



Sawblade Ventures, LLC

Austin, Texas

Accessing Sawblade DAFCA*TM from Software

2021

* Design Automation for Flexible Chip Architecture

Foreward

This presentation was originally prepared by David Whelihan, PhD, Director of Cybersecurity Research and Development for DAFCA interests.

The paper has been revised by Keith Guidry, Chief Technology Officer for Sawblade Ventures, LLC.

The material is intended as a means of educating those unfamiliar with DAFCA's extensible and interoperative theory, practice and potential.

The principles displayed demonstrate the essence of granular security and defense, a concept not available in modular security or trust methods commonly used in semiconductor operations. Of particular interest is DAFCA's ability to extend internal circuit operations to external software systems for data, information, command and control.

This unique ability to offer reprogrammable, reconfigurable and rerouteable circuit structure is a hallmark of the patented methods underpinning the DAFCA facilities and remains a powerful tool for problems and threats that loom over the horizon yet are near us today since 2010.



Objective and Scope

- Inform software engineers how DAFCA enables offensive and defensive security functions in your system
- Detail the software API, which provides access to the deep, system-wide information provided by the DAFCA distributed security sub-system



What is DAFCA?

- A hardware-based monitoring and control fabric
 - Inserted directly into your system
 - Programmable via a kernel-level software module
 - Capable of operating in two modes
 - Autonomous - invisible to the running system
 - Interactive - driven by system-level software
 - Distributed with small hardware devices located at each *system interface* of interest



What is a System Interface?

- A communication path between physical components in your system
 - Access profiles (e.g. rules) for communication and secure resources can be formulated
 - Specific system resources or components can be isolated and closely monitored



What Can DAFCA Do?

- Monitor access patterns on a system interface
 - Programmable “on-the-fly” via software to look for different patterns
- React to patterns with countermeasures independent from the main processes
 - Block resource accesses
 - Inject resource accesses



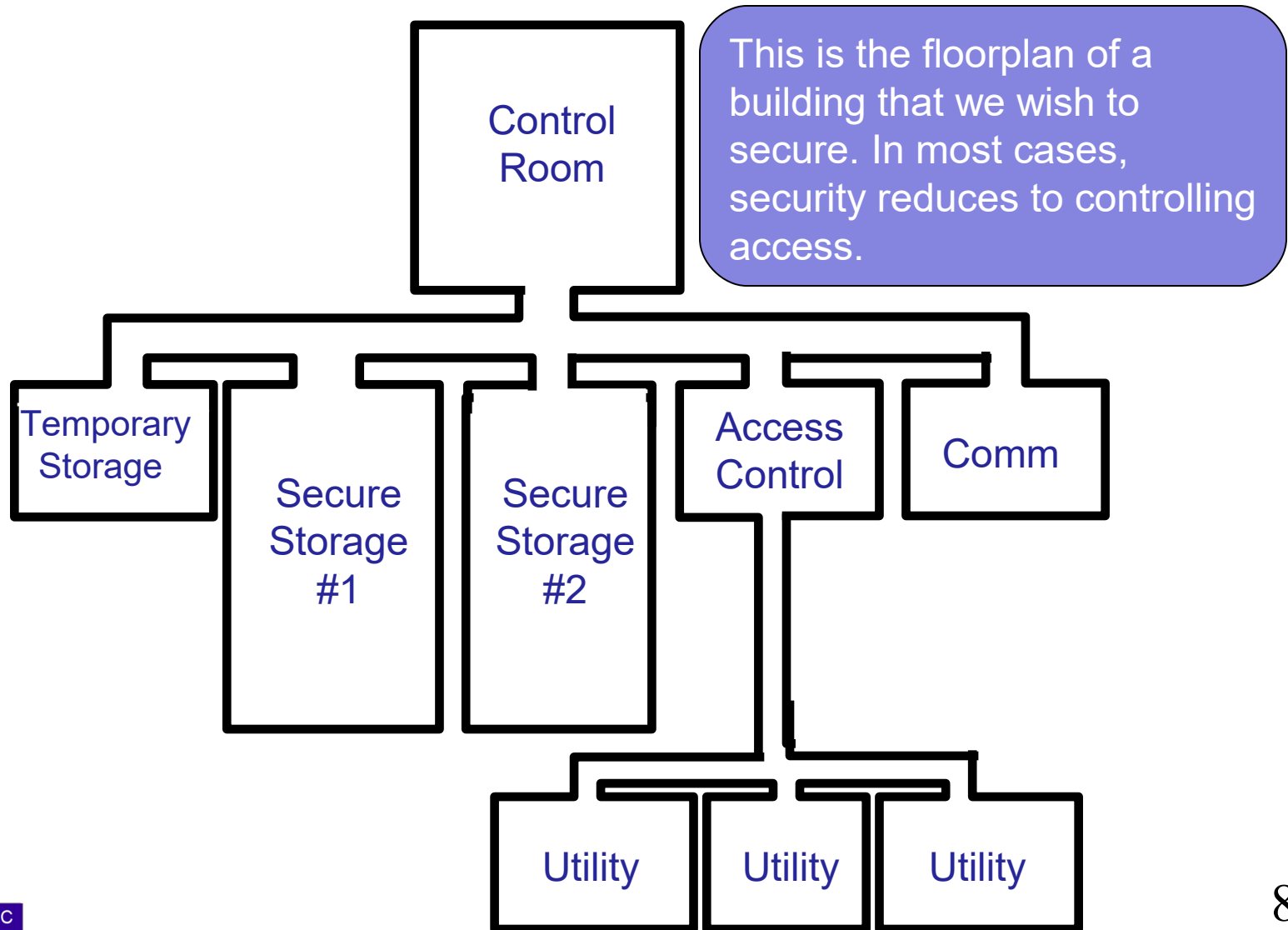
Securing a Building

The following analogy illustrates where DAFCA is in your system:

A Chip is Like a Building

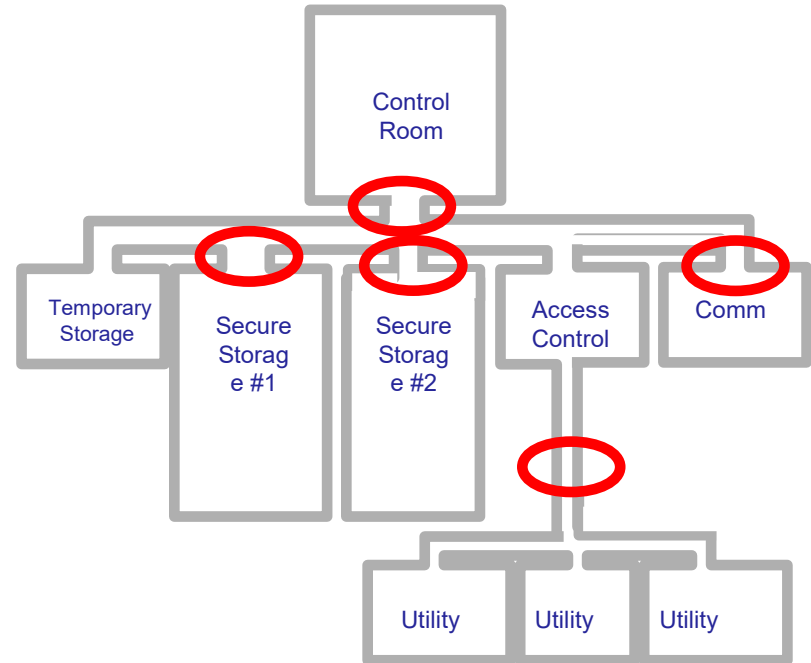


Securing a Building



Securing a Building

To control access:
Identify key access points

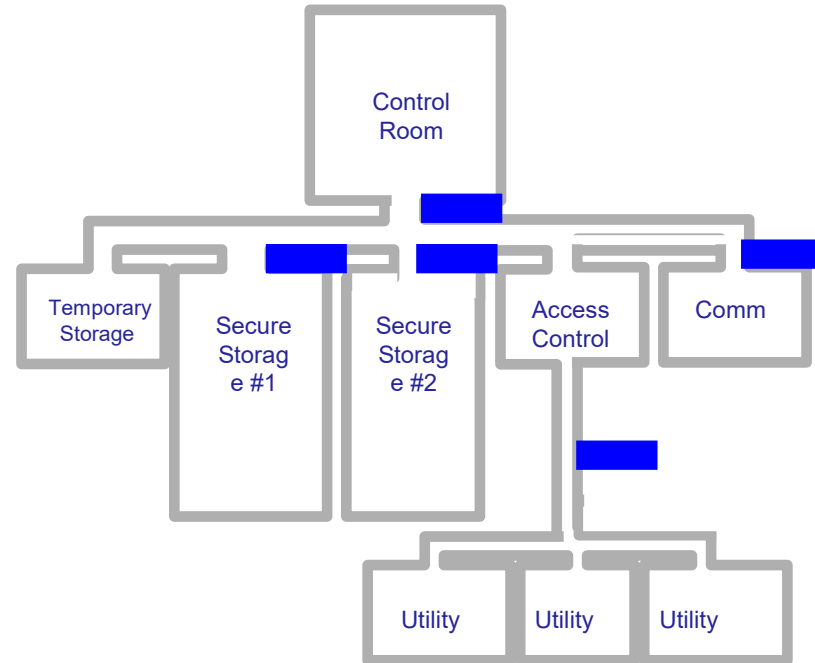


Securing a Building

To control access:

Identify key access points

Place security monitoring stations (one-way mirrors) at the critical points



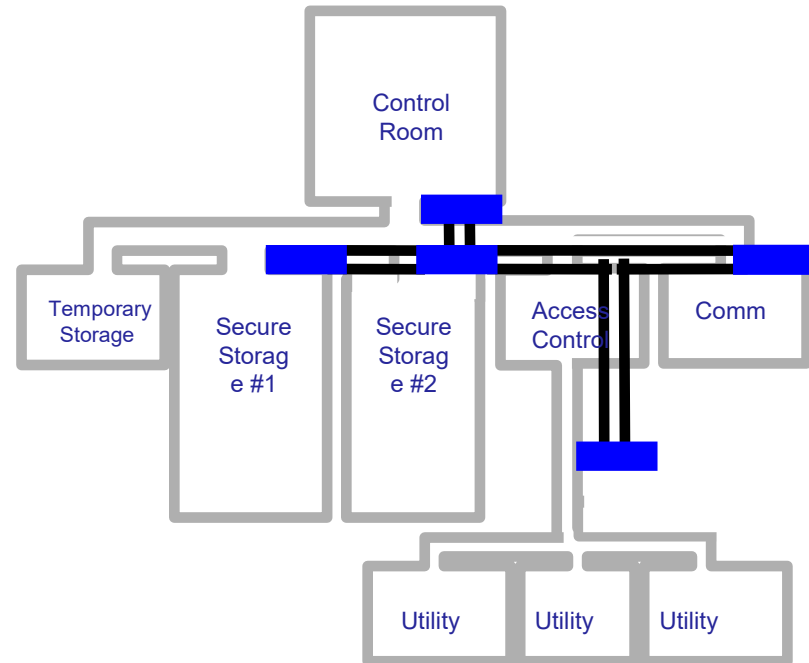
Securing a Building

To control access:

Identify key access points

Place security monitoring stations (one-way mirrors) at the critical points

Interconnect the stations: This access structure is completely separate!



Securing a Building

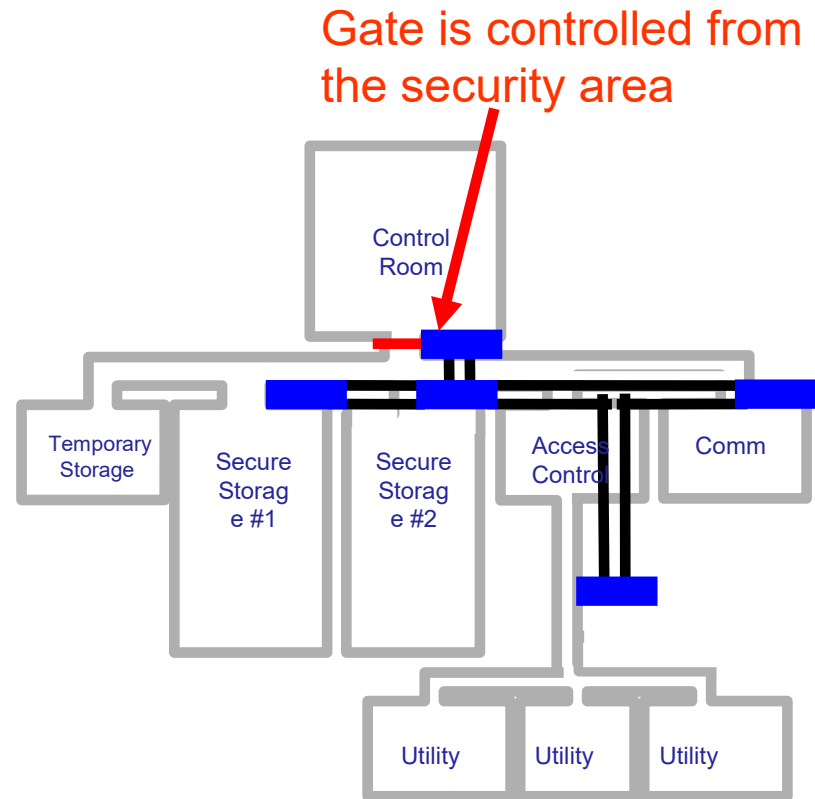
To control access:

Identify key access points

Place security monitoring stations (one-way mirrors) at the critical points

Interconnect the stations: This access structure is completely separate!

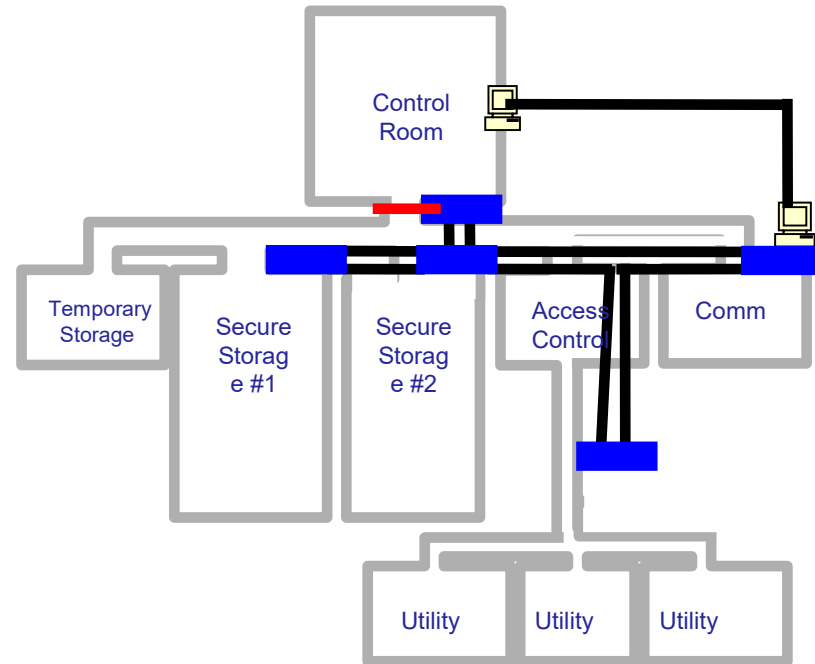
One can only get to the control room if there is someone watching!



Securing a Building

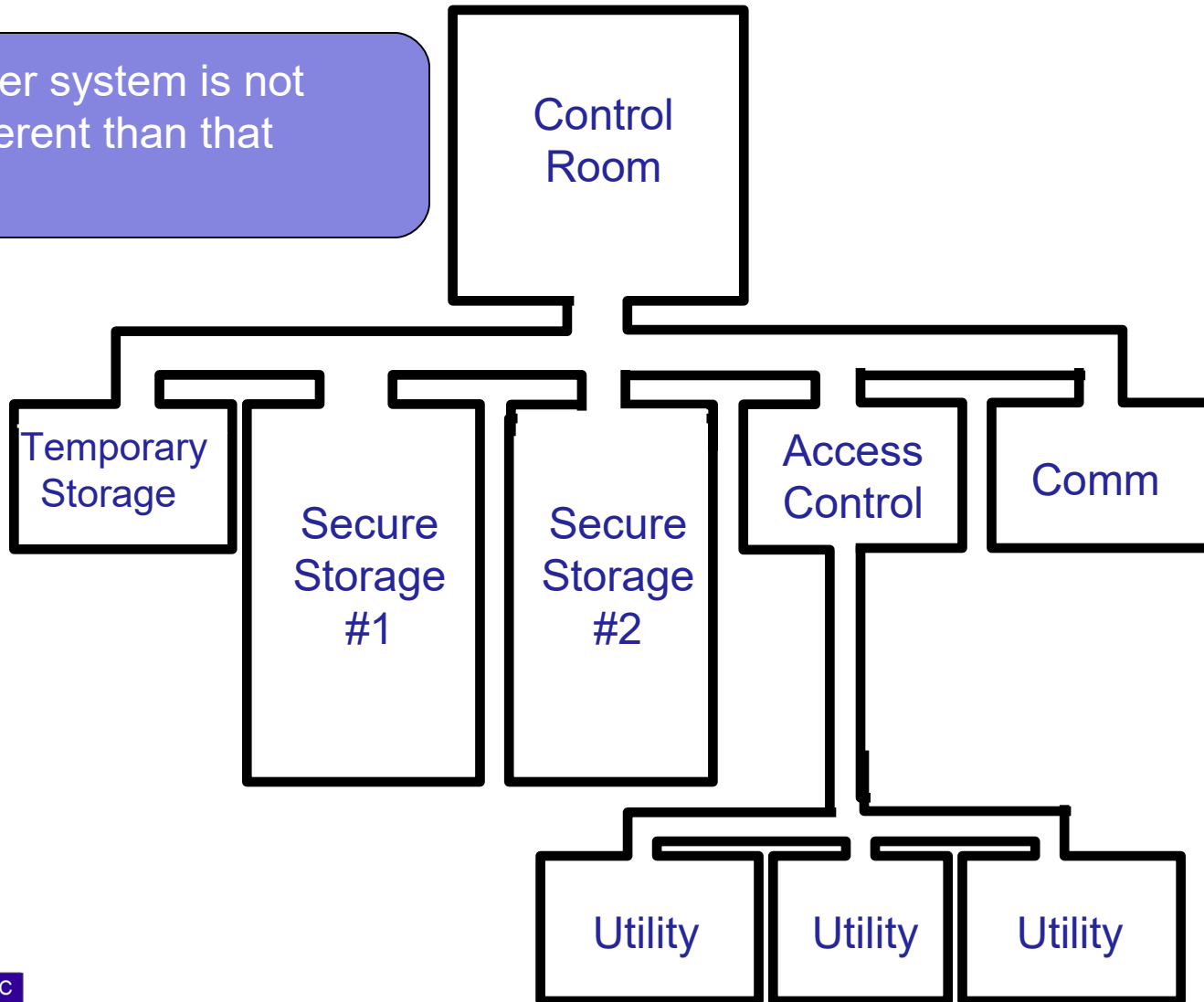
Communication:

Add a secure channel to present security status information to the control room



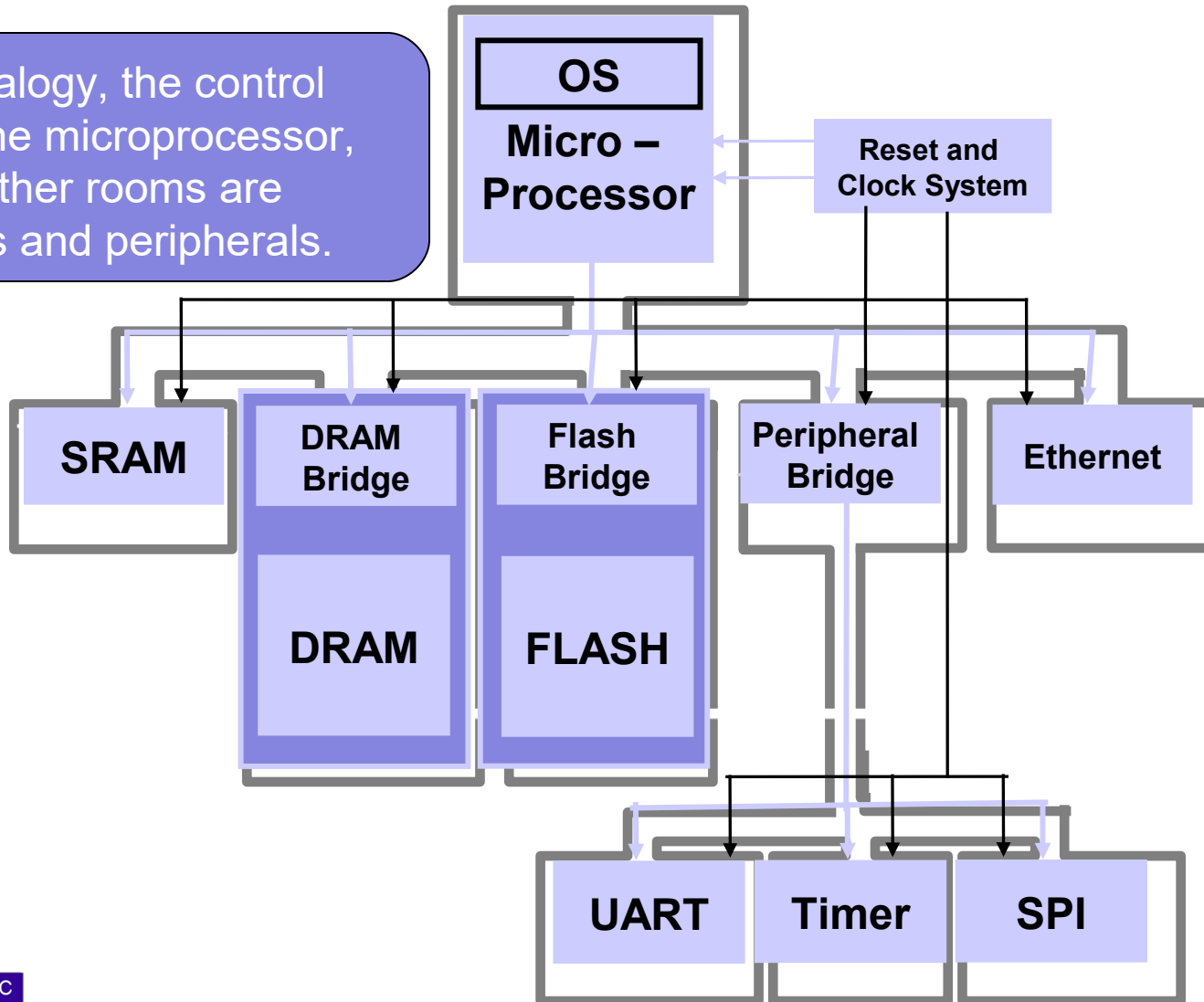
The Micro-world vs The Macro-world

A computer system is not much different than that building.



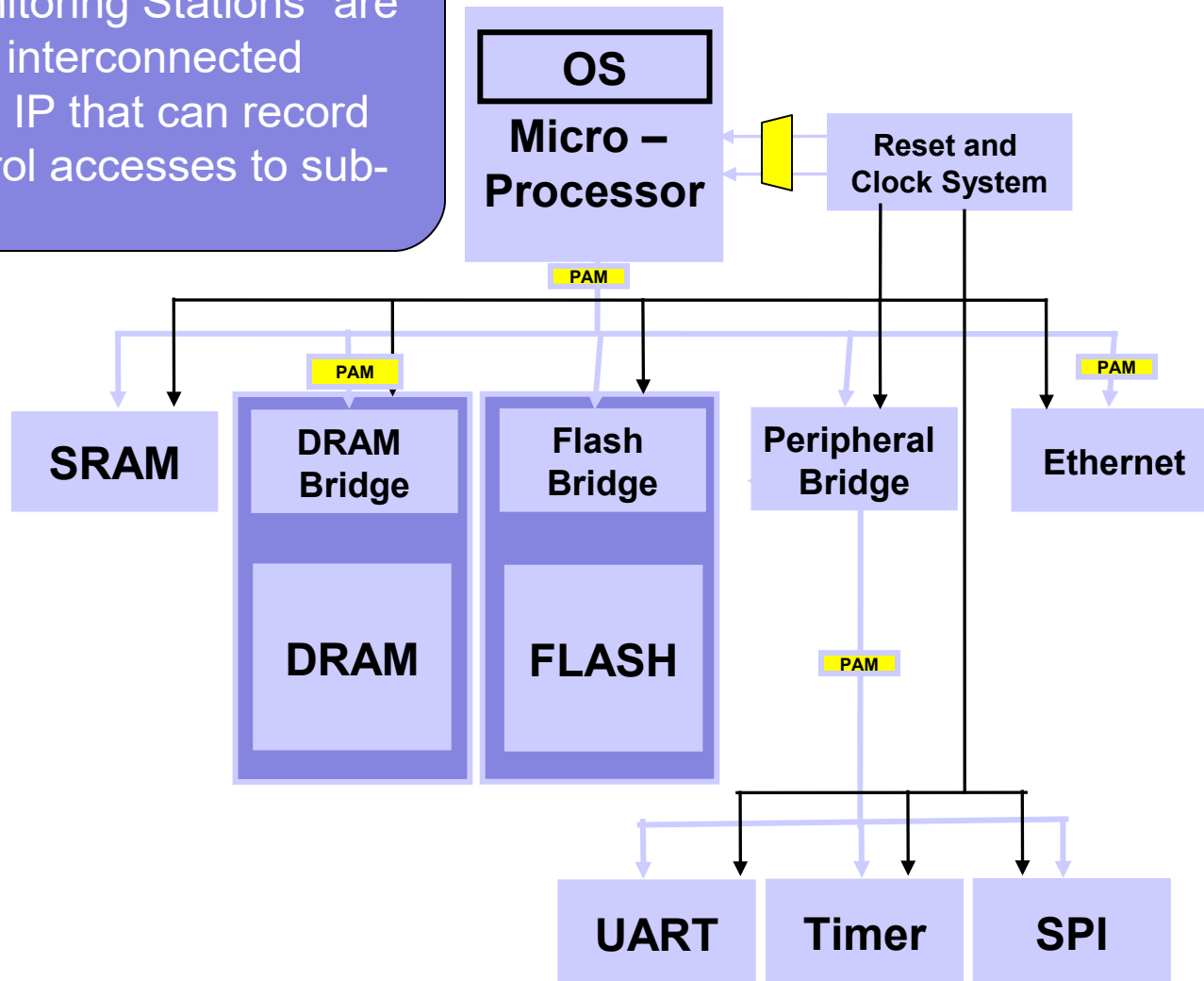
The Micro-world vs The Macro-world

In this analogy, the control room is the microprocessor, and the other rooms are memories and peripherals.



