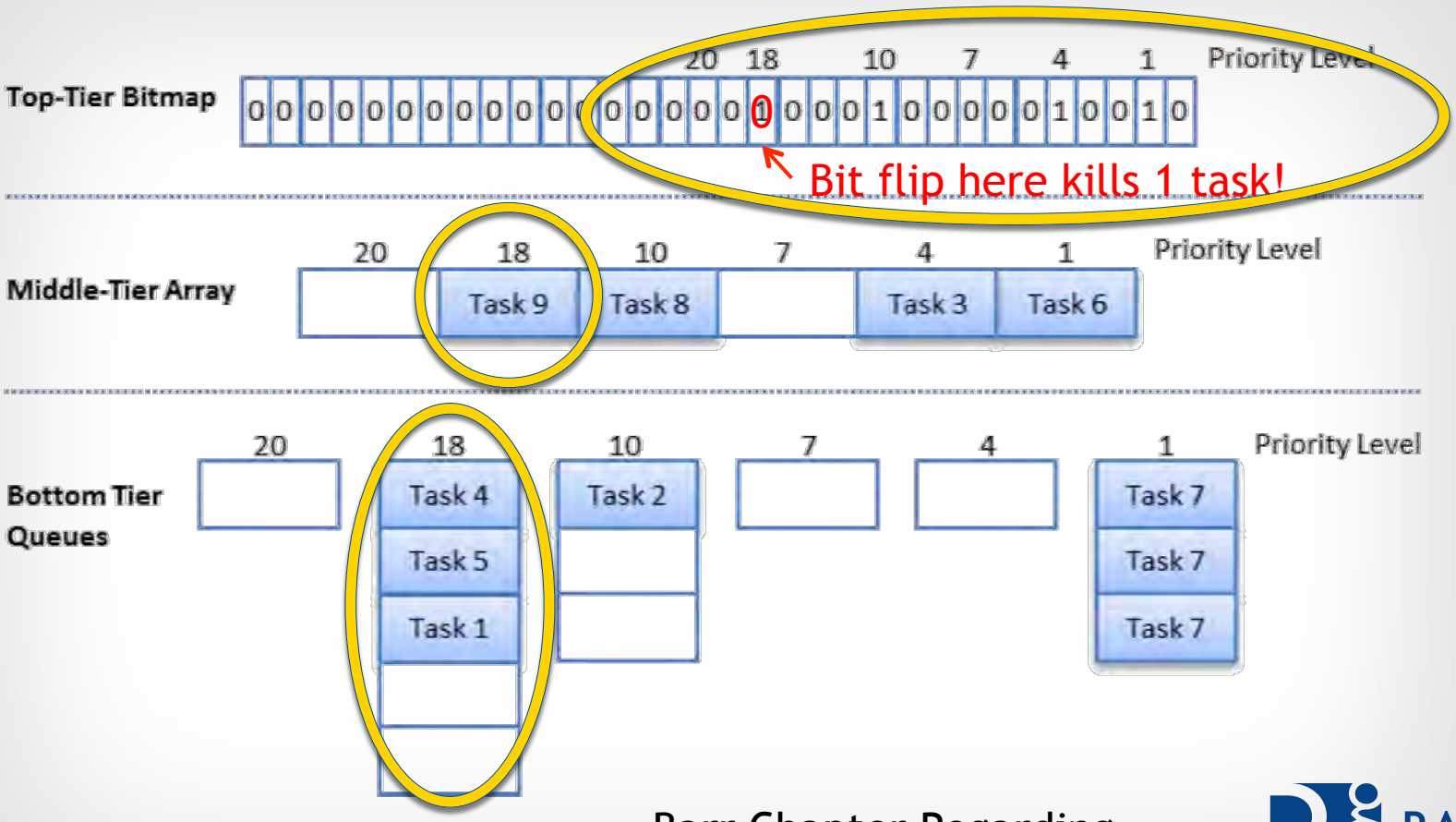
OSEK'S CRITICAL DATA STRUCTURES



