Flash Memory in Embedded Java Programs

Stephan Korsholm
VIA University College
Horsens, Denmark
sek@viauc.dk

JTRES'2011, York, UK

Overview

- Presentation, 20 min

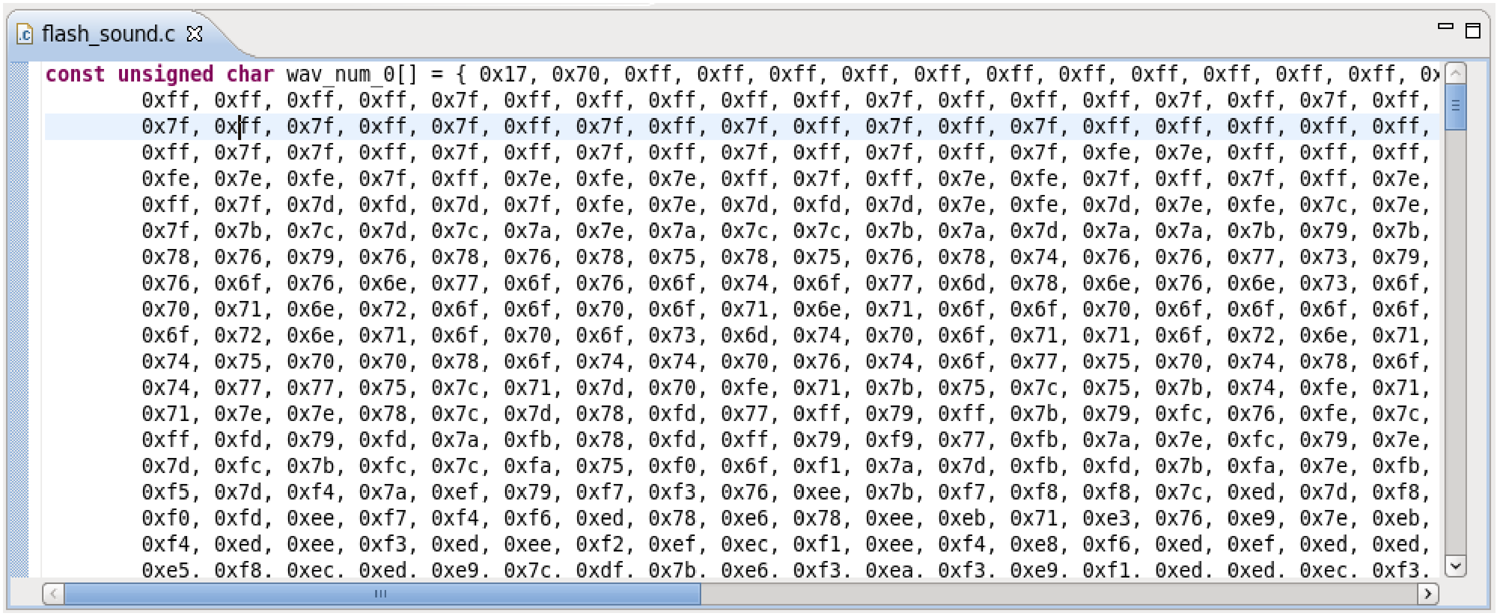
  - Background
  - What is constant data, and why keep it in Flash?
  - Constant data in a JVM
  - Marking and initialization of constant data
  - Accessing constant data
  - Cost of constant data
  - Future work
  - Perspective

- Discussion, 10 min

Background

- Java on small embedded devices
  - > 256 Kb flash
  - < 8 Kb RAM
  - 8/16/32 bit architecture
  - Usually programmed by engineers familiar with C

What is constant data?


