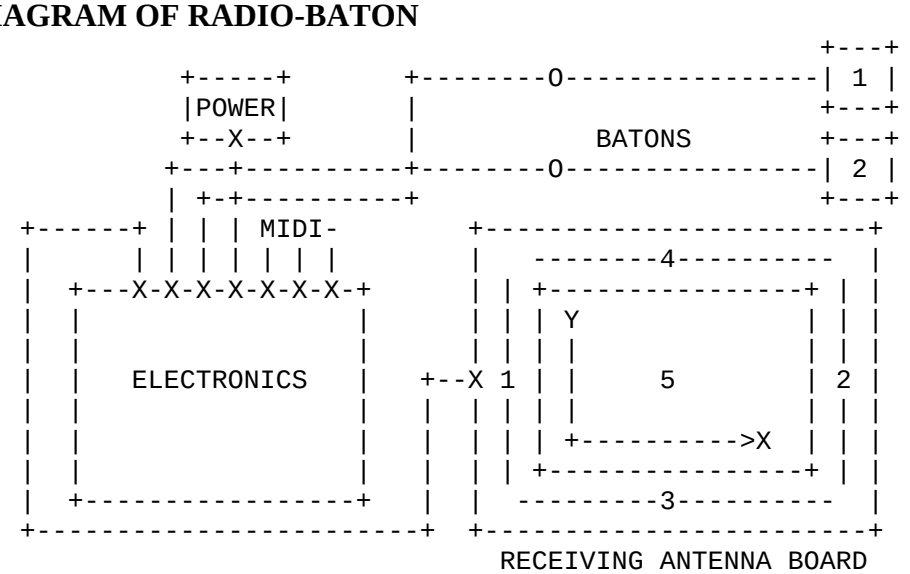
THE 1997 MATHEWS RADIO-BATON AND IMPROVISATION MODES

Richard Boulanger, rcb@media.mit.edu
Max Mathews, mvm@ccrma.stanford.edu
Music Synthesis Department
Berklee College of Music

INTRODUCTION

The Radio-Baton is a controller for live computer music performances. It tracks the motions, in three dimensional space, of the ends of two Batons which are held in the hands of a performer. The X, Y and Z trajectories of each Baton are used to control the performance. The Radio-Baton is a MIDI instrument in the sense that it has MIDI input, output, and thru connectors. All electrical communication with the Baton is done over MIDI cables using standard MIDI conventions. The Baton was designed to work with MIDI synthesizers and MIDI-based sequencing and programming software.

BLOCK DIAGRAM OF RADIO-BATON



HOW THE RADIO-BATON WORKS

The Radio-Baton uses a simple technique to determine the XYZ coordinates of the batons. At the end of each baton is a small radio transmitting antenna. On the receiving antenna surface are 5 receiving antennas as sketched on the figure--four long thin antennas arranged along the four edges of the board and one large antenna covering the entire center area of the board. The closer a baton is to a given receiver, the stronger the signal at that receiver. By comparing the signal strengths at the #1 and #2 antennas, the computer in the electronics box can determine the X position of the baton. Comparing the #3 and #4 strengths gives the Y position. The #5 strength gives the height above the board or Z position. The two batons operate at different frequencies and thus can be independently tracked.

A low frequency of about 50kHz is used for the radio signals. It is appropriate to describe the Radio-Baton as a capacitance sensor. The radio-frequency signal is really used as a method of measuring the capacitance between a transmitting antenna electrode and a receiving antenna electrode. The measurement

technique is simple and robust. The relationship between antenna signal strengths and XYZ coordinates is complex but the Radio-Baton computer is good at making the needed conversions. A table lookup procedure is used to speed the calculation.

The Radio-Baton can compute the positions of the batons every 4 milliseconds. The accuracy of measurements is about 1 part in 100. The X and Y information is linearized and mapped onto the standard MIDI range of 0-127. The Z information is not linearized. The Z data decreases from about 100 when a Baton is on the antenna surface to about 30 when the Baton is about 3 feet above the antenna surface.

TRIGGERS

In addition to providing XYZ data, one of the most important functions of the Radio-Baton is to send triggers over MIDI. A trigger can be generated when either Baton touches an invisible plane which can be positioned at various heights above the antenna board. This plane is called the HIT-LEVEL. In order to avoid DOUBLE-TRIGGERING, a second plane is also positioned slightly above the HIT plane. This plane is called the SET-LEVEL. A Baton must be raised above the SET-LEVEL before a second trigger can be generated. This method for generating triggers and avoiding double triggers using a HIT-LEVEL and a SET-LEVEL works well for most musical purposes.