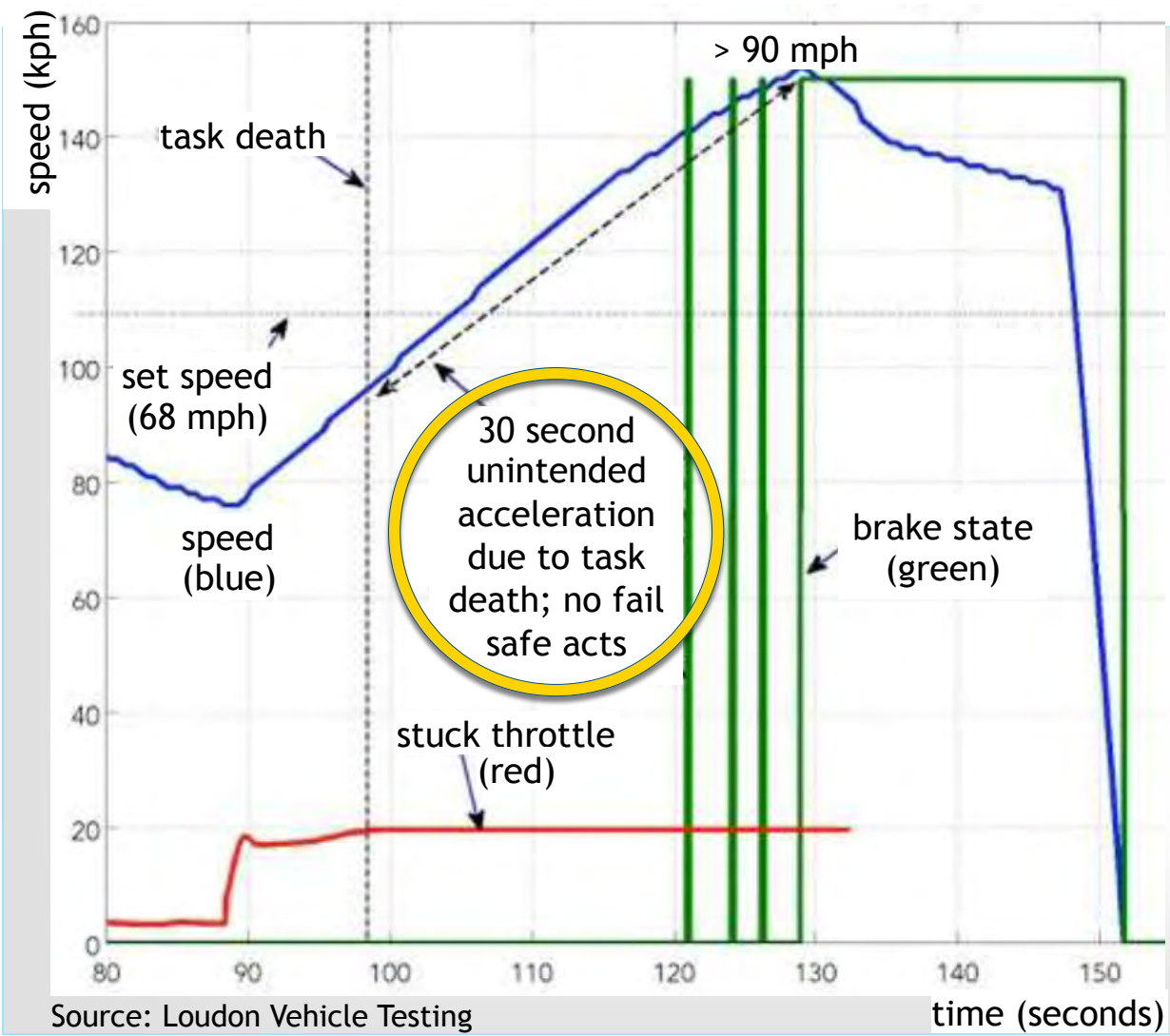
Barr Chapter Regarding
Toyota's Operating Systems

