Software crítico para sistemas espaciales

2024/25

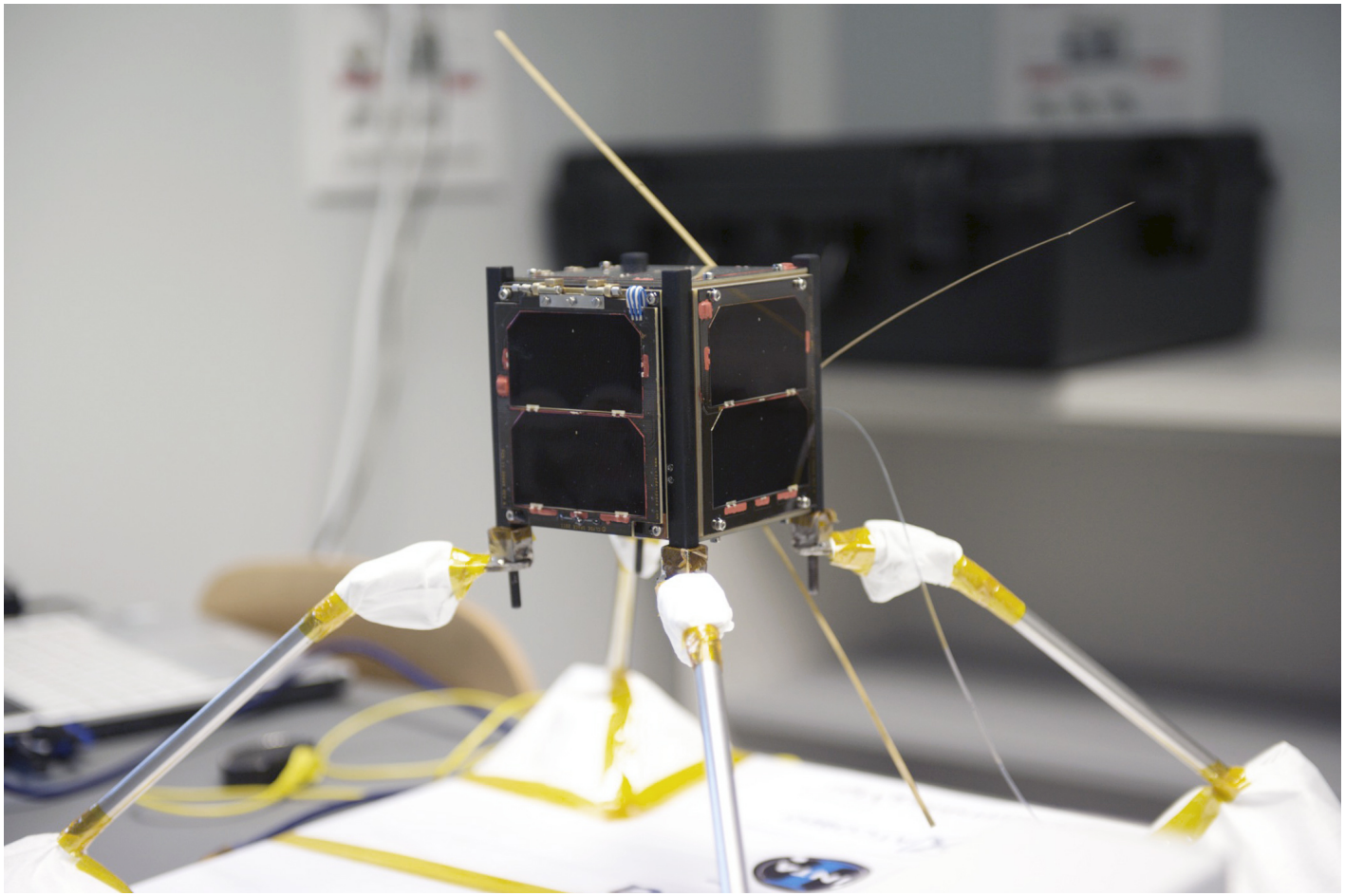
Master Universitario en Enxeñaría Aeroespacial