flash_sound.c

```
const unsigned char wav_num_0[] = { 0x17, 0x70, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x
        0xff, 0xff, 0xff, 0xff, 0x7f, 0xff, 0xff, 0xff, 0xff, 0xff, 0x7f, 0xff, 0xff, 0xff, 0x7f, 0xff, 0x7f, 0xff,
        0x7f, 0xff, 0x7f, 0xff, 0x7f, 0xff, 0x7f, 0xff, 0x7f, 0xff, 0x7f, 0xff, 0x7f, 0xff, 0xff, 0xff, 0xff, 0xff,
        0xff, 0x7f, 0x7f, 0xff, 0x7f, 0xff, 0x7f, 0xff, 0x7f, 0xff, 0x7f, 0xff, 0x7f, 0xfe, 0x7e, 0xff, 0xff, 0xff,
        0xfe, 0x7e, 0xfe, 0x7f, 0xff, 0x7e, 0xfe, 0x7e, 0xff, 0x7f, 0xff, 0x7e, 0xfe, 0x7f, 0xff, 0x7f, 0xff, 0x7e,
        0xff, 0x7f, 0x7d, 0xfd, 0x7d, 0x7f, 0xfe, 0x7e, 0x7d, 0xfd, 0x7d, 0x7e, 0xfe, 0x7d, 0x7e, 0xfe, 0x7c, 0x7e,
        0x7f, 0x7b, 0x7c, 0x7d, 0x7c, 0x7a, 0x7e, 0x7a, 0x7c, 0x7c, 0x7b, 0x7a, 0x7d, 0x7a, 0x7a, 0x7b, 0x79, 0x7b,
        0x78, 0x76, 0x79, 0x76, 0x78, 0x76, 0x78, 0x75, 0x78, 0x75, 0x76, 0x78, 0x74, 0x76, 0x76, 0x77, 0x73, 0x79,
        0x76, 0x6f, 0x76, 0x6e, 0x77, 0x6f, 0x76, 0x6f, 0x74, 0x6f, 0x77, 0x6d, 0x78, 0x6e, 0x76, 0x6e, 0x73, 0x6f,
        0x70, 0x71, 0x6e, 0x72, 0x6f, 0x6f, 0x70, 0x6f, 0x71, 0x6e, 0x71, 0x6f, 0x6f, 0x70, 0x6f, 0x6f, 0x6f, 0x6f,
        0x6f, 0x72, 0x6e, 0x71, 0x6f, 0x70, 0x6f, 0x73, 0x6d, 0x74, 0x70, 0x6f, 0x71, 0x71, 0x6f, 0x72, 0x6e, 0x71,
        0x74, 0x75, 0x70, 0x70, 0x78, 0x6f, 0x74, 0x74, 0x70, 0x76, 0x74, 0x6f, 0x77, 0x75, 0x70, 0x74, 0x78, 0x6f,
        0x74, 0x77, 0x77, 0x75, 0x7c, 0x71, 0x7d, 0x70, 0xfe, 0x71, 0x7b, 0x75, 0x7c, 0x75, 0x7b, 0x74, 0xfe, 0x71,
        0x71, 0x7e, 0x7e, 0x78, 0x7c, 0x7d, 0x78, 0xfd, 0x77, 0xff, 0x79, 0xff, 0x7b, 0x79, 0xfc, 0x76, 0xfe, 0x7c,
        0xff, 0xfd, 0x79, 0xfd, 0x7a, 0xfb, 0x78, 0xfd, 0xff, 0x79, 0xf9, 0x77, 0xfb, 0x7a, 0x7e, 0xfc, 0x79, 0x7e,
        0x7d, 0xfc, 0x7b, 0xfc, 0x7c, 0xfa, 0x75, 0xf0, 0x6f, 0xf1, 0x7a, 0x7d, 0xfb, 0xfd, 0x7b, 0xfa, 0x7e, 0xfb,
        0xf5, 0x7d, 0xf4, 0x7a, 0xef, 0x79, 0xf7, 0xf3, 0x76, 0xee, 0x7b, 0xf7, 0xf8, 0xf8, 0x7c, 0xed, 0x7d, 0xf8,
        0xf0, 0xfd, 0xee, 0xf7, 0xf4, 0xf6, 0xed, 0x78, 0xe6, 0x78, 0xee, 0xeb, 0x71, 0xe3, 0x76, 0xe9, 0x7e, 0xeb,
        0xf4, 0xed, 0xee, 0xf3, 0xed, 0xee, 0xf2, 0xef, 0xec, 0xf1, 0xee, 0xf4, 0xe8, 0xf6, 0xed, 0xef, 0xed, 0xed,
        0xe5. 0xf8. 0xec. 0xed. 0xe9. 0x7c. 0xdf. 0x7b. 0xe6. 0xf3. 0xea. 0xf3. 0xe9. 0xf1. 0xed. 0xed. 0xec. 0xf3.
```

	Code	Variable Data	Constant Data
DECT	234KB	7KB (8%)	78KB (92%)
Modbus	171KB	207KB (83%)	41KB (17%)
HVM	22KB	4KB (29%)	10KB (71%)

Table 1: Overview of data usage

Why keep it in flash?