MEMORY CORRUPTION AND TASK DEATH



Barr Chapter Regarding
Toyota's Operating Systems

EXAMPLE OF UNINTENDED ACCELERATION



Source: Loudon Vehicle Testing

- Representative of task death in real-world
- Dead task also monitors accelerator pedal, so loss of throttle control
  - ✓ Confirmed in tests
- When this task's death begins with brake press (any amount), driver must fully remove foot from brake to end UA
  - ✓ Confirmed in tests

BARR group

SOFTWARE CAUSES OF MEMORY CORRUPTION

Type of Software Defect	Causes Memory Corruption?	Defect in 2005 Camry L4?
Buffer Overflow	Yes	Yes
Invalid Pointer Dereference/Arithmetic	Yes	Yes
Race Condition (a.k.a., “Task Interference”)	Yes	Yes
Nested Scheduler Unlock	Yes	Yes
Unsafe Casting	Yes	Yes
Stack Overflow	Yes	Yes

Barr Chapter Regarding Toyota’s Software Bugs

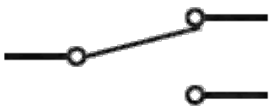
SPAGHETTI CODE DEFINED

space *n.* A logic 0 on an RS-232 link. Any voltage between +3 and +25 V. *See also* mark.

spaghetti code *n.* Incomprehensible source code, typically including apparently meaningless jumps or `gotos` or a high degree of unnecessary coupling between modules.

spawn *v.* To create a new thread of execution.

SPDT (as letters) *abbr.* A type of switch that has one actuator (pole) that connects to one of two contacts. Short for Single Pole, Double Throw. Used to select one of two conditions. *Compare to* SPST.



The schematic symbol for an SPDT switch makes its design and purpose clear.

spec (speck) *abbr. See* specification.

- Difficult to follow data/control paths
- Bugs likely to appear when modified
- Unnecessarily complex

Ganssle&Barr, Embedded Systems Dictionary, 2003

BARR group

TOYOTA'S SPAGHETTI CODE

3. Software assembly for power train ECU TOY-MDL04983210

After the 4th Steering Committee, rebuilding of engine control and actions for software assembly were started.

(1) Achievements

① Identification of current issues with software assembly ..... Ongoing

- There are C sources for which there is no specification document. (e.g., communication related)
- Specification document and C source do not correspond one-to-one. (e.g., cruise, communication related)

② Activities to improve the spaghetti-like status of engine control application were started.

(Control structure reform has already started in Engine Div. In coordination with this, software structure reform will be carried out. As a first step, it has been decided to transfer two employees from Engine Div. and carry out trial with purge control.)