Arno Formella
Miguel Ramón González Castro
Manuel Pérez Cota

Departamento de Informática
Universidade de Vigo

24/25

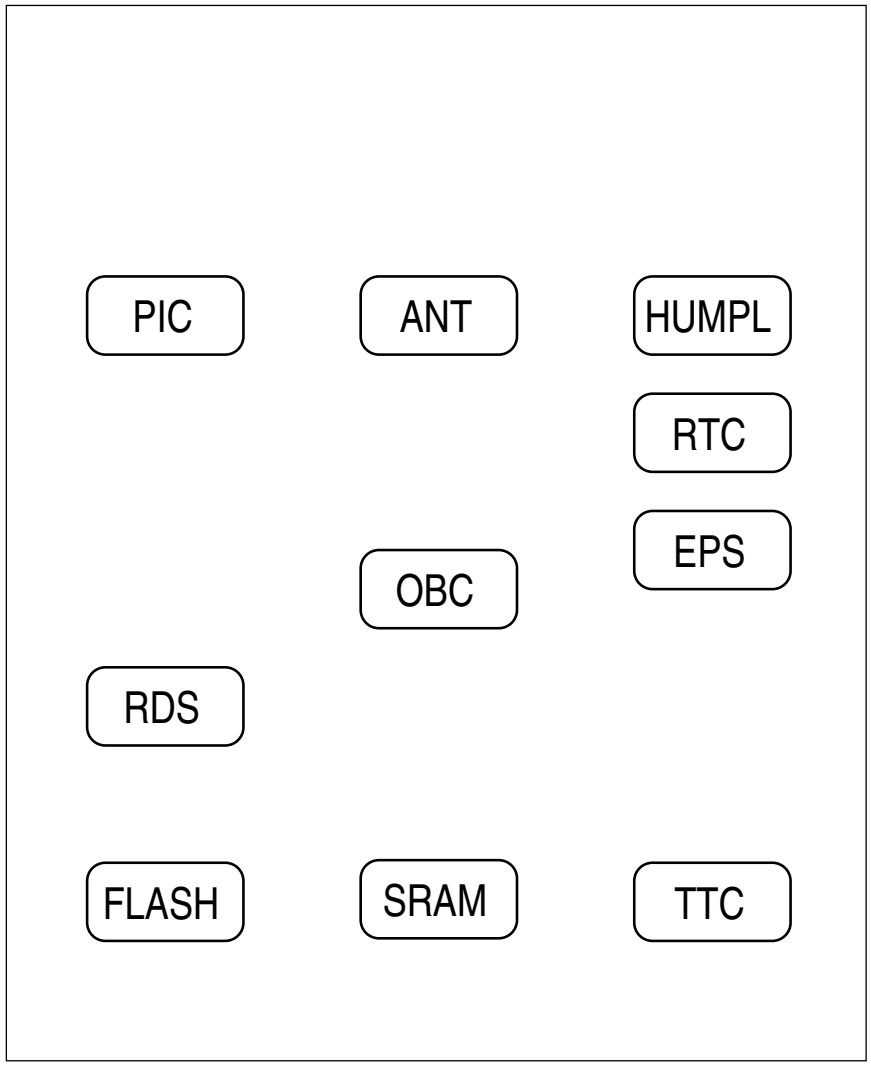
el hardware de XaTcobeo



tres misiones exitosas

- XaTcobeo
  lanzado 14 febrero 2012
- HumSat-D
  lanzado 21 noviembre 2013
- Serpens-B
  lanzado 10 agosto 2015

HumSAT-Demonstrator: punto de vista desde software



- Electric Power System
- On-Board Computer
- Tracking, TeleCommand
- FLASH memory (528 MiByte)
- SRAM memory (1 MiByte)
- Real Time Clock
- Radio Dose Sensor
- Power Integrated Control
- HumSAT PayLoad
- Antenna

especificación de requisitos software

- requisitos generales
- requisitos funcionales
- requisitos de rendimiento
- requisitos de interfaces
- requisitos operacionales
- requisitos de recursos
- requisitos de diseño
- requisitos de seguridad y privacidad
- otros requisitos

especificación de requisitos software: unos ejemplos

de ingeniería de sistemas (¿qué?)
a ingeniería de software (¿cómo?)

requisitos generales:

- debe actuar como aplicación autónomo una vez haberse iniciado
- debe ejecutar en el ordenador de abordo
- debe gestionar las cargas útiles
- etc.

especificación de requisitos software: unos ejemplos

requisitos funcionales:

- debe esperar media hora antes de activar la radio
- debe desplegar la antena
- debe coleccionar datas de la gestión interna (*housekeeping data*) de los subsistemas y de las cargas útiles
- debe ejecutar un planificador de tareas con operaciones programadas
- etc.