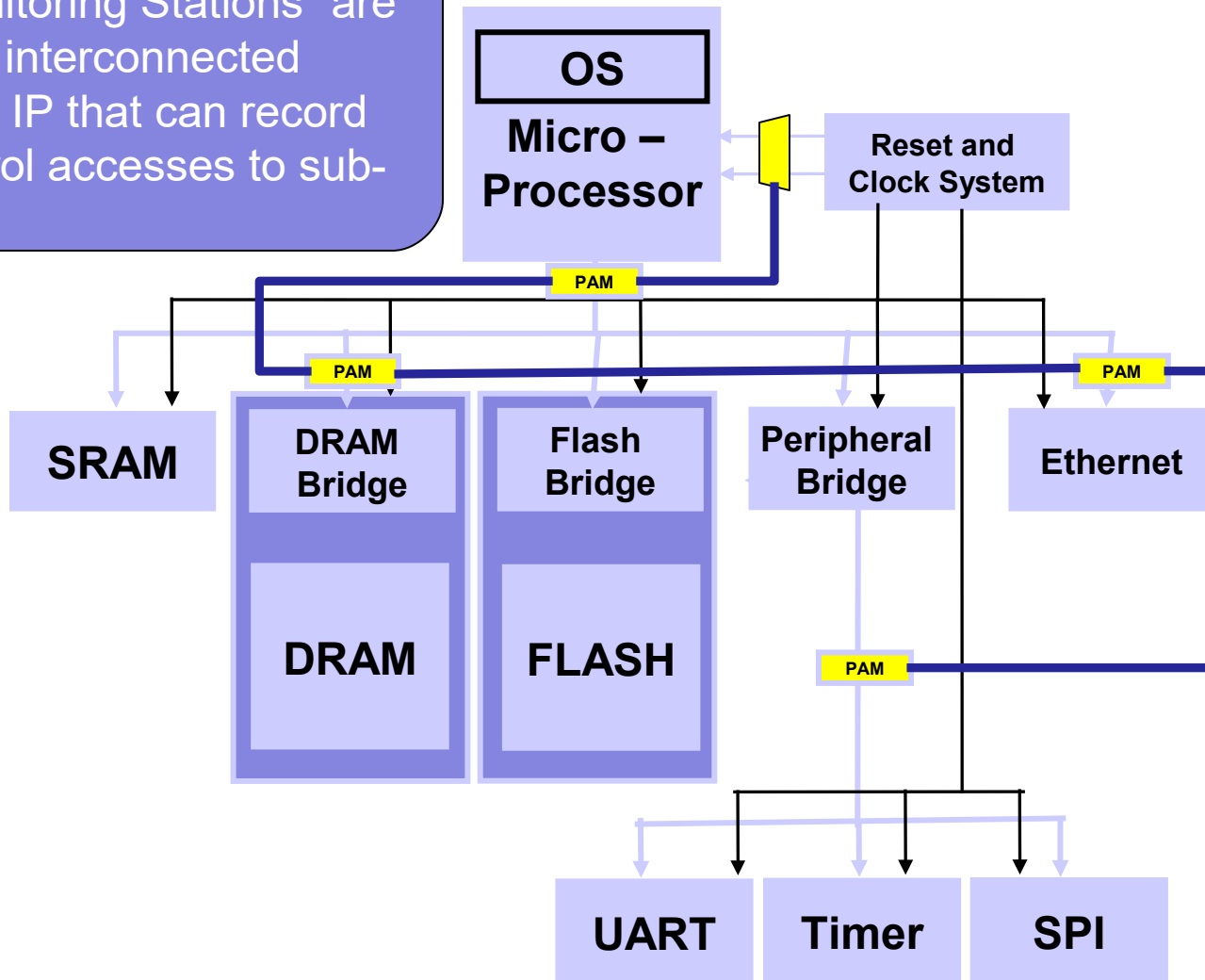
A DAFCA-Enabled System

The “Monitoring Stations” are pieces of interconnected hardware IP that can record and control accesses to sub-systems.



A DAFCA-Enabled System

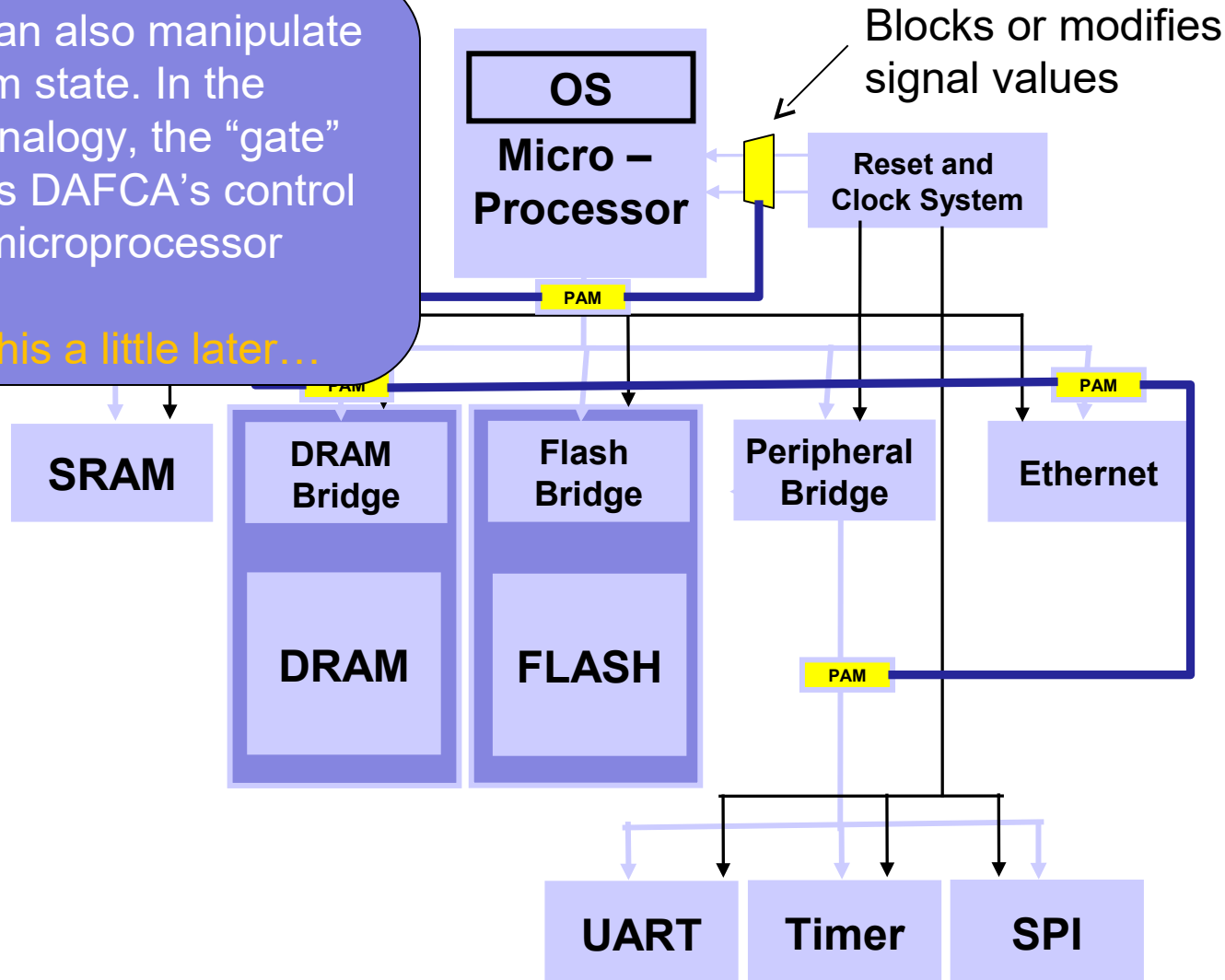
The “Monitoring Stations” are pieces of interconnected hardware IP that can record and control accesses to sub-systems.



A DAFCA-Enabled System

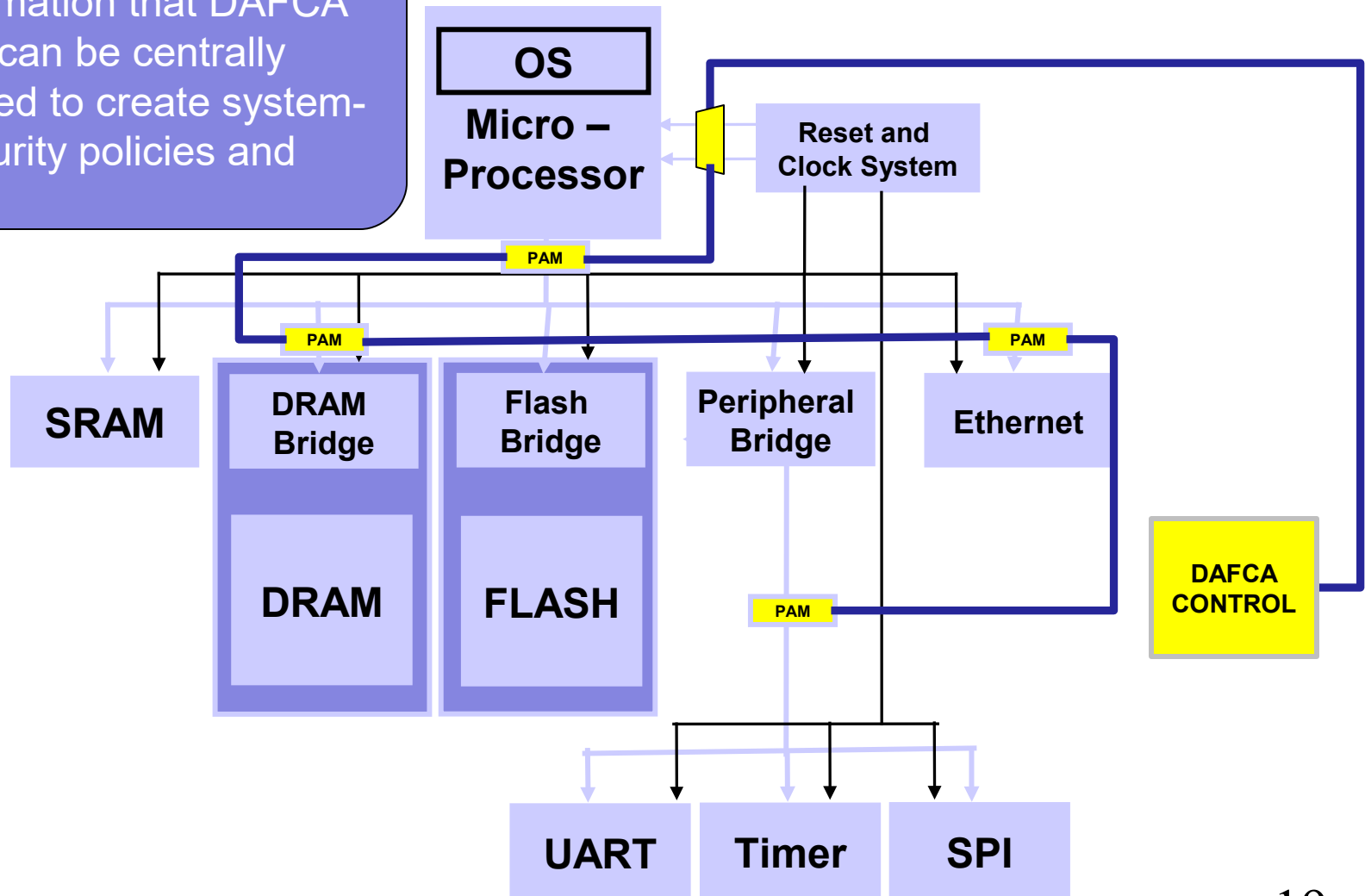
DAFCA can also manipulate the system state. In the building analogy, the “gate” represents DAFCA’s control over the microprocessor reset.