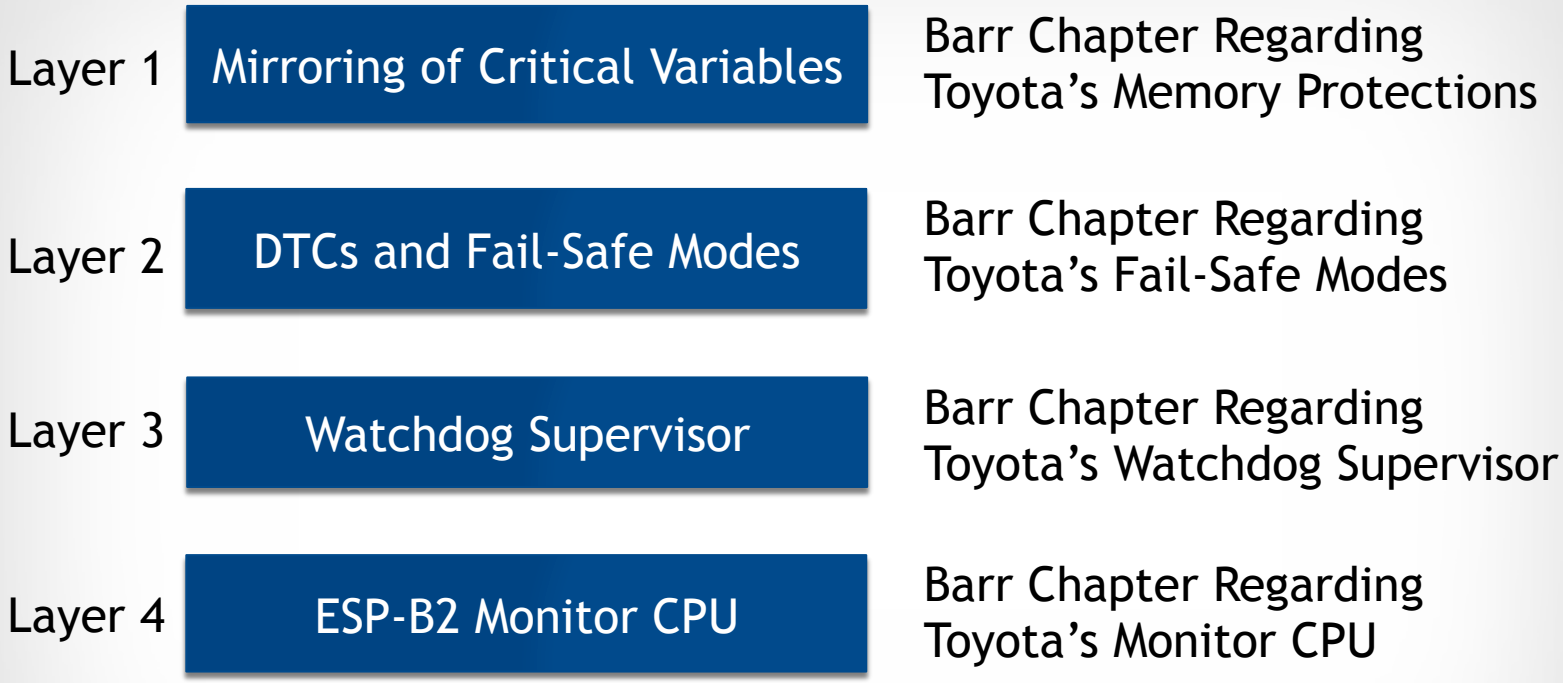
Because structure design is not being implement, a "spaghetti" state arises, both TMC and suppliers struggle to confirm overall situation

TOY-MDL04983219

Without care, systems can quickly get too big and complex, and like dinosaurs, will eventually go extinct.

TOY-MDL04983253
TOY-MDL04983252P-0002

TOYOTA'S DEFECTIVE "SAFETY LAYERS"



BARR group

LAYER 1: MIRRORING OF CRITICAL VARIABLES

Toyota's engineers sought to protect numerous variables against software- and hardware-caused corruptions

- e.g., by "mirroring" their contents in a 2nd location

But FAILED TO MIRROR several key critical variables

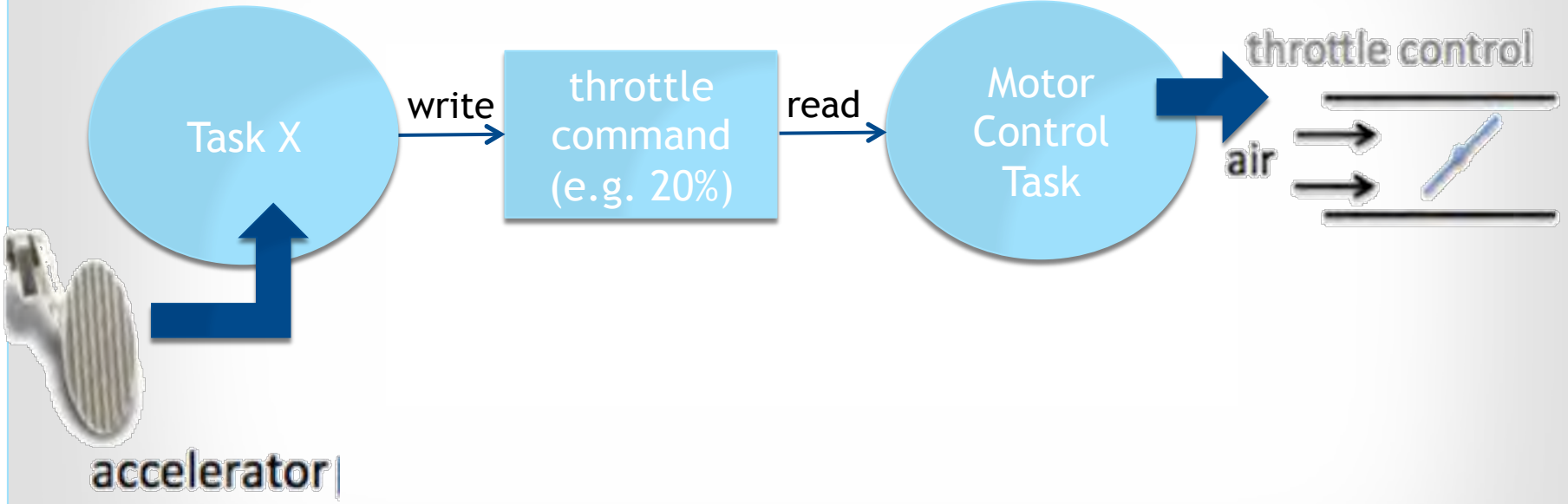
- OSEK's critical internal data structures
- THE *target throttle angle* global variable!
  - Commands a part of the software to open the throttle
    - Recalculated every 8 ms (when the tasks are all alive)