especificación de requisitos software: unos ejemplos

requisitos de interfaces:

- debe interactuar con el TTC (radio) via el bus IIC
- debe interactuar con el EPS (potencia) via el bus IIC
- debe interactuar con con el RDS (medidor de radiación) via la interfaz ADC
- debe interactuar con la estación terrena via la interfaz de telecommandos(TC)/telemetría(TM)
- etc.

especificación de requisitos software: unos ejemplos

requisitos operacionales:

- debe operar los modos de operaciones en una máquina de estados
- debe resetear el perro de guardía de forma periódica
- debe apagarse de forma periódica antes del apagado en hardware
- debe alternar modo RX (recepción) y modo TX (emisión) durante las comunicaciones
- etc.

especificación de requisitos software: unos ejemplos

requisitos de recursos:

- el software y sus datos deben caber en la memoria disponible
- el software de arranque debe caber en la memoria disponible para tal fin
- etc.

especificación de requisitos software: unos ejemplos

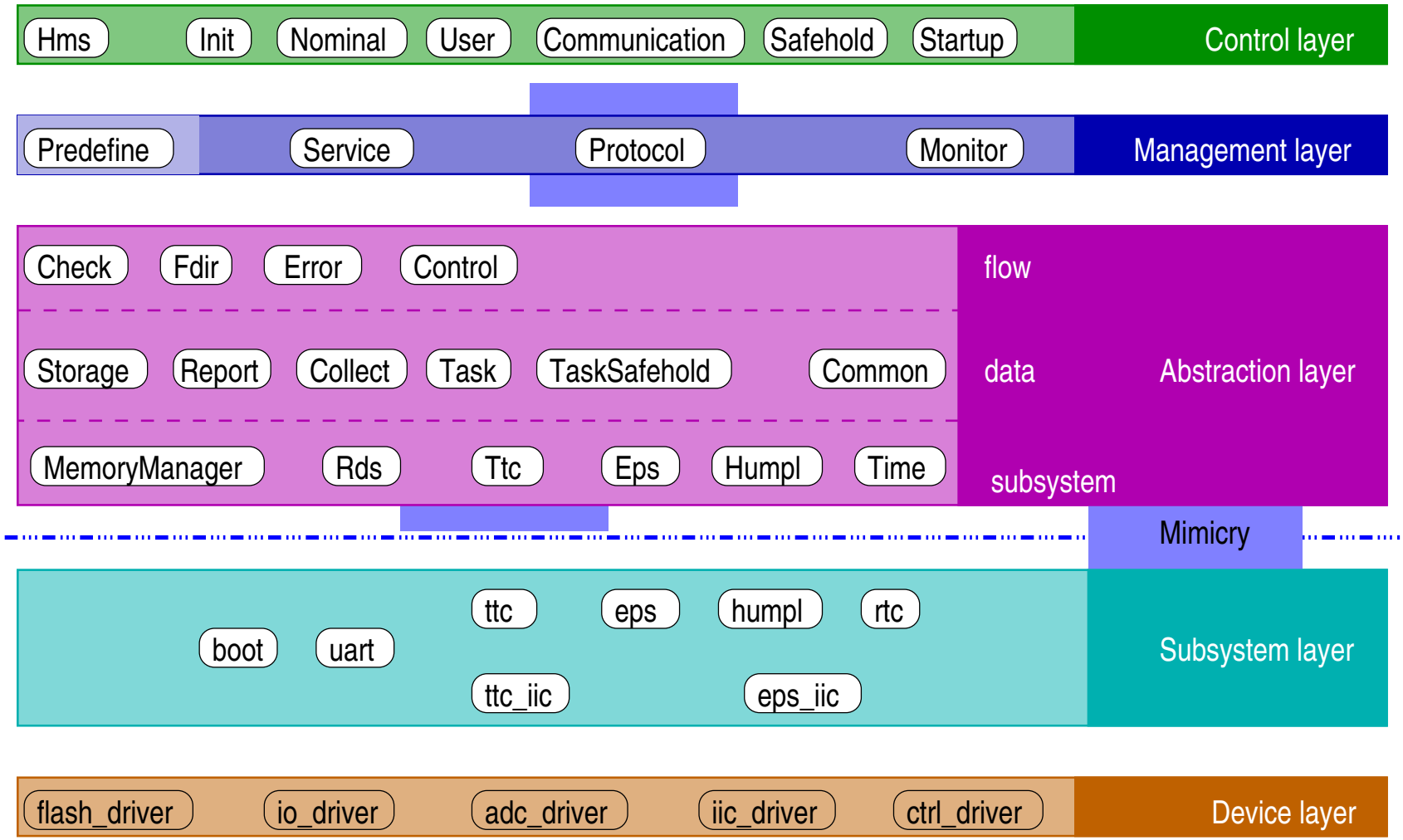
requisitos de diseño:

- debe estar escrito en C
- debe estar compatible y desarrollado en la EDK de Xilinx
- debe estar escrito de forma modular y permitir la integración y actividades de pruebas
- etc.