More on this a little later...



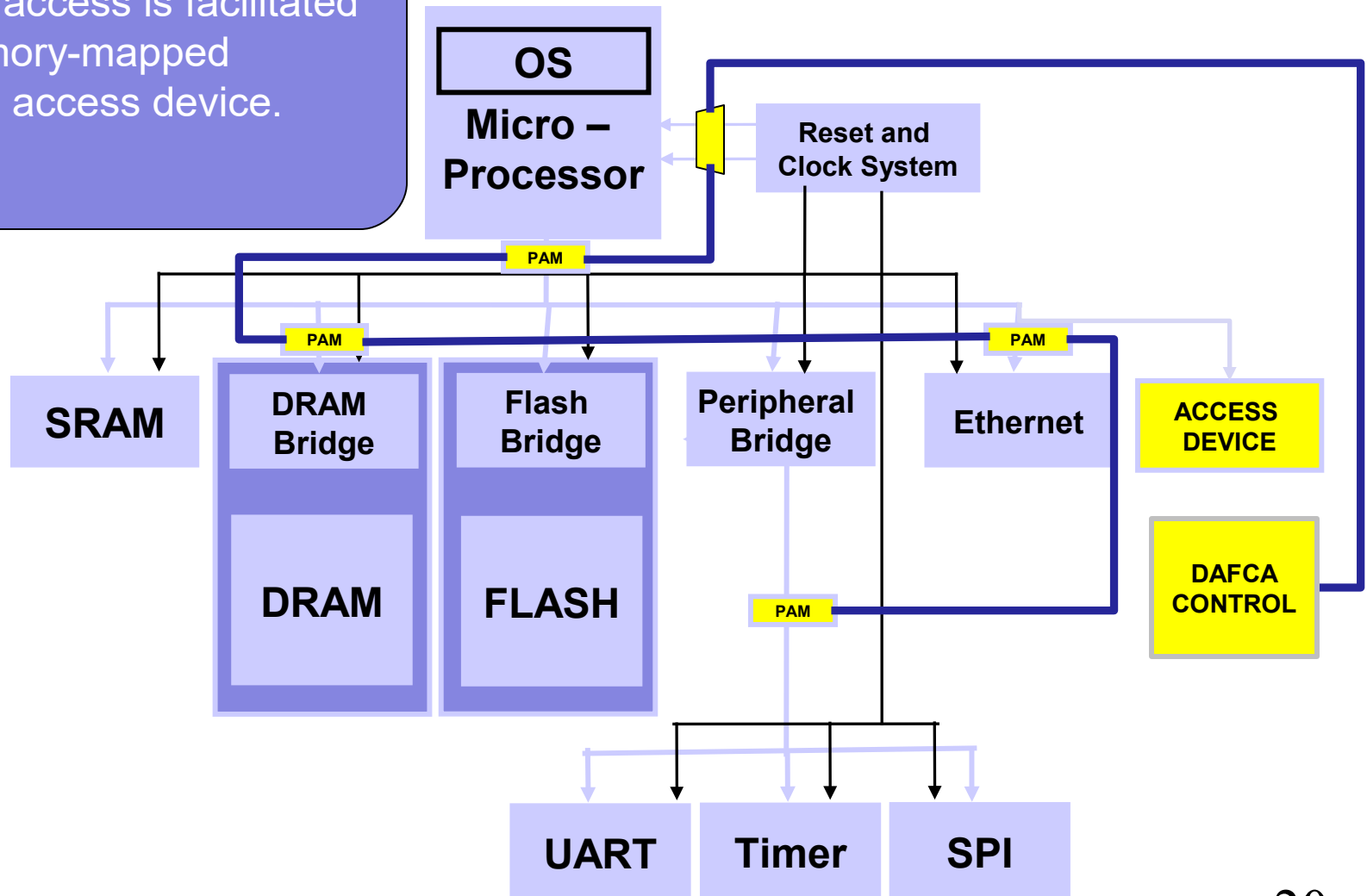
A DAFCA-Enabled System

The information that DAFCA provides can be centrally aggregated to create system-wide security policies and profiles.



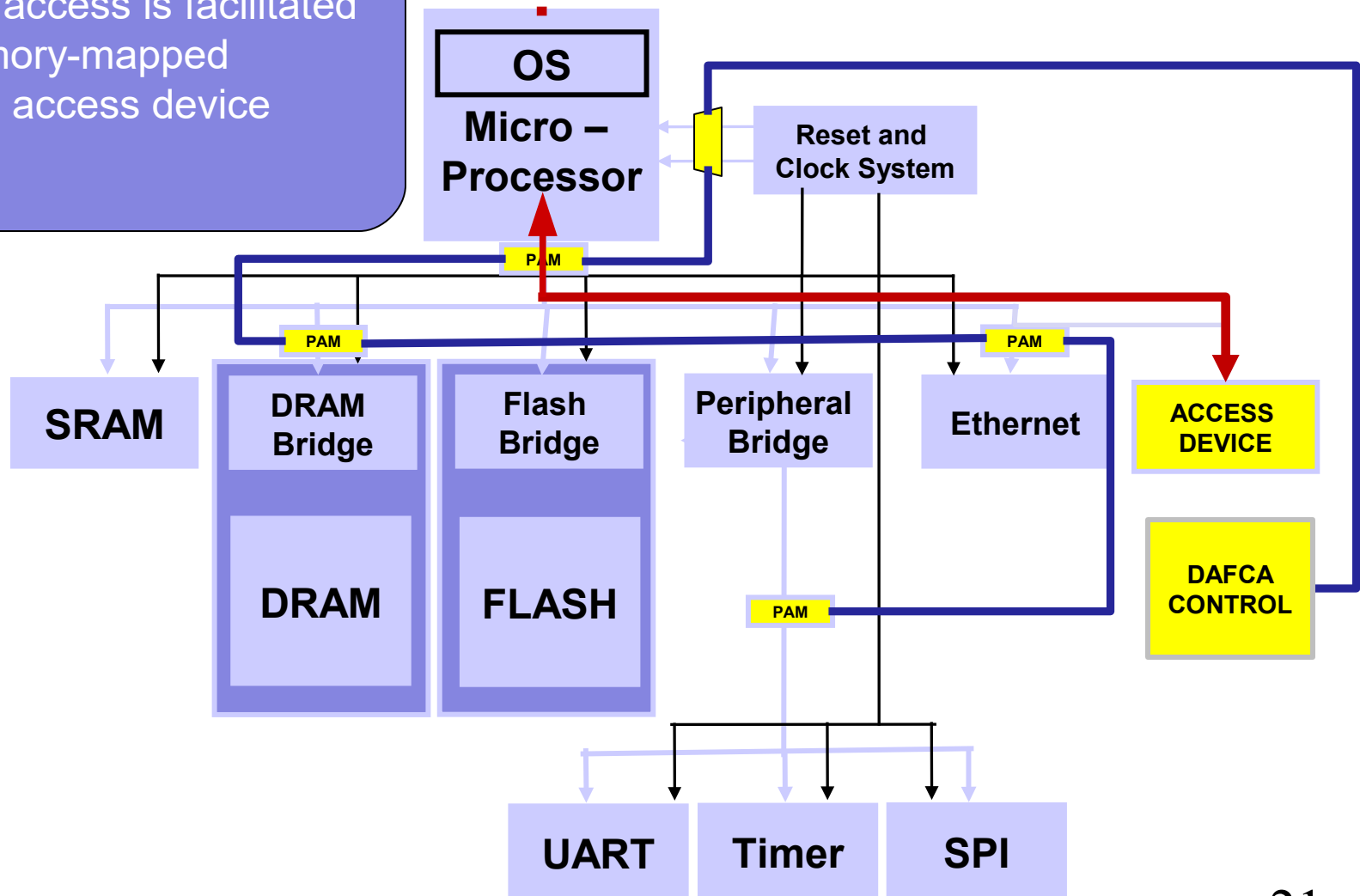
A DAFCA-Enabled System

Software access is facilitated by a memory-mapped hardware access device.