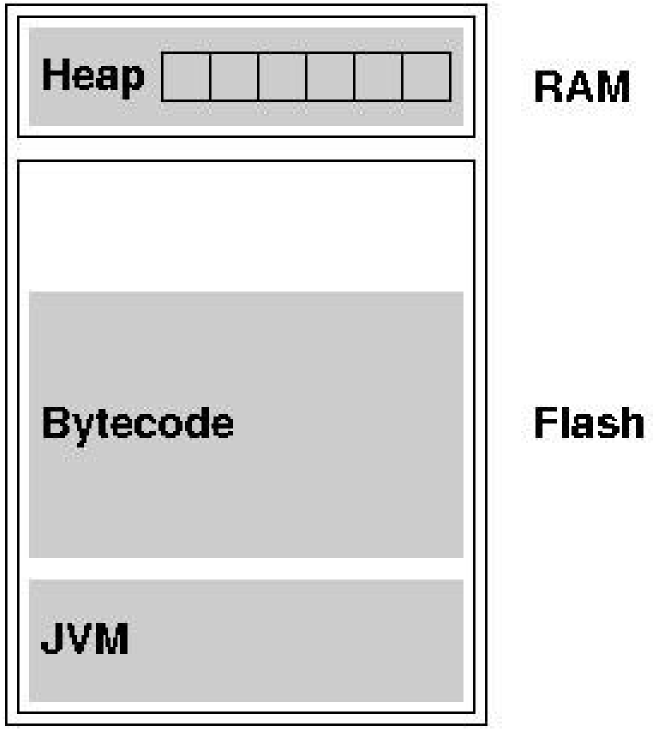
- Grundfos devices
  - Circulation pumps in households
  -
- Polycom devices
  - Wireless DECT handsets
  - Conference systems (formula 1, tall ships racing)

Sold in large quantities

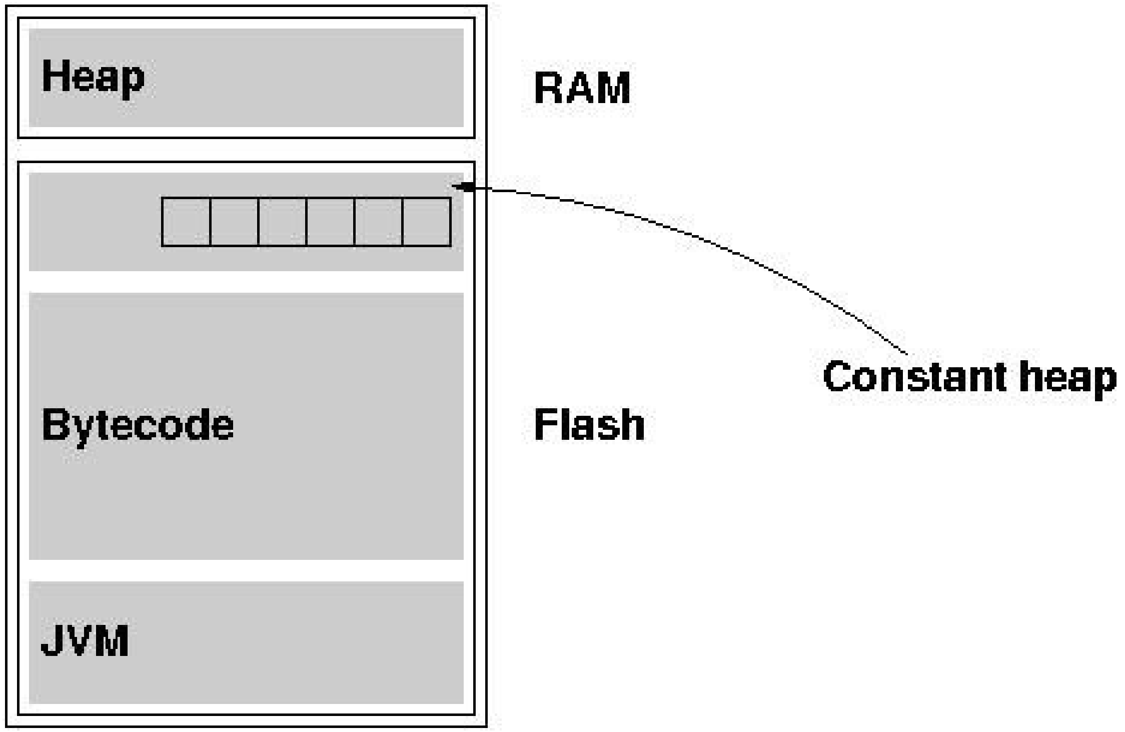
Competitive market

Flash is cheaper

Constant data in a JVM



Constant data in a JVM



Constant data in a JVM

- The HVM is a lean Java VM

  1. Intelligent class linking
  2. SDK independence
  3. OS independence
  4. Simple build procedure (gcc -nostdlib *.c)
  5. HW Objects & 1st level interrupt handling
  6. ROM/RAM aware