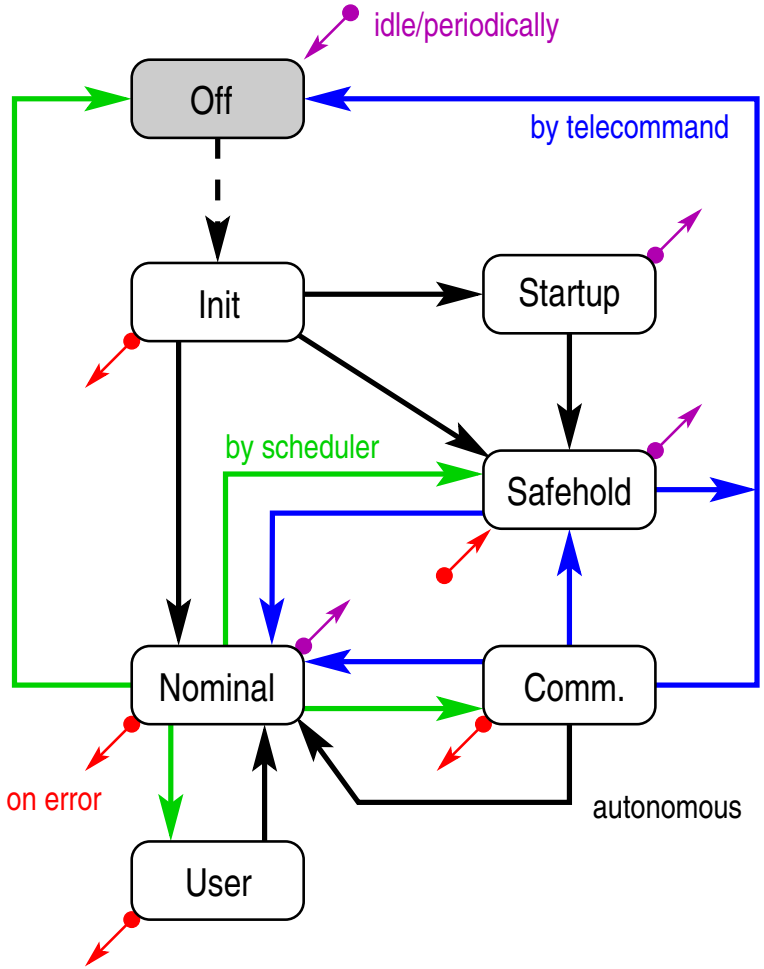
decisiones de diseño de software

- máquina de estados finitos a nivel cima
- sin recursión
- sin concurrencia
  (menos aquella impuesta por el hardware)
- sin gestión de memoria dinámica