A DAFCA-Enabled System

Software access is facilitated by a memory-mapped hardware access device



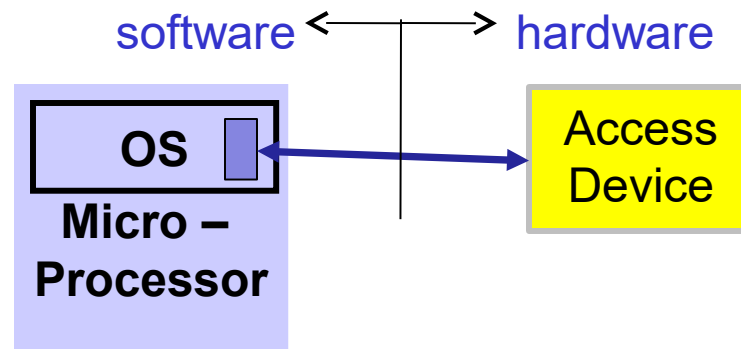
The Software API

- With an understanding of what DAFCA is we can now look at how it can be used by system software
- We will describe how you can:
 - Access basic system information that is always provided by the DAFCA sub-system
 - Customize the data delivered by the DAFCA system “on-the-fly”



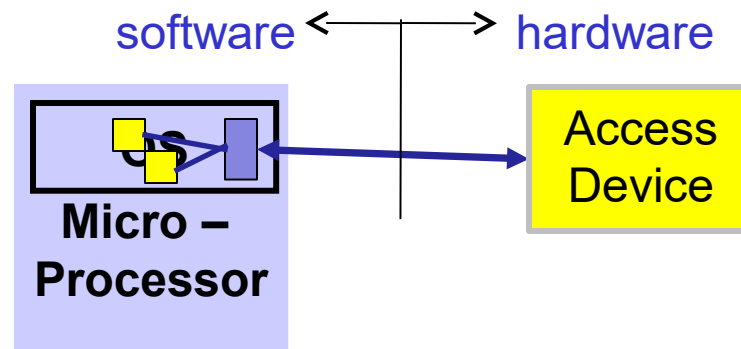
The Software API: **The Module**

- A kernel-level module called the *Security Supervisor* communicates with the *Access Device*
- The two are paired, and only the module is allowed to communicate with the *Access Device*



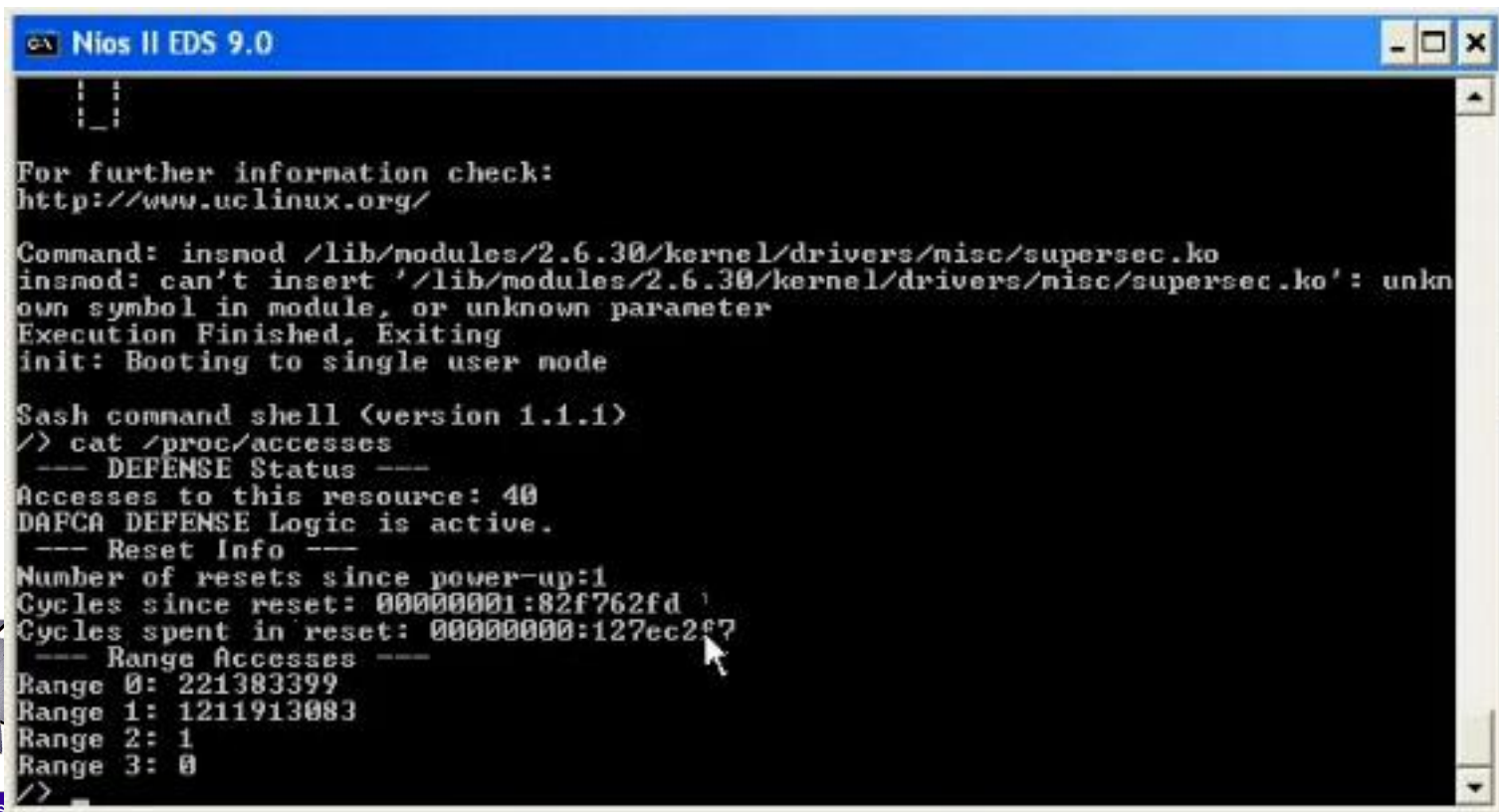
The Software API(cont)

- Higher-level software can attach to a device managed by the module and obtain information from the entire system (not just the microprocessor)



The Software API: Basic

- Under Linux, system state information can be read by opening a virtual file:



```
Nios II EDS 9.0
_
_

For further information check:
http://www.uclinux.org/

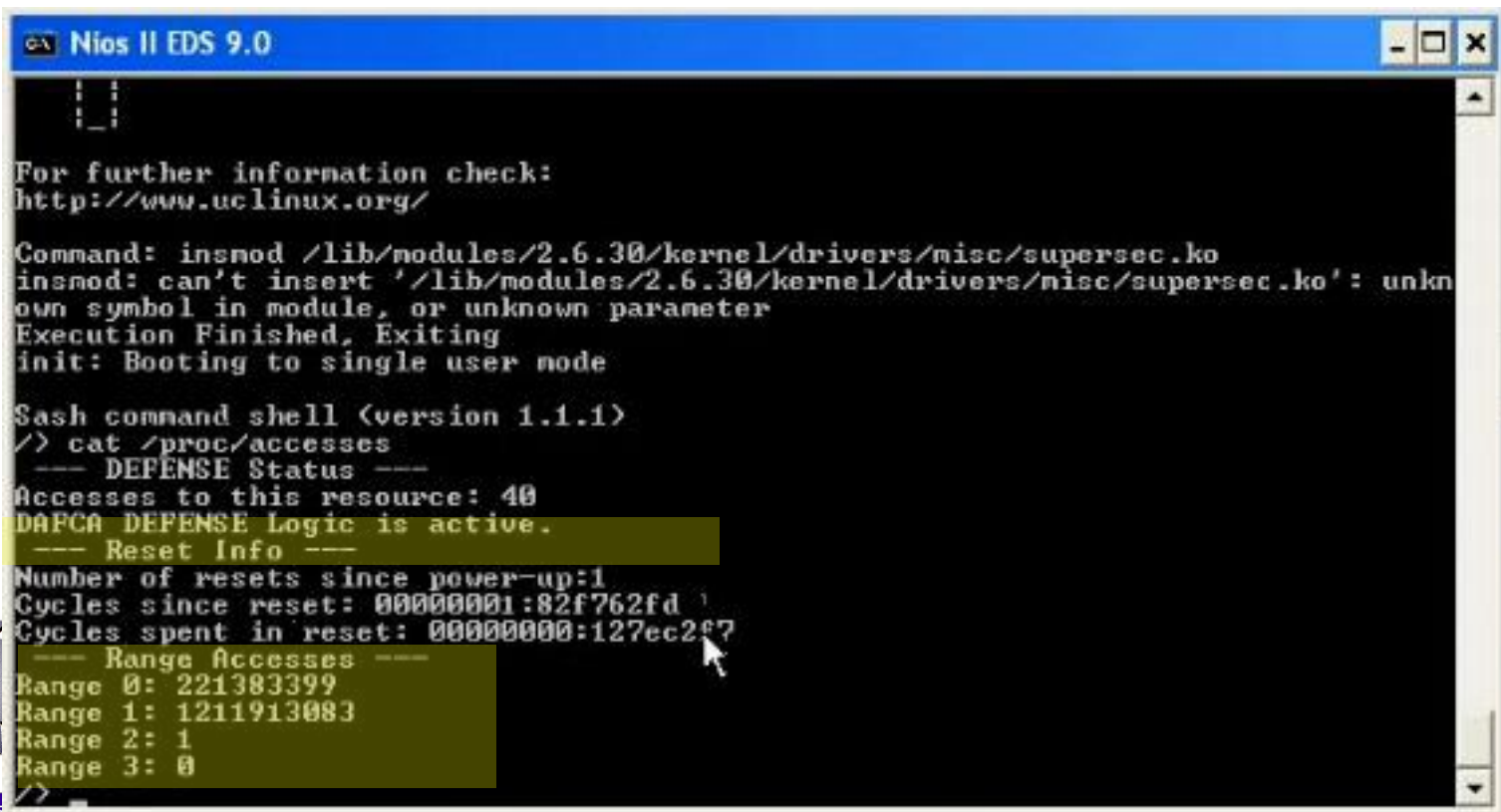
Command: insmod /lib/modules/2.6.30/kernel/drivers/misc/supersec.ko
insmod: can't insert '/lib/modules/2.6.30/kernel/drivers/misc/supersec.ko': unkn
own symbol in module, or unknown parameter
Execution Finished, Exiting
init: Booting to single user mode

Sash command shell (version 1.1.1)
/> cat /proc/accesses
--- DEFENSE Status ---
Accesses to this resource: 40
DAFCA DEFENSE Logic is active.
--- Reset Info ---
Number of resets since power-up:1
Cycles since reset: 00000001:82f762fd
Cycles spent in reset: 00000000:127ec2f7
--- Range Accesses ---
Range 0: 221383399
Range 1: 1211913083
Range 2: 1
Range 3: 0
/>
```



The Software API: Basic

- Basic information provided includes:
 - Number of system resets since power-on
 - Raw accesses to pre-defined memory ranges



```
GA Nios II EDS 9.0

For further information check:
http://www.uclinux.org/

Command: insmod /lib/modules/2.6.30/kernel/drivers/misc/supersec.ko
insmod: can't insert '/lib/modules/2.6.30/kernel/drivers/misc/supersec.ko': unkn
own symbol in module, or unknown parameter
Execution Finished, Exiting
init: Booting to single user mode

Sash command shell (version 1.1.1)
/> cat /proc/accesses
--- DEFENSE Status ---
Accesses to this resource: 40
DAPCA DEFENSE Logic is active.
--- Reset Info ---
Number of resets since power-up:1
Cycles since reset: 00000001:82f762fd
Cycles spent in reset: 00000000:127ec2f7
--- Range Accesses ---
Range 0: 221383399
Range 1: 1211913083
Range 2: 1
Range 3: 0
/>
```



The Software API: Session-based

- An alternate interface is session-based
 - A device is opened by a calling program
 - Information is obtained piecemeal by reading and writing to the device
 - Example (pseudo-code):

```
int value = 0;
handle = open("/dev/secure", "r+w");
write(handle, "get ddr0_throughput")
read(handle, &value)
printf("DDR0 throughput:%d\n", value)
```



The Software API: Access Tracking

- The module can restrict access such that:

One session is allowed per power-on (works even through system RESET)



Because the underlying DAFCA Control hardware records all accesses, it can also record, and limit the number of times a higher-level program opens the device.