Corruption is indistinguishable from a driver gas pedal press!

Barr Chapter Regarding
Toyota's Memory Protections

BARR group

THROTTLE COMMAND DESIGN



Barr Chapter Regarding
Toyota's Software Architecture

BARR group

UA VIA MEMORY CORRUPTION

Task X death causes loss of throttle control by driver

- Changes at the accelerator pedal have no effect on throttle angle
- Cruise control switches have no effect

Motor Control Task continues to drive throttle motor; engine powered

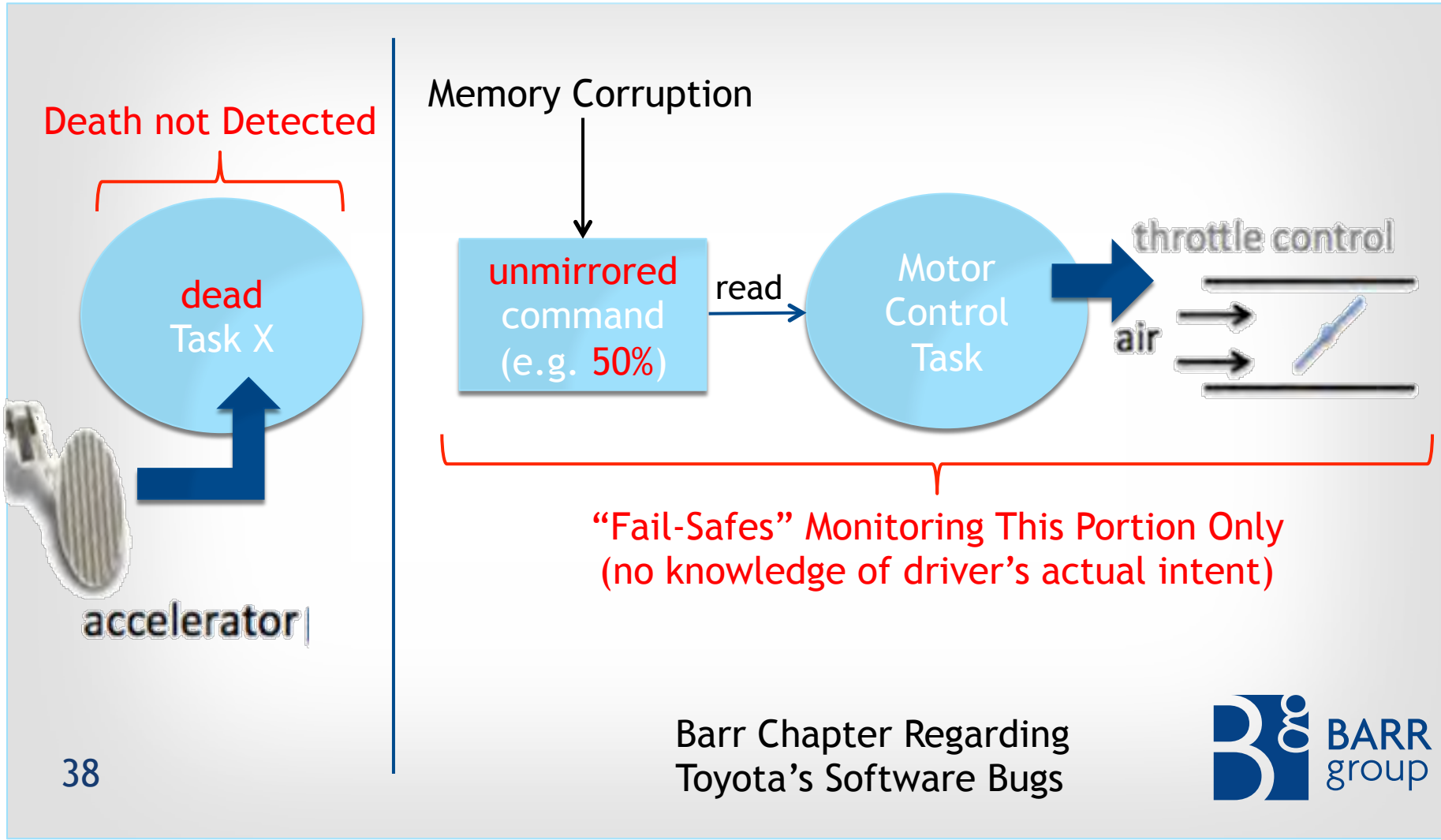
- Throttle could stick at last computed *throttle command*, or
- Change angle via corruption of *throttle command* global variable

