

Embedded Control for 2.017/2.019

Notes adapted from T.R. Consi 2003

Embedded Microprocessors

- *What defines microprocessors* → They are primarily made of switches: thousands or millions of small, cheap, and extremely fast switches.
- *Embedded = used for a specific task or subsystem.* A car has hundreds of embedded microprocessors, e.g., smart sensors, switches, displays, etc.
 - No user programming in an embedded microprocessor.
 - Real-time operation.
- Why embedded microprocessors instead of circuits?
 - Versatile, cheap, common, reliable, reprogrammable, etc.

Major Issues with Embedded Microprocessor (EMB) Applications

- **Fast** EMB signals vs. **slow** signals from peripherals
- **Low-power** EMB signals vs. **high-power** peripherals
- Interfacing EMB **data space** with peripheral devices' data
- **Digital** (switched: ON or OFF) vs **analog** information (continuously variable)
- **Parallel** digital (one bit per wire) vs. **serial** digital communication (bits sent sequentially over one wire).
- All relevant to the TT8!

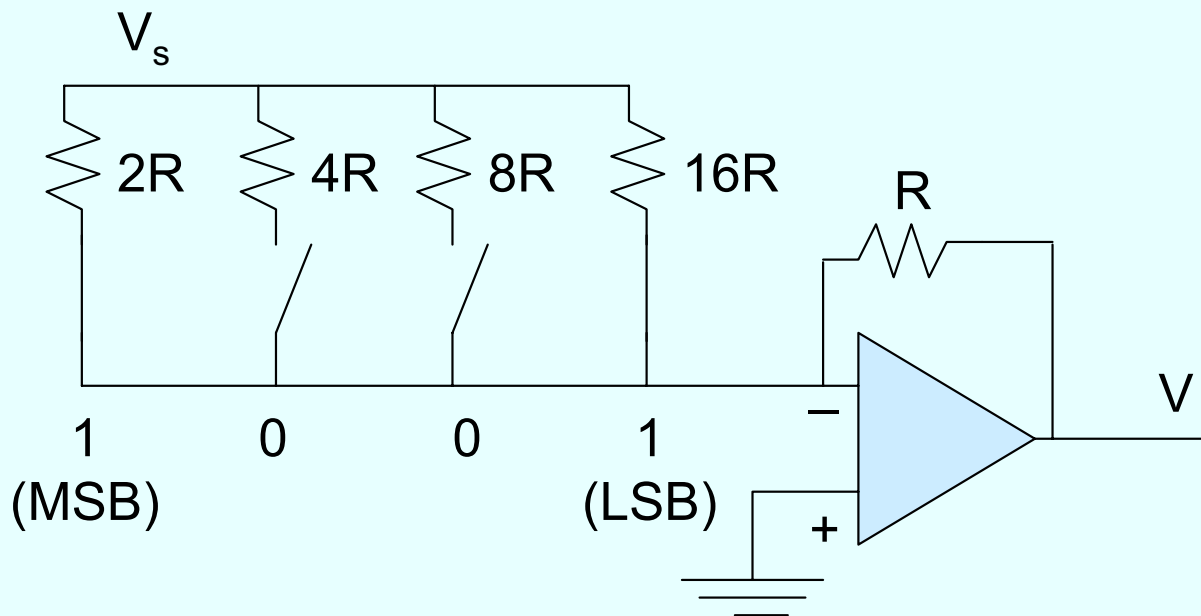
Digital to Analog Conversion (D/A)

If setting is [1,0,0,1], then

$$R_e = 1 / (1/2R + 1/16R) = 16R/9 \text{ and}$$

$$V = -R/R_e * V_s = -9/16 * V_s = -0.5625 V_s$$

If $R_e = \text{OPEN}$,	i.e., [0,0,0,0],	then $V = 0$	MIN VALUE
If $R_e = 16R$,	i.e., [0,0,0,1],	then $V = -0.0625 * V_s$	
If $R_e = 2R$,	i.e., [1,0,0,0],	then $V = -0.5000 * V_s$	
If $R_e = 16R/15$,	i.e., [1,1,1,1],	then $V = -0.9375 * V_s$	MAX VALUE



Resolution:
 $V_s/2^N$ where N is
the number of
switches, or
 $V_s/16$ in this
case.

