Music Format

1 5 2 6 2
p tne chan note amp

if amp = 0, note is a command
switch(note)
0: turn off specified channel
1: continue for specified time with no change
else
set specified channel to specified note at specified amplitude (1-3)
Data Rate

Worst Case Data Rate: 4 simultaneous packets/event = 8 bytes/event at 200 events/sec * 8 bytes/event = 1600 bytes/sec = 12.8 Kbps can we run at 19.2?
Serial Communication: RS-232 (IEEE Standard)

- Serial protocol for point-to-point low-cost, low speed applications
- Commonly used to connect PCs to I/O devices
- RS-232 wires
  - TxD – transmit data
  - TxC – transmit clock
  - RTS – request to send
  - CTS – clear to send
  - RxD – receive data
  - RxC – receive clock
  - DSR – data set ready
  - DTR – data terminal ready
  - SG – Signal Ground
Transfer modes

- Synchronous
  Clock signal wire is used by both receiver and sender to sample data

- (Psuedo) Asynchronous
  No clock signal in common
  Data must be over sampled to “synchronize”
  Needs only three wires (one data for each direction, and ground)

- Flow control
  Handshaking signals to control byte rate, not bit rate. Signals like RTS, CTS, DSR, DTR
  Optional
Data Format

Start Bit, Stop bit, Data bits, Parity Bit (odd, even, none)

- Logic 0 (space): between +3 and +25 Volts.
- Logic 1 (mark): between -3 and -25 Volts.
- Undefined between +3 and -3 volts.
Level Converter: DS 275

Pin Description

- **RXout**: RS-232 Receiver Output (-0.3V to Vcc)
- **Vdrv**: Transmit driver +V (hook to Vcc)
- **TXin**: RS-232 Driver Output (-0.3V to Vcc)
- **GND**: System ground
- **TXout**: RS-232 Driver Output (+/- 15 V)
- **RXin**: RS-232 Receive Input (+/- 15 V)
- **Vcc**: System Logic Supply (+5V)

voltage conversion
NULL Modem Adapter

- Using only TD, RD, and SG
- No need for flow control (May miss characters if sent too fast)
- Both ends ready to send/receive at any time

This adapter does the swapping for you.
SW Flow Control Protocol

Serial Task
enqueue()
do protocol w/
Host

Tone Task
out = f(…)
Slice--;

Event Task
decqueue()
decode()

queue

What determines our true bit rate for keeping the buffer full?

Worst Case Data Rate: 4 simultaneous packets/event = 8 bytes/event at 200 events/sec * 8 bytes/event = 1600 bytes/sec = 12.8 Kbps
Review Questions?
Comparison Non OS Architectures

See Chapter 5, table 5.1 Simon
Real Time Operating Systems

- What is the basic thing we want the OS to do to help us improve worst case latency? Enable multithreading

- How? Define an OS time-slice (tick) at which highest priority ‘runnable’ task is continued. Priority function determines response behavior.

- Simplest Scheduling algorithm: each task gets at most 1 tick at a time to run. Round Robin Scheduling. Worst case task latency = #tasks*tick. Worst case run time = ticks/task * #tasks

- Some properties of such system: liveness, safety, fairness, latency, overhead.

- Other niceties: Device Drivers, Synchronization, Message passing, Memory Management
void tone_isr(void) interrupt ... {
    process_tones();
    if (!--sliceCount) {
        updateToneParameters();
        sliceCount = SliceSize
        isr_send_signal(MUSIC);
    }
}

void serial_isr(void) interrupt ...{
    timeCritical();
    os_send_signal(SERIAL);
}

void play(void) _task_ MUSIC {
    os_create(SERIAL);
    while (1) {os_wait();
        process_next_event();}
}

void serial(void) _task_ SERIAL {
    while (1) {os_wait();
        process_serial_data();} // os_create(MUSIC)?
}

Advantages:
• Deterministic response time even w/ non deterministic tasks lengths.
• Incremental development

Resources:
• Task switching overhead
• Memory overhead
• Use of system timer
• Degrades best case response time.

Tasks are threads
Another Solution

- Multiprocessor: Dedicate one processor to each (or a few) tasks.
- Still need **synchronization** and **communication**.
- A network of M-BOXes could be an example of a multiprocessor system.
Basic Architecture of an RT OS

- Task Table
  - Process state, signal flag, time_out counter, context
- System Interrupt Service Routine (timer)
- System Calls (Code, time)
Embedded Software

- Software States v. Finite State Machines
- Hierarchical State
- Thread/Process Communication
  - Critical Sections
  - Synchronization
  - Messaging and Signaling