AVR® 8-bit Microcontroller

**AVR42778: Core Independent Brushless DC Fan Control Using Configurable Custom Logic on ATtiny817**

APPLICATION NOTE

**Features**

- Base setup for performing core independent brushless DC motor (BLDC) commutation and dead time insertion using Configurable Custom Logic (CCL)
- 100% CPU available after initialization for additional tasks like control, monitoring, and communication
- Motor commutation based on Hall sensor signal using Analog Comparator (AC) and Configurable Custom Logic (CCL)
- Dead time insertion based on Hall sensor signal using Analog Comparator (AC), Event System (EV SYS), and 16-bit Timer/Counter Type B (TCB)
- Pulse Width Modulated (PWM) signal generation using 16-bit Timer/Counter Type A (TCA)
- Example setup for Atmel® ATtiny817 Xplained Mini is provided

**Introduction**

As package dimensions go down and power consumption figures go up thermal management becomes an increasingly important factor in modern day electronics design. Perhaps the simplest form of thermal management is forced convection, i.e. increasing the dissipation of heat by moving the air inside and around the heat source. This is most conveniently done using fans, which are powered by BLDC motors. BLDC motors are commutated electronically, eliminating problems such as mechanical wear of brushes, but also reducing EMI (Electro-Magnetic Interference). The most straightforward fan designs simply spin the fan rotor at full speed, but as the number of fans tends to increase so does the noise and power consumption. In many cases it would be desirable to keep noise and power consumption at a minimum. This, in turn could create a demand for adjusting the speed of the fan according to environmental conditions and/or other external factors.

This application note describes a core-independent method of using an AVR® device to control a simple BLDC fan. Many peripherals are configured to work together, enabling the Configurable Custom Logic (CCL) module to
achieve BLDC motor commutation and dead time insertion while feeding a PWM signal to a dual driver BLDC motor circuit, independent of CPU operation. After initialization, no CPU cycles are needed to operate the motor at the set PWM duty cycle, achieving minimal process delay compared to an interrupt based configuration.

The device configuration described in this application note could serve as starting point for an intelligent BLDC motor control application with features like automatic speed control and external monitoring. A typical application could be monitoring and control of a cooling fan.
# Table of Contents

Features............................................................................................................................................. 1

Introduction....................................................................................................................................... 1

1. Theory of Operation .....................................................................................................................4
   1.1. Common BLDC Fan Construction.........................................................................................4
   1.2. Hall Effect Sensors in BLDC Motors..................................................................................4
   1.3. BLDC PWM Operation.........................................................................................................5
   1.4. Dead Time Insertion..............................................................................................................5

2. CCL BLDC Fan Controller Implementation .................................................................................6

3. Device Specific Implementation Details.....................................................................................10

4. CCL BLDC Fan Hardware Suggestions.......................................................................................11

5. CCL BLDC Fan Further Development........................................................................................13

6. Get the Device Datasheet............................................................................................................14

7. Get the ATtiny817 Xplained Mini Evaluation Kit.......................................................................15

8. Get Atmel Studio 7.0..................................................................................................................17

9. Get Source Code from Atmel START........................................................................................18

10. Terms and Abbreviations............................................................................................................19

11. Revision History..........................................................................................................................20
1. **Theory of Operation**

The design presented in this application note could be applied to a range of systems using a simple BLDC motor with Hall sensor feedback. Since a very common application would be that of a cooling fan, the application note focuses mainly on that particular usage.

1.1. **Common BLDC Fan Construction**

Basic brushless DC motors can be broken into two main components: the rotor and the stator. As implied by its name, the rotor is the rotating part while the stator is the static construct around which the rotor rotates. The rotor houses permanent magnets, and in the case of a fan motor, the fan blades are attached to it. See the figure below.

![Common BLDC Fan Rotor](image1)

The motor coils are housed in the stator, which when energized correctly will make the motor turn. In common two-phase BLDC motors there are typically four coils. See the figure below.

![Common BLDC Fan Stator](image2)

Torque to turn the motor is produced by the interaction between the magnetic fields created by the energized motor coils and those of the permanent magnets in the rotor. What motor coils should be energized at what time is given by the angular position of the rotor relative to the stator. The act of applying current to the correct coils as the motor turns is called commutation. Information about the orientation of the rotor is most commonly fed back to the motor controller by a Hall effect sensor mounted on the stator.