The Software API: Access Tracking

- The module can restrict access such that:

One session is allowed per power-on (works even through system RESET)

Or

Many sessions can be opened, but only one session is allowed at a time

Each session must be individually authenticated.



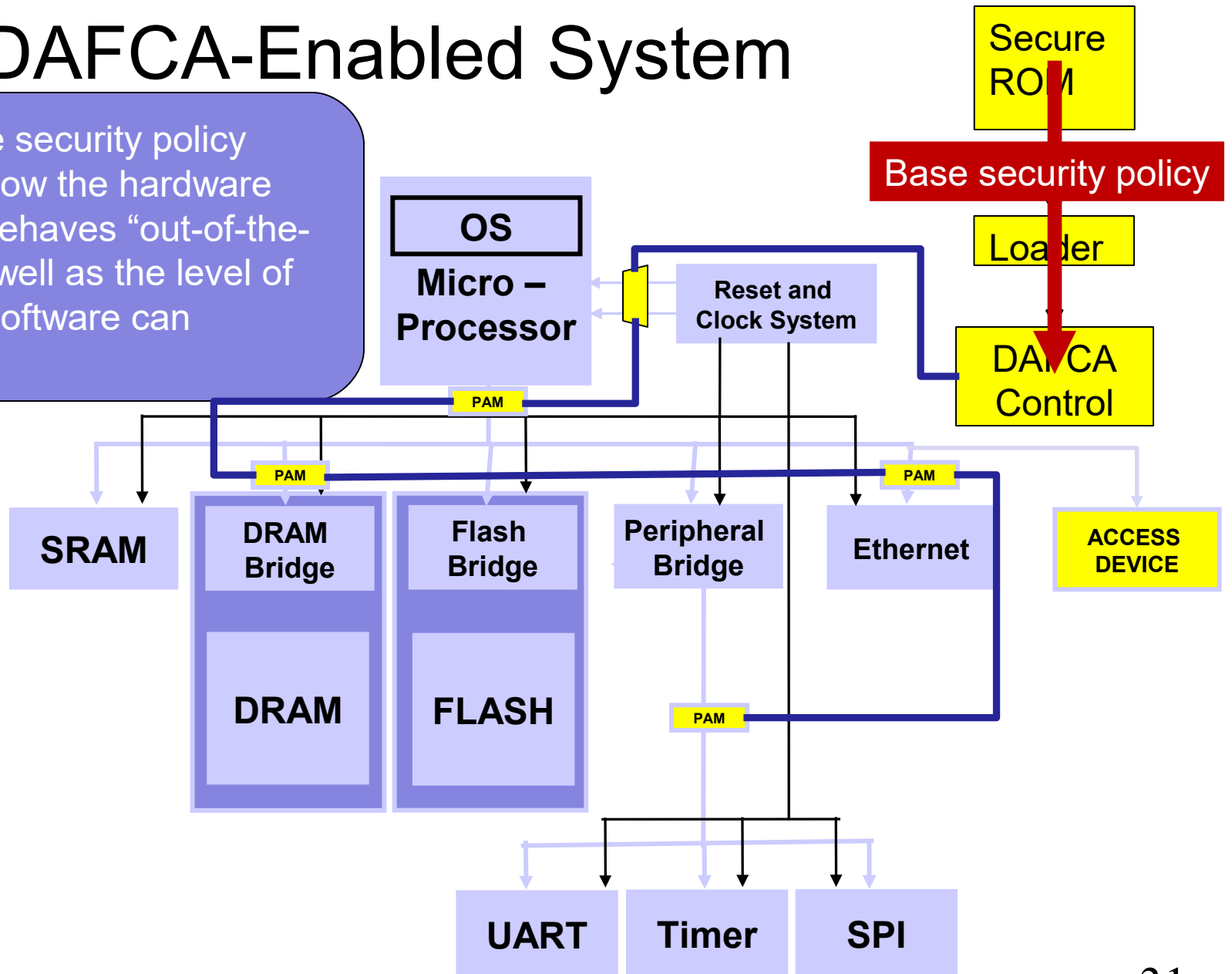
The Software API: **Modifying DAFCA Behavior**

- A limited amount of control of the underlying system can be granted to the software
- The underlying system is inherently flexible and can change focus or mission rapidly
- The hardware system is programmed with a *base security policy*, that is loaded from an encrypted ROM at power-on



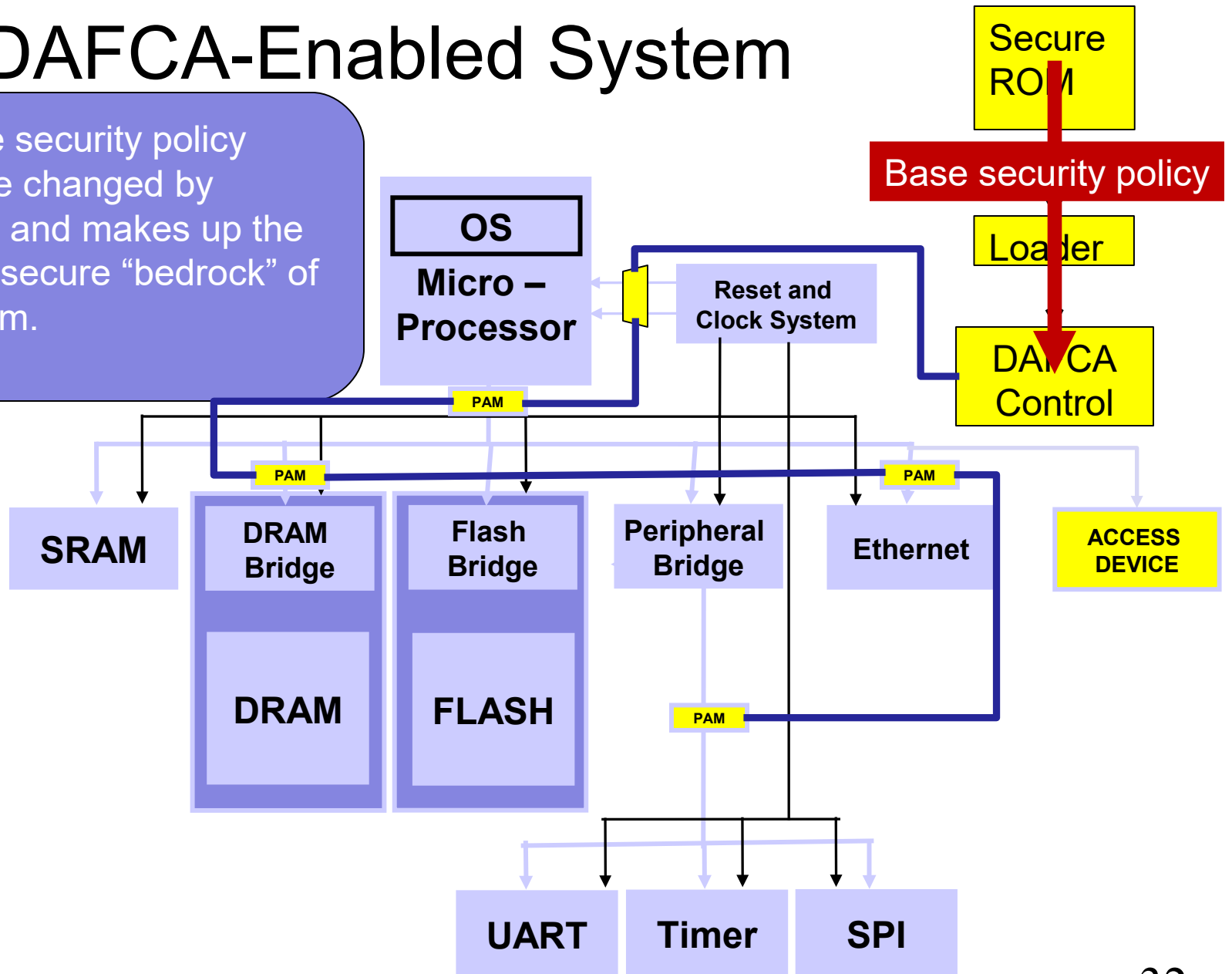
A DAFCA-Enabled System

The base security policy defines how the hardware system behaves “out-of-the-box”, as well as the level of control software can exercise.