One corruption event can cause task death and open throttle

- Memory corruptions are like ricocheting bullets

Barr Chapter Regarding
Task Death and UA

TOYOTA'S DEFECTIVE THROTTLE CONTROL



Barr Chapter Regarding
Toyota's Software Bugs

BARR group

LAYER 2: DTCs AND FAIL-SAFE MODES

NASA talks about 5 fail-safe modes (pp. 79-83)

- Limp home modes 1-3 (*degrees of gas pedal sensor mistrust*)
- Idle mode fuel cut (*2,500 rpm limit at idle*)
- Engine off (*via several different "class 2" failures*)

However, all 5 fail-safes are in same Task X

- Throttle control and fail-safes in same fault containment region

  Unreasonable design; alternative structures well-known

Most diagnostic trouble codes need Task X too!

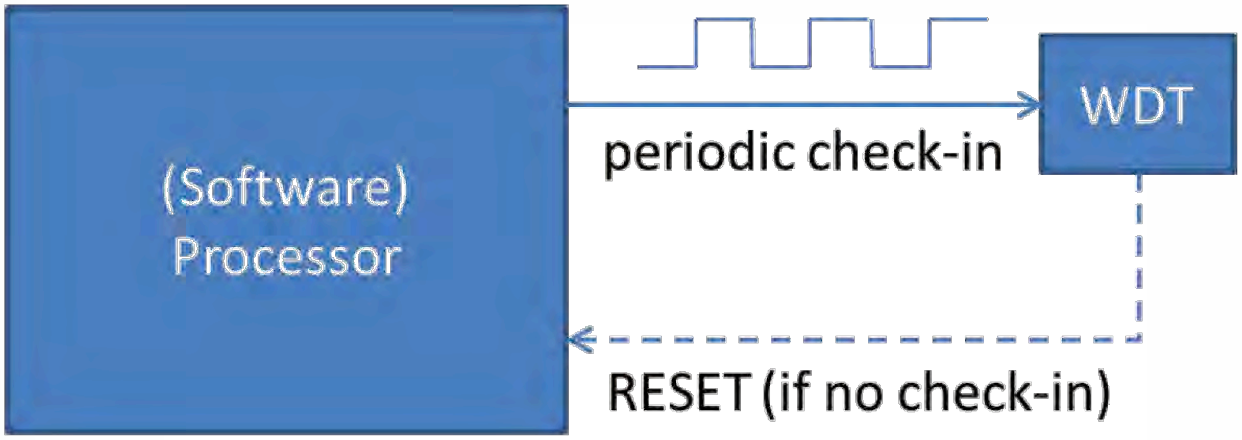
Barr Chapter Regarding
Toyota's Fail Safe Modes