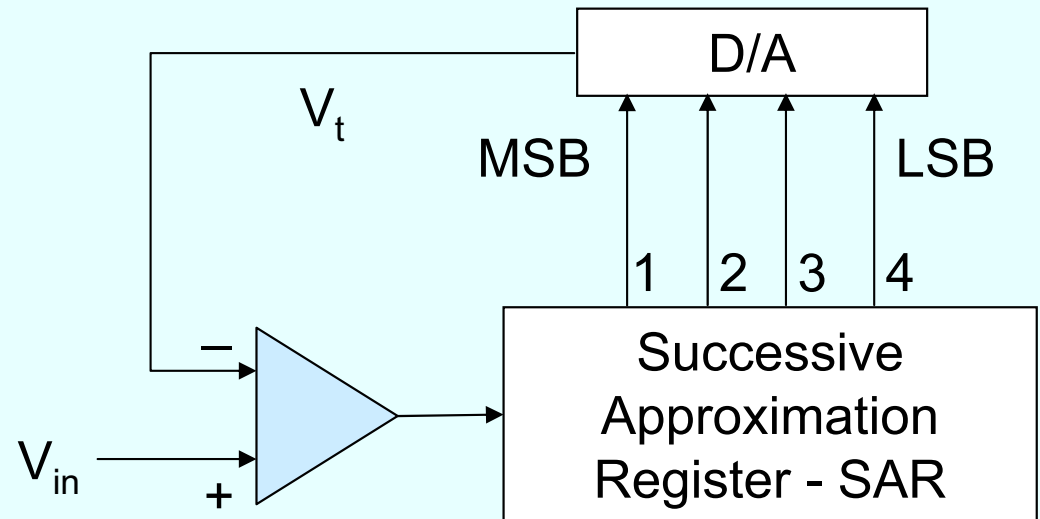
Analog to Digital Conversion (A/D)

- Uses a comparator (op-amp) and a D/A converter.

The idea:

- Set bit k
- Do the D/A conversion
- If $V_t > V_{in}$, leave bit set
Else reset bit
- Go through all the bits

The SAR and D/A are typically used multiplexed because they are so fast!



Step	SAR	Compare	Decision
1	1000	$V_t < V_{in}$	Leave bit set
2	1100	$V_t > V_{in}$	Reset bit
3	1010	$V_t > V_{in}$	Reset bit
4	1001	$V_t < V_{in}$	Leave bit set

DONE!

What is the Onset TattleTale Model 8?

- *A small, low-power, inexpensive, and self-contained system for mobile data acquisition, control, and computing.*
- Can be compared to PC-104, Octagon, etc.
- Why do we use the TT8? The board off-the-shelf can do an extremely wide array of tasks:
 - Motorola 68332 processor
 - analog A/D (8 channels, 12 bit)
 - Digital i/o lines (at least 14; these can all be configured as serial lines or PWM inputs/outputs)
 - Two dedicated serial ports for you to program with
 - Expandable memory to 64MB (and more by now) → an exceptional platform for data logging