diseño del software en capas



modos de operaciones

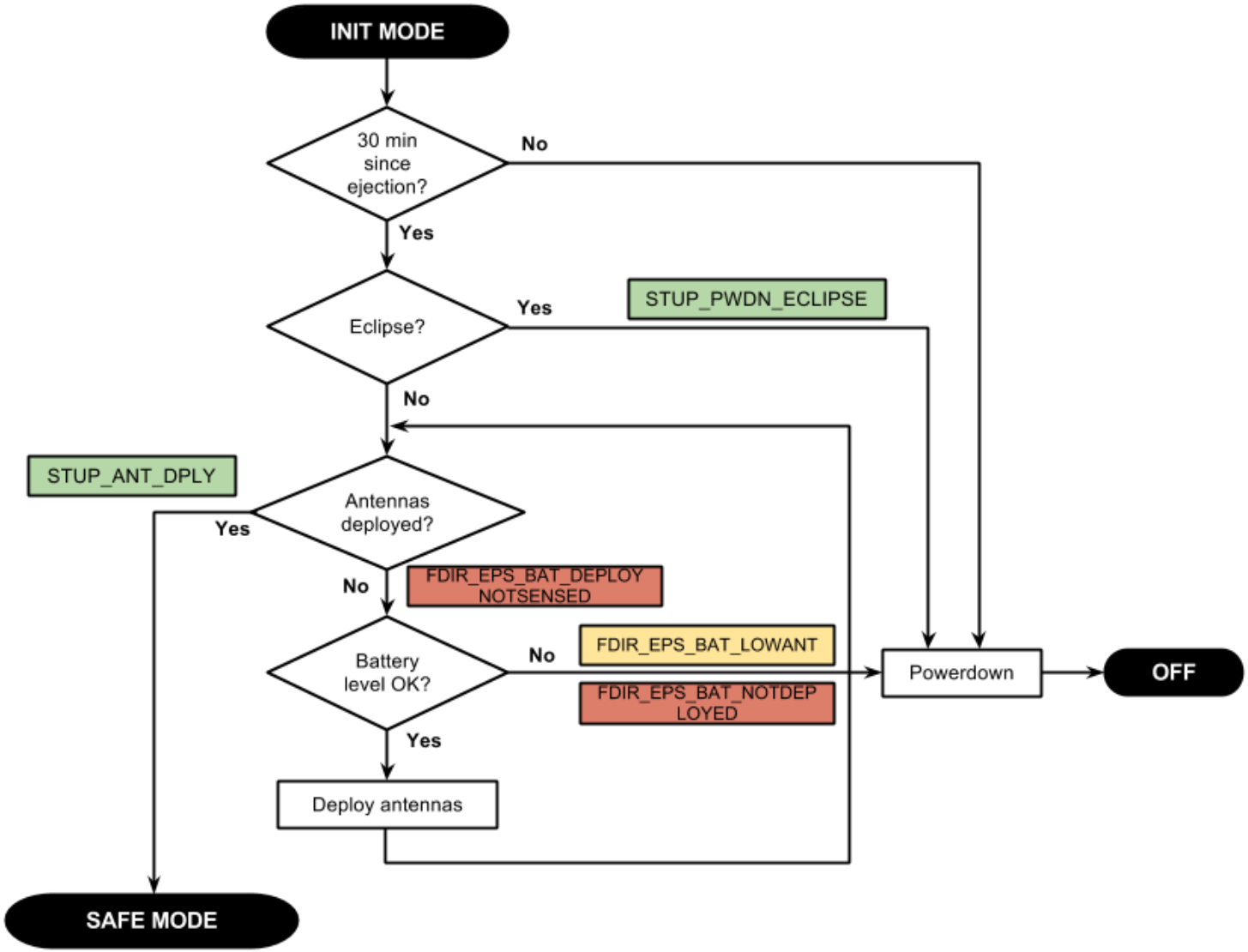


- capa más alta es máquina de estados
- 6 modos principales
- tipos diferentes de transiciones
  - autónomo via flujo de control
  - via telecomando
  - via tarea planificada
  - debido a error
  - periódico o apagado por inactividad

máquina de estados, bucle principal

```
1   int main(
2   ) {
3     ObswReset();
4     Init();
5       // It must be certain that we arrive here without a watchdog
6       // double rollover.
7       // If this was happening, we got an infinite loop.
8
9       // The top level of the state machine performing the transitions
10      // between modes.
11    while(1) {
12        // Make sure ControlHandleNextTask() or similar is called
13        // in all loops of all modes.
14      ControlHandleNextTaskNoEclipse();
15      switch(system_variable.current_mode) {
16        case MODE_STARTUP: Startup(); break;
17        case MODE_NOMINAL: Nominal(); break;
18        case MODE_OPERATION: Operation(); break;
19        case MODE_COMMUNICATION: Communication(); break;
20        default:
21          ControlModeChange(MODE_SAFEHOLD);
22          ErrorProgramFlow(XEC_OBSW_MAIN_MODE);
23          // Fall through is on purpose.
24        case MODE_SAFEHOLD: Safehold(); break;
25      }
26    }
27      // Infinite loop to wait for powerdown the system.
28    for(;;);
29    return 0;
30  }
```