PRINCIPAL PROGRAMS

The processor in the Radio-Baton can execute a number of different programs but the two main modes are the IMPROV mode and the CONDUCTOR mode. The IMPROV mode is used for executing improvisations and algorithmic programs and the CONDUCTOR mode is used to play traditional scores.

THE IMPROV MODE

In the IMPROV mode, the Radio-Baton is a simple controller which sends XYZ position information and triggers to a control computer. The control computer interprets this information musically according to whatever program is being run in the control computer.

THE CONDUCTOR MODE

When the Conductor Program is running in the Radio-Baton, a control computer loads a score into the processor in the Radio-Baton. The Baton plays this score according to the conductor-gestures made by the performer. In response, the Radio-Baton sends MIDI commands from its MIDI out connector to a synthesizer which plays the notes.

TYPICAL COMMANDS

When the Radio-Baton is powered up or reset, it's processor enters a polling loop looking for commands received at it's MIDI input terminal. Most commands are encoded as MIDI system-exclusive messages which contain only an operation code as a single information character. For most commands the Radio-Baton sends a confirmation message from its MIDI output to the control computer.

TEST

Received Message	240 1 247
Response	240 1 247

This command is used to test whether the Baton is working and whether the MIDI is correctly patched.

UART

Received Message	240 33 247
Response	240 33 247

This command puts the Baton into the UART mode in which all MIDI characters received are simply transmitted from the MIDI out jack. The Radio-Baton can only be gotten out of the UART mode by pushing the reset push button or button B15+ on the Baton box. When B15+ is pushed the Baton transmits:

240 44 247

to confirm that it has left the UART mode.

IMPROV

Received Message	240 46 247
Response	240 46 247

This command puts the Baton into the IMPROV mode where it functions as a simple controller.

PLAY

Received Message	240 4 247
Response	none

This command puts the Baton into the CONDUCTOR mode where it will play a specified score under the expressive control of baton gestures. This mode has already been described in other publications and will not be further discussed here.

THE IMPROV PROTOCOL

When in the IMPROV mode, the Baton functions as a simple controller and sends triggers and other controller information from its MIDI out jack. Usually a computer receives this information. The task of making musical sense from the controller information falls entirely on the computer, its program, and its programmer. The conversation between the Baton processor and the computer is two-way. For example, in order to reduce the load on the midi cable, the Baton only sends the XYZ position data for the Batons when this information is requested by the computer.

Although a number of examples are provided with the system, the user is expected to program their own improvization algorithms. To assist in this task, a support program and template are provided. Starting with the template, the user merely needs to fill in the blank functions. In addition to those functions provided in the template, the user can access various functions for sending MIDI commands to a synthesizer and can read a millisecond clock which is provided automaticaly by the support program.

*** THE IMPROV PROGRAMMING TEMPLATE IN THE C LANGUAGE:

```
#include <stdio.h>
#include <conio.h>
#include "drivers.h"
#include "define.h"
#include "impr.h"
/*--------------------initialization algorithms-------------------------*/
initialization(){}
/*--------------------main loop algorithms -----------------------------*/
mainloopalgorithms(){}
/*--------------------triggered algorithms-----------------------------*/
stick1trig(){}
stick2trig(){}
b14plustrig(){}
b15plustrig(){}
b14minusup(){};
b14minusdown(){};
b15minusup(){};
```

```
b15minusdown(){};
noteplay(){}
/*--------------------------poll response algorithms----------------------*/
stick1pollresponce(){}
stick2pollresponce(){}
potspollresponce(){}
/*----additional functions and variables provided by the support program---*/
m_send(a)-----------------------send midi character
m_poll()------------------------receive midi character
m_play(chan,keyno,keyvel)-------play note
m_cont(chan,controller,value)---control change
m_pc(chan,program)--------------program change
t_time--------------------------clock in milliseconds
```

*** A SIMPLE IMPROVISATION PROGRAM: PLAY NOTES WITH BATON1 & SUSTAIN WITH PEDAL

```
#include <stdio.h>
#include <conio.h>
#include "drivers.h"
#include "define.h"
#include "impr.h"
  short keyno;
/*---------------------initialization algorithms-------------------------*/
initialization(){
  printf("Baton1 Trigger plays note on MIDI channel#1: X Axis = Note\n");
  m_pc2(0,25); printf("Initial Patch is #25 on MIDI Channel #1\n");
}
/*--------------------triggered algorithms-------------------------------*/
stick1trig(){
  m_play2(0,keyno,0);
  keyno=44+x1t/3;
  m_play2(0,keyno,127);
  printf("keyno= %d\n",keyno);
}
b14minusup(){m_cont2(0,64,0);printf("pedal b14- up\n");}
b14minusdown(){m_cont2(0,64,127);printf("pedal b14- down\n");}
```