1.2. **Hall Effect Sensors in BLDC Motors**

A digital Hall effect sensor (also called buffered Hall effect sensor) detects the strength of the currently present magnetic field, and has a single output wire with two possible output states; high and low. BLDC motors are typically equipped with Hall effect sensors to provide information about the orientation of the rotor relative to the stator. When the rotor and its permanent magnets move the magnetic flux detected by the Hall sensor changes. This can be used for commutation by a motor controller, as it can tell the controller when to activate which coils in the motor in order for it to turn.

For a simple BLDC motor with one Hall sensor the frequency of the sensor signal will be twice the rotation frequency of the motor, toggling level four times per motor rotation. This is illustrated in the figure below.
1.3. **BLDC PWM Operation**

A basic way of turning a fan motor would be to constantly apply the supply voltage directly to the motor coils in accordance with the Hall sensor signal. This would make the fan rotate at its maximum achievable speed for the given supply voltage.

In many applications it would be desirable to control the fan speed. As the steady-state fan speed is closely coupled to the average applied voltage, open loop speed control can be achieved by toggling the applied voltage between supply level and ground at high frequency using transistors and PWM drive signals. The average applied voltage will then be proportional to the duty cycle of the PWM signal, which can be set and adjusted by a motor controller to manipulate the fan speed.

1.4. **Dead Time Insertion**

In order to avoid short circuiting the supply voltage side to ground, also called shoot-through, for a brief moment when toggling a PWM signal from driving one motor driver to another, the drivers should be held in inactive state for a number of PWM cycles. This is called dead time insertion.

Since a BLDC motor controller typically uses a Hall sensor signal to determine which of the stator coils to power, this signal should also be used to trigger the insertion of a dead time period.
2. **CCL BLDC Fan Controller Implementation**

Some basic elements and features that are often needed for operating a BLDC motor with a microcontroller are PWM control signals, a Hall sensor input signal, motor commutation and PWM dead time insertion. For a simple BLDC fan motor with two drive signal inputs and a Hall sensor output, the resulting waveforms when the motor is rotating should look similar as illustrated in the figure below. The names of the waveforms in the figure are the names of the specific module signals selected for design and implementation of the application described in this document.

By making the device produce the outlined waveforms based on the provided Hall sensor signal it will act as a simple BLDC Fan Controller. As the name of the application note suggests, the Configurable Custom Logic (CCL) plays the main role in accomplishing this.

**Figure 2-1. Sketch of Relevant Waveforms for the CCL BLDC Fan Controller**

```
<table>
<thead>
<tr>
<th>Waveform</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hall sensor (ACOUT)</td>
</tr>
<tr>
<td>Base PWM (TCA0 WO)</td>
</tr>
<tr>
<td>Dead time (TCB0 WO)</td>
</tr>
<tr>
<td>PWM A (CCL LUT0-OUT)</td>
</tr>
<tr>
<td>PWM B (CCL LUT1-OUT)</td>
</tr>
</tbody>
</table>
```

**The Configurable Custom Logic (CCL) module**

The Configurable Custom Logic (CCL) module is a programmable logic peripheral, which can be connected to the device pins, events, or peripherals. This allows the user to eliminate external logic gates for simple glue logic functions on the PCB, reducing bill-of-materials (BOM) cost. Each LUT consists of three inputs, a truth table, and a filter/edge detector. Each LUT can generate an output as a user programmable logic expression with three inputs.

**Implementation Overview**

From examining the waveforms in the figure above it can be seen that the logical value of the signals PWM A and PWM B are each given by a combination of the logical values of the three other signals. This represents a typical use-case for the CCL module, which can generate the two desired signals by using the Hall sensor signal, base PWM signal, and dead time signal as inputs.

The Hall sensor signal waveform can be generated by connecting the physical Hall sensor to the Analog Comparator (AC) module, while the 16-bit Timer/Counter Type A (TCA) is suited for generating the base PWM signal, also called "pulse train". By combining two additional device modules it is also possible to create the dead time signal. The signal from the Analog Comparator (AC) can be routed through the Event System (EVSYS) to 16-bit Timer/Counter Type B (TCB). In 16-bit Timer/Counter Type B (TCB) it can be used to trigger a square wave which will represent the dead time signal in the Configurable Custom Logic (CCL) module.