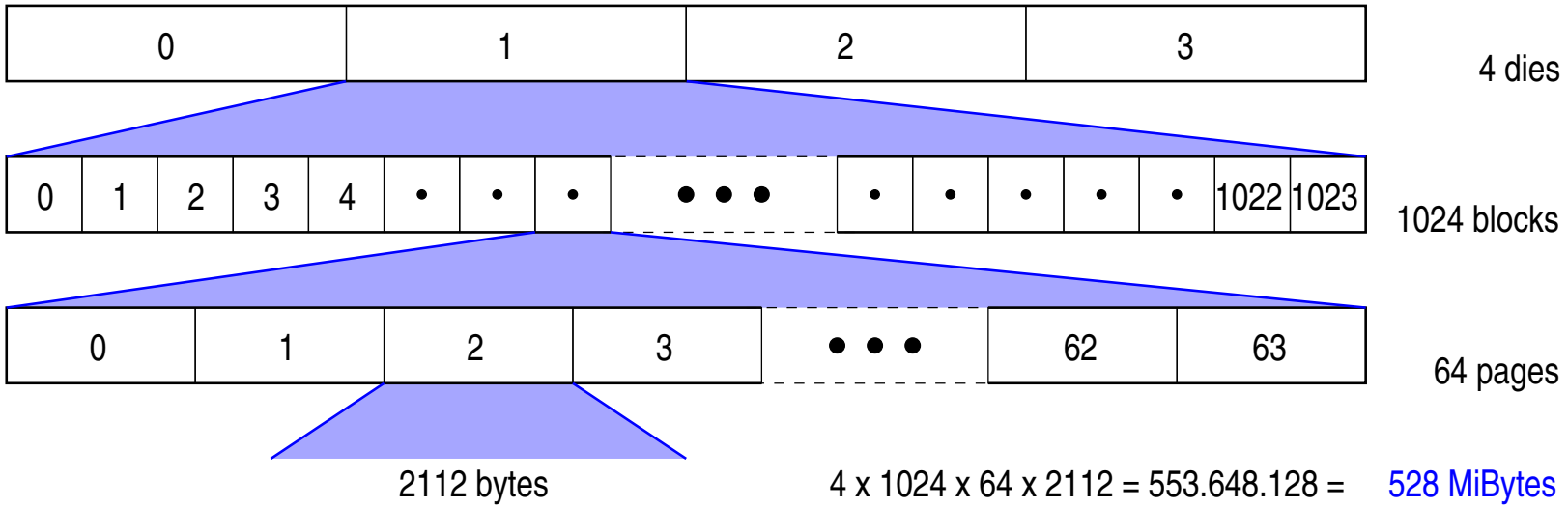
modo *startup*



modo *startup*

```
1  void Startup() {
2    if(!MemManAntennaIsUnDeployed()) {
3      MemManMarkFirstHalfHourPassed();
4      MemManMarkFirstAntennaDeployDone();
5      MemManMarkAntennaDeployed();
6    }
7    else if(!ControlAntennaIsDeployed()) {
8      if( /* WithinFirstHalfHour */ )) {
9        StartupWaitFirstHalfHour();// may powerdown
10     }
11     ControlHandleEclipse();       // may powerdown
12     StartupDeployAntenna();
13   }
14   ControlModeChange(MODE_SAFEHOLD);
15 }
```

memoria FLASH: pastillas, bloques, páginas, bytes ...



- FLASH con tecnología de puertas NAND
- solo se puede borrar bloques enteros (todo con 0xFF)
- no puede escribir secuencias de datos que pasan de una página a otra
- solo se puede escribir bits de 0
- solo se puede borrar cierto número de veces (*wear out*)