Constant data in a JVM

- The HVM is a lean Java VM

  1. Intelligent class linking
  2. SDK independence
  3. OS independence
  4. Simple build procedure (gcc -nostdlib *.c)
  5. HW Objects & 1st level interrupt handling
  6. ROM/RAM aware
  7. Supports constant data in flash

Marking constant data

```
@Flash
private int[] array = { 23, 112, -1, -1};
```

- Explicit marking of constant data
- Using annotations
- Same as in C environments
- Room for improvements

Accessing constant data

```
public class ConstantData extends Object{
public ConstantData();
  Code:
   0:   aload_0
   1:   invokespecial #1; //"<init>":()V
   4:   aload_0
   5:   iconstant_4
   6:   newarray byte
   8:   dup
   9:   iconstant_0
   10: bipush          23
   12: bastore
   13: dup
   14: iconstant_1
   15: bipush          112
   17: bastore
   18: dup
   19: iconstant_2
   20: iconstant_m1
   21: bastore
   22: dup
   23: iconstant_3
   24: iconstant_m1
   25: bastore
   26: putfield        #2; //wav_num_0:[B
   29: return
}
```

```
@Flash
private int[] array = { 23, 112, -1, -1};
```

Accessing constant data

```
public class ConstantData extends Object{
public ConstantData();
  Code:
   0:   aload_0
   1:   invokespecial #1; //"<init>":()V
   4:   aload_0
   5:   iconstant_4
   6:   newarray byte
   8:   dup
   9:   iconstant_0
   10: bipush          23
   12: bastore
   13: dup
   14: iconstant_1
   15: bipush          112
   17: bastore
   18: dup
   19: iconstant_2
   20: iconstant_m1
   21: bastore
   22: dup
   23: iconstant_3
   24: iconstant_m1
   25: bastore
   26: putfield        #2; //wav_num_0:[B
   29: return
}
```

Host handling

Create in constant heap

No changes on host

Because this field is marked as constant!

After initialization

```
package test.icecapvm.minitests;
public class TestVolatile4 {
    private static class ConstantData {
        public volatile int NUM1 = 42;
        public volatile byte[] bytes = { 23, 112, -1, -1 };
        public volatile int NUM2 = 43;
    }

    public static void main(String[] args) {
        ConstantData cdata = new ConstantData();
        devices.System.lockROM();

        if (cdata.NUM1 == 42) {
            if (cdata.bytes != null) {
                if (cdata.bytes.length == 4) {
                    int sum = 0;
                    for (int i = 0; i < 4; i++) {
                        sum += cdata.bytes[i];
                    }
                    if (sum == 133) {
                        if (cdata.NUM2 == 43) {
                            args = null;
                        }
                    }
                }
            }
        }
    }
}
```