TT8 Development Environment

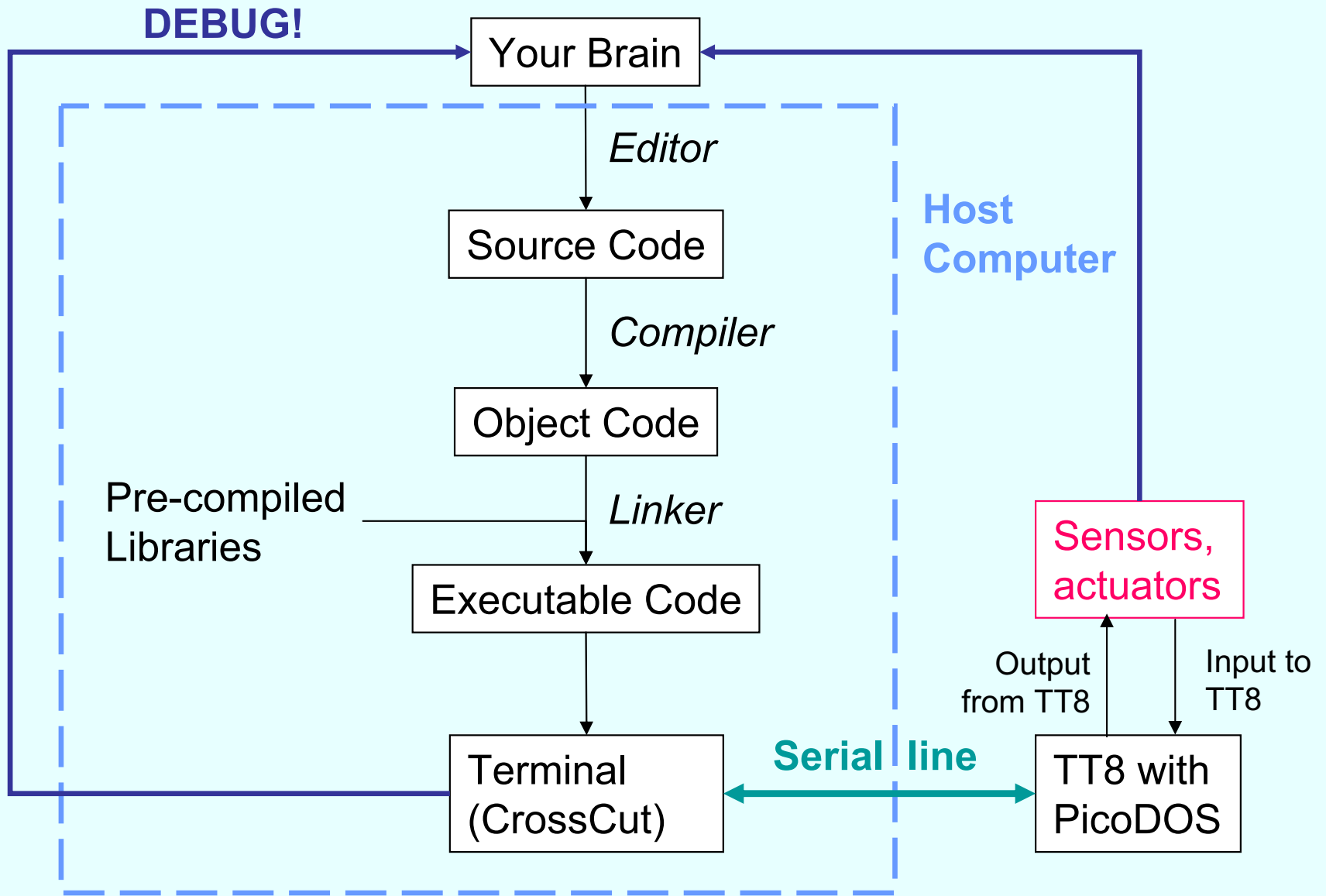
1. Source code *written* on a host computer using an editor.
 2. Code is *compiled* to object code on the host computer
 3. Object code is *linked* to functions in libraries using a linker (your own functions, pre-supplied functions, assembly language functions, etc.), making executable code.
 4. The executable code is *downloaded* to the TT8 through a serial port – it is binary.
- On the TT8, Step 1 is 99.5 percent of your development time; Step 4 is 0.5 percent of your time (annoying).

High vs. Low-Level Languages

- High-level
 - easier to write complex programs, and powerful commands.
 - easier to understand source code.
 - portable to other platforms, if recompiled for each case and specific hardware is compatible.
- Low-level:
 - more compact
 - usually faster and more streamlined
 - easier to interact directly with system hardware.
- We use C because it is common in engineering and science, fairly compact, fast, hardware-accessible → a good “middle-ground”

Operating Systems

- Govern overall operation of a computer:
 - What runs when
 - User interface
 - Memory management (e.g., RAM)
 - File system management (e.g., hard drive)
 - Peripheral devices (printer, screen, etc.)
- Multi-tasking O/S can “run more than one program at a time” - but it only looks this way on a single processor because computers are so fast!
- Real-time O/S responds quickly to input, and can execute programs in a guaranteed time.
- DOS, Window, UNIX, ... and ... **PicoDOS on the TT8!**



A TT8 “Minimum Program”

```
/* template.c
by Franz Hover, 24 January 2005, MIT
Reboot the TattleTale Model 8.
*/
#include <stdio.h>
#include <tt8lib.h>
#include <pDOS8.h>

void main()
{
    InitTT8(NO_WATCHDOG,TT8_TPU);
    INITCF8(CF8StdCS,CF8StdAddr);

    // your code here

    Reset();
}
```

Comments are delimited by `/*...*/`
This heading includes the file name, who wrote it when, and a description.

These tell the linker to pull in object code from these three library files. You might use many more than this.

This is the beginning of the program itself.

We have to do a few special lines – which we will never change – for the TT8.

`//` means the rest of the line is a comment

This reboots the TT8 – very quickly.

The end of the program.

Files for the TT8

- We use MetroWerks CodeWarrior compiler and linker for Palm O/S → an integrated project management and editing suite.
 - *Source code:* **XXX.c** ←
 - *Library or header files:* **YYY.h** ←
 - *Assembly language:* **XXX.a**
 - *Object code:* **XXX.r**
 - *Debugging file:* **XXX.dbg**
 - *Executable file (host):* **XXX.rhx** ←
 - *Executable file (TT8):* **XXX.run** ←
- NOTE: if you accidentally load the **.ahx** file instead of the **.rhx**, the TT8 runs your program automatically when it boots! ACK! (Don't worry, we can fix it ...)