almacenamiento de datos en la FLASH

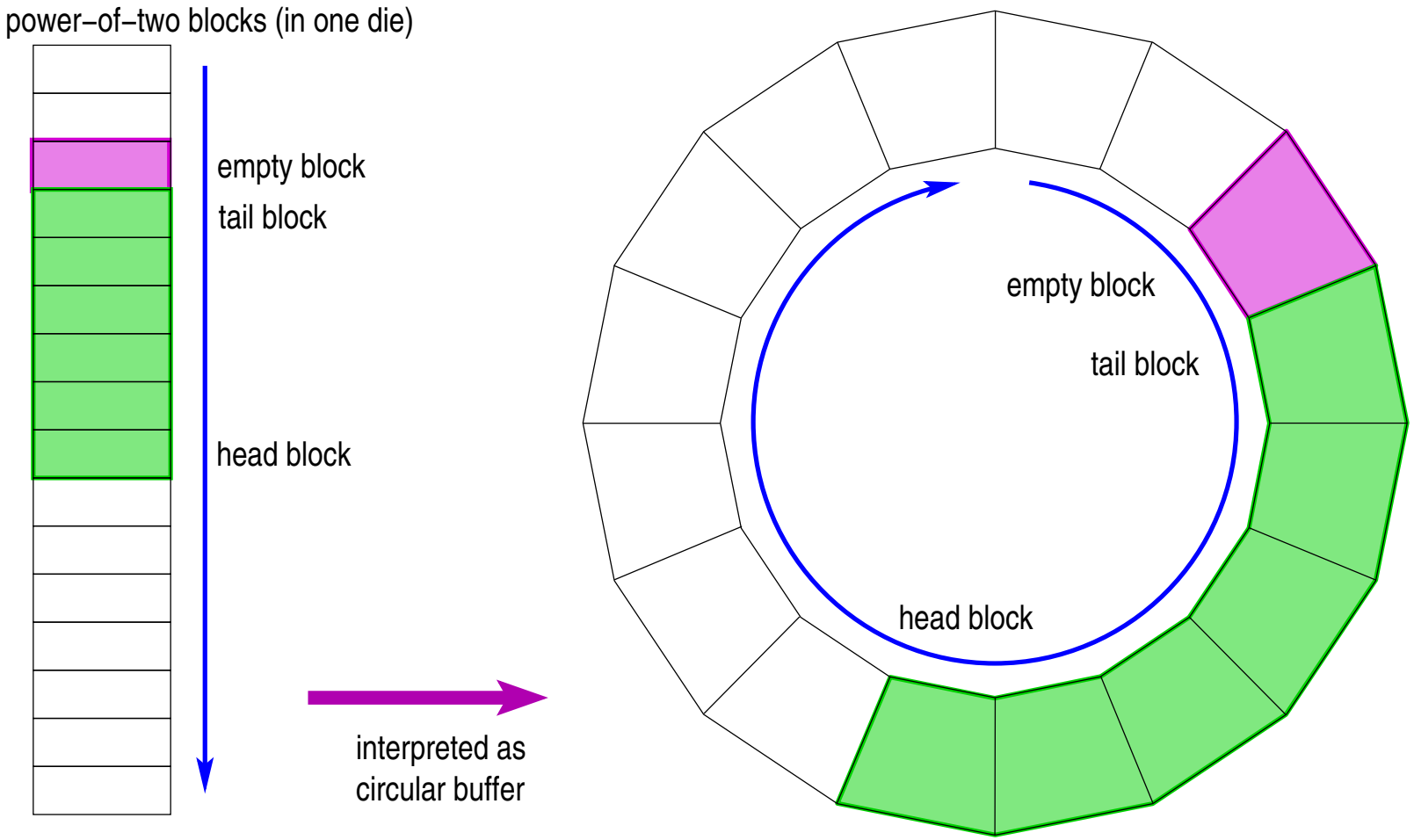
la siguiente información está almacenada en la FLASH en bloques dedicados:

- informes (telemetría) que fueron generados por razones diversas
  por ejemplo, por telecomandos, eventos, datos de la carga útil
- definiciones de informes y eventos por generar
- parametros por usar en la colección de datos internos (o bien de forma periódica o bien por demanda directa)
  dichos valores de los parámetros pueden ser monitorizados
- estado el sistema para reiniciar después de un apagado (incluye el planificador de tareas)

estructura del almacenamiento en la FLASH memory

reports:	en un búfer circular, un invento mio (con 3 punteros como información del estado9
definitions:	en bloques modificables e inicializables
parameters:	en un conjunto de pares de bloques
system state:	en un doble búfer controlado por máquina finita de estados, otro invento mio (contiene constantes y variables de la configuración/estado actual)

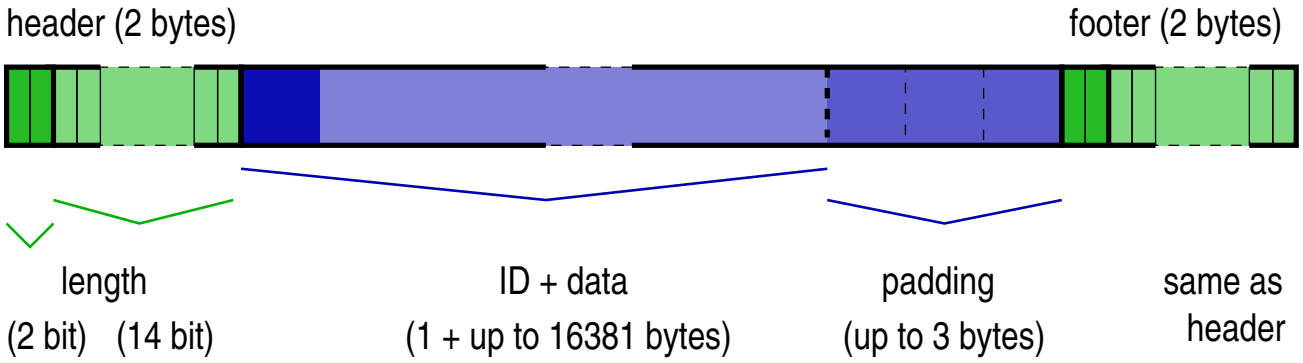
búfer circular para almacenar telemetría