An overview of the involved modules is presented below. By configuring these correctly, a controller for a simple BLDC fan that performs motor commutation and dead time insertion without using the CPU can be realized. A more specific description of how the different modules are set up is presented in the following sections.
**Configurable Custom Logic (CCL) Setup**

Two programmable Look-up Tables, also referred to as LUTs or Truth tables, are needed to output two separate PWM signals. Enabling of the LUTs and sending the outputs to pads is done by writing to the CTRLA register of each LUT. Since the value of both output signals is given by a combination of the Hall sensor signal, the dead time signal, and the base PWM signal, both LUTs are set up to take these three inputs by writing to their respective CTRLB and CTRLC registers.

The behavior of each LUT is determined by its truth table, which is set up by writing to its respective TRUTH register. The hexadecimal value to be written to the TRUTH registers for this application is determined from TRUTHn[0:7] to be 0x02 for LUT0 and 0x08 for LUT1 as shown in the table below. The two figures following the table illustrate the two LUT setups using logic gates. Which signal is routed to what LUT input can be seen from the table and figures below.

**Table 2-1. Truth Table for LUT0 and LUT1 with Identical Input Selection**

<table>
<thead>
<tr>
<th>IN[2], TCB W0</th>
<th>IN[1], ACOUT</th>
<th>IN[0], TCA W0</th>
<th>CCL LUT0 OUT</th>
<th>CCL LUT1 OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0 (TRUTH0[0])</td>
<td>0 (TRUTH1[0])</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1 (TRUTH0[1])</td>
<td>0 (TRUTH1[1])</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0 (TRUTH0[2])</td>
<td>0 (TRUTH1[2])</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0 (TRUTH0[3])</td>
<td>1 (TRUTH1[3])</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0 (TRUTH0[4])</td>
<td>0 (TRUTH1[4])</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0 (TRUTH0[5])</td>
<td>0 (TRUTH1[5])</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0 (TRUTH0[6])</td>
<td>0 (TRUTH1[6])</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0 (TRUTH0[7])</td>
<td>0 (TRUTH1[7])</td>
</tr>
</tbody>
</table>
The values of the two TRUTH registers implement the desired behavior for the LUTs according to the illustrated waveform in the introduction of this chapter. The output of LUT0 will follow the base PWM signal as long as the dead time signal is low and the Hall sensor signal is low, while the LUT1 output will follow the base PWM signal as long as the dead time signal is low and the Hall sensor signal is high.

**Analog Comparator (AC), Voltage Reference (VREF) and Digital to Analog Converter (DAC) Setup**

To provide the status of the connected Hall sensor, which indicates the position of the fan rotor, the Analog Comparator (AC) is set up to take the physical Hall sensor signal on its positive input pin and the Digital to Analog Converter (DAC) output as its negative input. This is set up via the MUXCTRLA register of the Analog Comparator (AC) module.

The output of the Digital to Analog Converter (DAC) is set to a fixed level by setting an initial input value and selecting the voltage reference. This is done by writing to the DATA register of the module, and the CTRLA register of the Voltage Reference (VREF) module, respectively.

The Analog Comparator (AC) output signal, ACOUT, will be high when the physical sensor signal is above the Digital to Analog Converter (DAC) output and low when its below. This aligns with the illustrated waveform in the introduction of this chapter.

**Note:** The Voltage Reference (VREF) and Digital to Analog Converter (DAC) modules need to be set up such that the Digital to Analog Converter (DAC) output lies between the upper and lower steady state voltage output levels of the physical Hall sensor.

**16-bit Timer/Counter Type B (TCB) and Event System (EVSYS) Setup**

Inserting PWM dead time using the CCL as outlined in this application note requires a signal indicating when the PWM outputs should be held inactive during the specified dead time period. Since the dead
time should be inserted when the PWM output is to be switched from one driver signal to the other, which is indicated by the Hall sensor signal toggling its level, the insertion should be triggered by the edges of the ACOUT signal as these will coincide with the edges of the Hall sensor output.