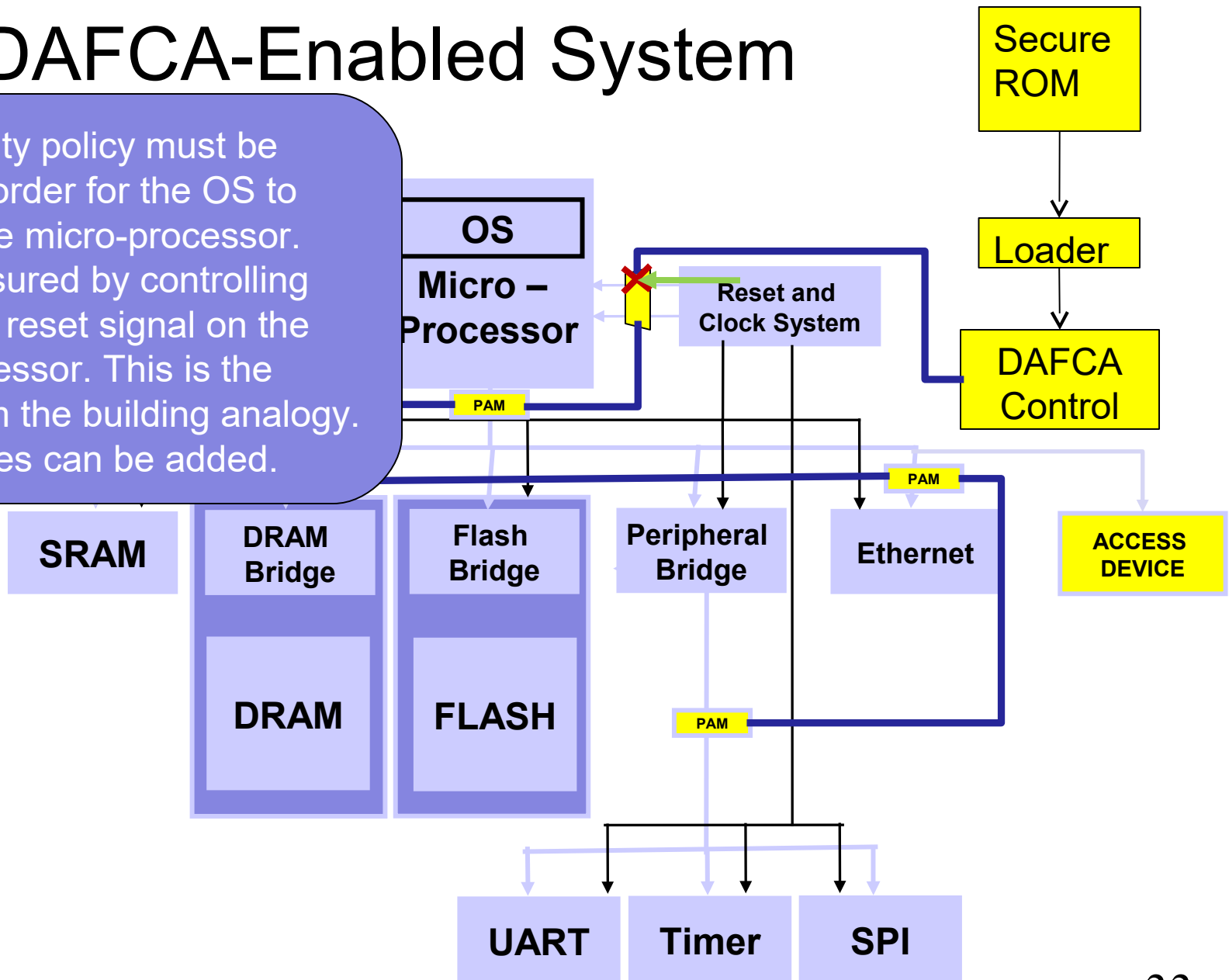
A DAFCA-Enabled System

The base security policy cannot be changed by software, and makes up the invariant secure “bedrock” of the system.



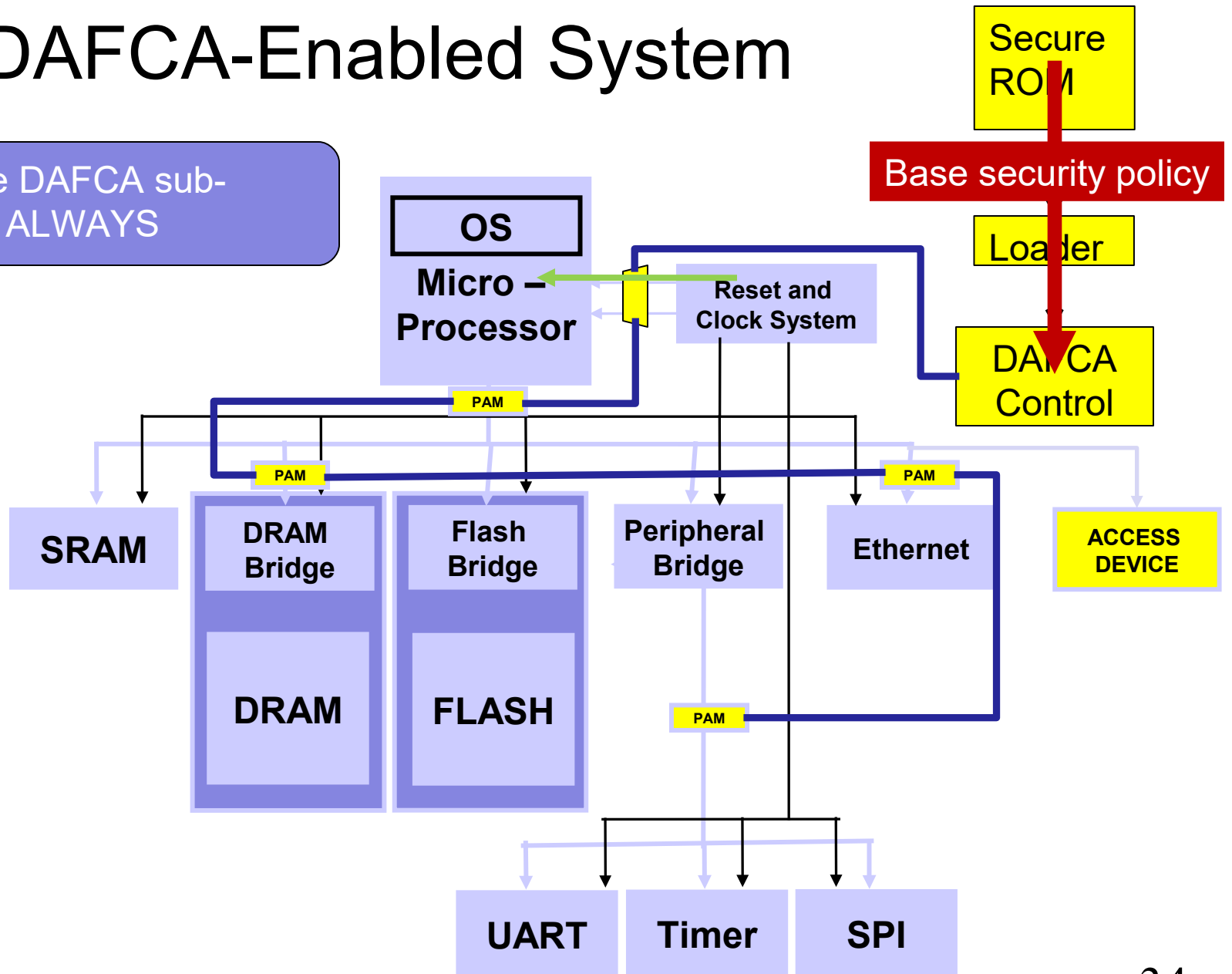
A DAFCA-Enabled System

The security policy must be loaded in order for the OS to boot on the micro-processor. This is ensured by controlling the critical reset signal on the microprocessor. This is the “gate” from the building analogy. Other gates can be added.



A DAFCA-Enabled System

Thus, The DAFCA sub-system is ALWAYS watching!



The Software API: **Function Libraries**

- On-the-fly modification of DAFCA functionality is done with opaque library objects
 - Encrypted sets of DAFCA commands that are created at a secure facility
 - Tightly bound to the hardware, thus VERY hard to spoof, even if the encryption is compromised
 - Functions can be stored inside of DAFCA for added security



The Software API: Function Libraries

- Using the session-based API to load a precompiled security function (pseudocode):

```
int value = 0;  
handle = open("/dev/secure", "r+w");  
write(handle, "load f45792")
```

The invoked functionality is concealed. This function could be specified by a local file, or it could be inside of DAFCA (loaded along with the *base security policy*).



The Software API: **Library Example**

- Using the API and Libraries, one could set a *Resource Trap* on an address range: (0x80:0x880):
 - Count accesses to the specified address range
 - Notify (interrupt) on access to the specified address range (this works if the DAFCA system can drive an irq)
 - Hash address and data values from accesses to the specified range to verify consistent configuration



The Software API: Providing Code

- DAFCA can also supply code to the running system
- Using memory shadowing technology, any region of code in the system can be temporarily or permanently “overlaid” with code or data specified in the *base security policy* set
- This is useful for :
 - On-the-fly modification of authentication code
 - Storage of the DAFCA module code
 - Secure software update



DAFCA: Value

- DAFCA is a distributed hardware-based monitoring and control fabric that is “baked into” your system and is optionally accessible to software via a kernel-level software module
- DAFCA can monitor access patterns and take numerous real-time actions such as blocking access to critical/protected system resources or wiping sensitive data
- DAFCA is flexible and its functionality can be securely modified via software at run-time, making DAFCA-enabled systems resilient to emerging threats





Sawblade Ventures, LLC

Austin, Texas

END

Accessing Sawblade
DAFCA
from Software