búfer circular: invariantes básicas

- número de bloques como potencia de dos
  (permite cálculo de indexación mñas simple)
- por lo menos un bloque vacio
- bloques ocupados están consecutivos
- cada bloque tiene marca indicando en uso o libre
- mantenimiento con:
  - puntero cabeza (*head*):
    apunta a marca de bloque de principio
  - puntero cola (*tail*):
    apunta a marca de bloque de final
  - puntero cima (*top*):
    apunta a primera entrada en bloque de principio
- los punteros *head/tail/top* se guardan en el estado del sistema

entradas del búfer circular



- encabezamiento y pie codifican la longitud de los datos más relleno
- una entrada vacia ocupa 4 bytes
  (o bien cabe o bien entrada anterior usa relleno)
- permita búsquedas en ambas direcciones con la información en los encabezamientos y pies
- 0xFFFF encabezamiento (o pie) significa no-ocupado
- entradas con identificador 0 son invalidadas

inicialización del búfer circular

- se asume que inicialmente esté vacio (todo 0xFF) a pastilla
- inicializar:
  - pon cabercera y cola a cualquier bloque
  - marca bloque como usado
  - pon cima a primera entrada libre
- auto re-initializar (caberca, cola y cima se perdieron):
  - pon cabecera a cualquer bloque
  - busca saltando bloques en dirección adelante hasta primer bloque usado
  - pon cola ahí
  - busca saltando bloques en dirección adelante hasta primer bloque vacio
  - retrocede con cabecera un bloque
  - busca saltando bloques en dirección hacía atrás hasta primer bloque vacio
  - avanza con cola un bloque
  - busca con cima primer entrada vacía

el búfer circular es robusto, pero hay mejoras...

- encabezamientos y pies tienen que coincidir
- se podría usar SECDED (detección de errores dobles y corrección de errores simples) código para encabezamientos y pies (p.ej., código de Hamming (16,11,4), permitiría tamaños de datos hasta 512 bytes)
- se podría usar marcas para bloque inusables (para ignorarlos)
- se podrían inventar métodos más rápidos de búsqueda (quizá con más datos estructurales)

Storage of report definitions

they say what to gather...

reports are either stored in the circular buffer
or transmitted directly to ground (comm and safe mode)

- informes de eventos:
  - se genera un informe cada vez que se alcanza algún punto en el código predefinido
  - hay tres tipos: de progreso, error gravedad baja, error gravedad alta
- informes como respuestas a pedidos del servicio de control. es decir, telemetría generada directamente a causa de telecomandos
- informes de datos de gestión interna y de datos diagnósticos (periódicamente almacenados)
- informes de la carga útil HumSAT (dos tipos)
- monitorización de informes de alerta

informes de eventos

- 230 (posible hasta 254) definiciones de informes de eventos
- se pueden activar y desactivar
  (truco de contar ceros, otro invento mio)
- se generan en puntos específicos del programa
- campo de bits de tamaño 64 para codificar parámetros requeridos

- idendificador (ID) del evento
- sello de tiempo del evento
- valores de los parámetros requeridos
- tamaño es implícito

informes de gestión interna y de diagnosis

- 20 (hasta 127) definiciones de informes de diagnosis (se ejecutan en el planificador del estado seguro)
- 19 (hasta 127) definiciones de informes de gestión interna (se ejecutan en el planificador del estado nominal)
- se pueden activar y desactivar
- se generan de vez en cuando (auto-repetición en el planificador)
- campo de bits de tamaño 64 para codificar parámetros requeridos
- los parámeters están incluidos en la monitorización

- identificador (ID) del evento
- sello de tiempo del evento
- valores de los parámetros requeridos
- tamaño es implícito