BARR group

LAYER 3: WATCHDOG SUPERVISOR

A "watchdog timer" is hardware to auto-reset software

- Healthy software should periodically "check-in" to prevent reset



With multiple tasks, health of all tasks must be checked

Barr Chapter Regarding
Toyota's Watchdog Supervisor

BARR group

TOYOTA'S DEFECTIVE WATCHDOG DESIGN

Toyota's watchdog supervisor design is unreasonable

- Incapable, ever, of detecting death of majority of tasks
- Incapable of properly and reliably detecting *CPU overload*
- Allows vehicle misbehavior due to overloads lasting up to 1.5s
- Resets the watchdog timer hardware in a timer tick ISR
- Explicitly ignores and discards most operating system error codes

Ignoring error codes violates a MISRA-C rule (1998: #86; 2004: #16.10)

Reasonable design alternatives were well known

- Indeed the primary purpose should've been to detect task death
- 2005 Prius (HV-ECU) watchdog is better

Barr Chapter Regarding
Toyota's Watchdog Supervisor

BARR group

LAYER 4: ESP-B2 MONITOR CPU

"System Guards"

- All (3) useless after Task X death (don't know driver intent)

"Brake Echo Check"

- Depends on the driver to take action—after UA has already begun!

  Sometimes a counter-intuitive/dangerous action

  ➢ Clearly this is not a "designed" fail-safe for UA or task death
- Takes the wrong actions (*should've reset ECM not stalled car*)
- Not 100% reliable

Does not detect all main CPU malfunctions

Barr Chapter Regarding
Toyota's Monitor CPU

TOYOTA FAILED TO REVIEW MONITOR CPU

A: ***With respect to [the monitor CPU], the development process is completely different. When it comes to the source code that would be embedded in [the monitor CPUs] we, Toyota, don't receive them. … there would not be a design review done on the software.***

Q: *Now, the monitoring software for the electronic throttle control system is in the [] ESP-B2 chip; correct?*

A: ***Yes.***

- Ishii 5/24/12 Deposition, pp. 36-37

Barr Chapter Regarding
Toyota's Monitor CPU

BARR group

AGAIN: FAILED TO REVIEW MONITOR CPU!

The critical "monitor CPU" that checks the main CPU has never been independently reviewed

- Toyota doesn't even have a copy of the source code
- NASA didn't review that critical system component either

ESP-B2 source code was not provided to NASA

Barr Group has reviewed Denso's ESP-B2 source code

- Monitor CPU for 2005-2009 Camry L4 (and some other models)

Barr Chapter Regarding
Toyota's Monitor CPU

BARR group

MONITOR CPU IS LAST LINE OF UA DEFENSE

But ESP-B2 monitor CPU could have included a proper UA defense:

- **IF (driver is braking & throttle is not closing) THEN reset ECM**
  Something is not right with the main CPU when that happens!
  Resets of main CPU barely noticeable at speed (brief rpm drop)
- CRITICAL to ending UA in vehicles with potential vacuum loss

Per car cost to add this safety feature is $0.00 (it's just bits)

- There was enough memory and CPU bandwidth for these instructions
- All of the required electrical inputs and outputs were already present
- In line with E-Gas Level 3 recommendations

Barr Chapter Regarding
Toyota's Monitor CPU

BARR group

TOYOTA'S DEFECTIVE SOFTWARE PROCESS

FMEA was incomplete; single points of failure are present

- Because: Toyota didn't adopt a formal safety process

Peer reviews not done on OS code and ESP-B2 code

- Because: Toyota didn't perform code reviews; used non-standard OSEK

Toyota's own "power train" coding standard not enforced

- Because: Toyota didn't follow through with software suppliers

Watchdog supervisor doesn't detect most task's deaths

- Generally costs less to push the limits than upgrade to faster CPU

No EDAC protection against hardware bit flips

- Generally costs less to make memory chips without EDAC

  If confident, why let NASA believe there was EDAC?