Host handling

- Explicit marking
- Produce rom file

After initialization

Host handling

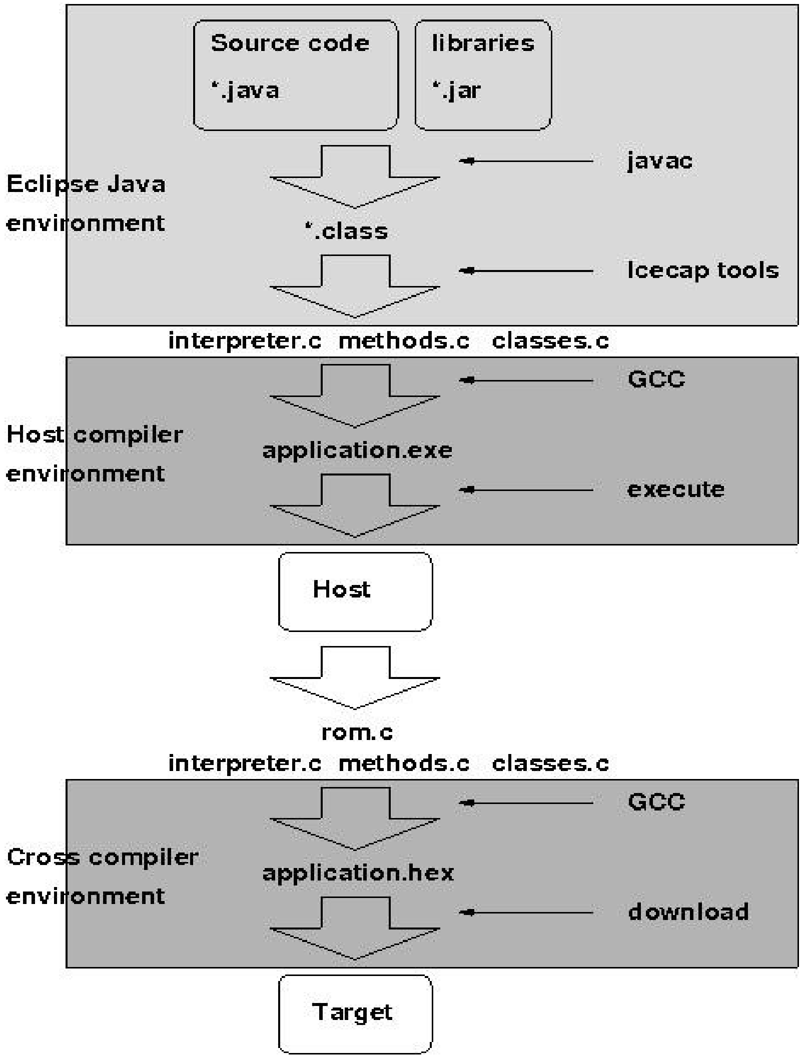
```
#include "types.h"

unsigned char pheap[29] PROGMEM = {
  0x0, 0x0, 0x0, 0x18, 0x5, 0x0, 0x0, 0x0, 0x2a,
  0x0, 0x0, 0x0, 0x26, 0x0, 0x0, 0x0, 0x2b,
  0x0, 0x0, 0xf8, 0x2, 0x0, 0x0, 0x0, 0x4,
  0x0, 0x1, 0x2, 0x3
};

unsigned char rom_writeable(void)
{
    return 0;
}
```

Autogenerated ROM file

Architecture



Accessing constant data

```
public class ConstantData extends Object{
public ConstantData();
  Code:
   0:  aload_0
   1:  invokespecial #1; //"<init>":()V
   4:  aload_0
   5:  iconstant_4
   6:  newarray byte
   8:  dup
   9:  iconstant_0
   10: bipush          23
   12: bastore
   13: dup
   14: iconstant_1
   15: bipush          112
   17: bastore
   18: dup
   19: iconstant_2
   20: iconstant_m1
   21: bastore
   22: dup
   23: iconstant_3
   24: iconstant_m1
   25: bastore
   26: putfield        #2; //wav_num_0:[B
   29: return
}
```

Target handling

Don't create in constant heap
(it's already there)
Just return reference to it

NOP

No changes

Accessing constant data

- Arrays
  - newarray
    must know if it is a constant array!
  - array load, array store
    must check if it is a constant array
    Harvard vs. Von Neumann
- Objects
  - new
    must know if it contains constant fields !
  - get field, put field
    must know if it is contains constant fields
    Harvard vs. Von Neumann again
  - instanceof, checkcast

Target handling

```
public class ConstantData extends Object{
public ConstantData();
  Code:
   0:  aload_0
   1:  invokespecial #1; //"<init>":()V
   4:  aload_0
   5:  iconstant_4
   6:  newarray byte
   8:  dup
   9:  iconstant_0
   10: bipush      23
   12: bastore
   13: dup
   14: iconstant_1
   15: bipush      112
   17: bastore
   18: dup
   19: iconstant_2
   20: iconstant_m1
   21: bastore
   22: dup
   23: iconstant_3
   24: iconstant_m1
   25: bastore
   26: putfield    #2; //wav_num_0:[B
   29: return
}
```

Cost of constant data

- If I don't use constant data what is the cost?
- If I do use constant data what is the cost for accessing that data?

Cost of constant data

- If I don't use constant data what is the cost? 7.5%

- If I do use constant data what is the cost for accessing that data? 33%

Conclusion

- Two industrial applications could not run on a JVM on the target
- We added the option for constant data in flash
- Now they may be ported
- Currently we are porting the Grundfos application to Java

Making it better

- Implicit marking of constant data
- Implicit marking of initialization phase
- Avoiding cost when not using constant data

Perspective

- Well known features from C adopted to the Java domain,

  - Hardware Objects (Schoeberl)
  - 1st level interrupt handlers
  - Constant data in flash

- Next steps,
  - Debugging devices using Eclipse

Questions/comments?