The 16-bit Timer/Counter Type B (TCB) module has a "Single-Shot" mode that starts a single counting session upon detecting an edge on its Event input signal. It will then count until it reaches its TOP value and stop. The timer output signal TCB W0 will be high during counting and low otherwise, making it very suitable as a dead time signal by using the ACOUT signal as Event input.

The ACOUT signal is routed to the 16-bit Timer/Counter Type B (TCB) via the Event System (EVSYS) by selecting ACOUT as input to an event channel and assigning the 16-bit Timer/Counter Type B (TCB) to be a user of the same event channel. In this application, this is set up by writing to the ASYNCCCH0 and ASYNCCUSER0 registers of the Event System (EVSYS), respectively. Furthermore, the 16-bit Timer/Counter Type B (TCB) is configured to asynchronous single-shot mode with dual Event edge detection by writing to the CTRLB and EVCTRL registers of the timer. The dead time duration is set up through the CCMP register.

16-bit Timer/Counter Type A (TCA) Setup
The base PWM signal represents the PWM signal that will be output to either PWM A or PWM B depending on the Hall sensor signal input. 16-bit Timer/Counter Type A (TCA) is used for this purpose in this application, implicating that the period and duty cycle of the PWM outputs must be set up in this module.

The timer is configured to use basic Single-Slope PWM Generation mode by writing to its CTRLB register, while the PWM period and duty cycle is set up by writing the desired values to the PER and CMP0 registers, respectively. The TCA W0 is used directly as an input to the Configurable Custom Logic (CCL) module. Open loop speed control of the connected BLDC motor can then be accomplished by adjusting the duty cycle between 0 and 100% of the timer period.

I/O Pin Controller (PORT) and Port Multiplexer (PORTMUX) Setup
Inputs and outputs are directed to and from the external pins of the device by setting up the I/O Pin Controller (PORT) module. Several signals in the device can be connected to an alternate pin instead of its default pin. This is configured in the Port Multiplexer (PORTMUX) module. The pins that need to be set up for this application is the positive input pin for the Analog Comparator (AC), which should be connected to the Hall sensor, and the two output pins from the Configurable Custom Logic (CCL), which should be connected to the two control inputs.
3. **Device Specific Implementation Details**

Some details are specific for the device or device group for which the accompanying code was written, thus they may or may not be applicable for other devices. The details described below applies to, but is not necessarily limited to, ATtiny817/ATtiny816/ATtiny814/ATtiny417.

**Configurable Custom Logic (CCL)**

On ATtiny817/ATtiny816/ATtiny814/ATtiny417 the Configurable Custom Logic (CCL) module contains two Look-up Tables (LUTs), taking up to six inputs and generating up to two different outputs.

**I/O Pin Controller (PORT) and Port Multiplexer (PORTMUX)**

For the ATtiny817/ATtiny816/ATtiny814/ATtiny417 the pin used for Hall sensor input in this application is shared with the default output pin of LUT1. The alternate pin for LUT1-OUT is therefore used by writing to the CTRLA register of the Port Multiplexer (PORTMUX) module. The specific pins used are listed in the table below.

<table>
<thead>
<tr>
<th>Pin</th>
<th>Description</th>
<th>Signal</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA7</td>
<td>Hall sensor input</td>
<td>P0</td>
<td>Analog Comparator (AC)</td>
</tr>
<tr>
<td>PA4</td>
<td>PWM output A</td>
<td>LUT0-OUT</td>
<td>Configurable Custom Logic (CCL)</td>
</tr>
<tr>
<td>PC1</td>
<td>PWM output B</td>
<td>LUT1-OUT</td>
<td>Configurable Custom Logic (CCL)</td>
</tr>
</tbody>
</table>

16-bit Timer/Counter Type A (TCA)

The accompanying code for this application note uses the default clock settings on ATtiny817/ATtiny816/ATtiny814/ATtiny417, which from the 20MHz CLK_MAIN and a division factor of 6, give a CLK_PER of 3.33MHz. The default PWM period is set to 100 timer counts and the default PWM duty cycle is set to 20 timer counts. As 16-bit Timer/Counter Type A (TCA) is set up with a clock division factor of 1, this corresponds to 30µs and 6µs, respectively.

16-bit Timer/Counter Type B (TCB)

The accompanying code for this application note uses the default clock settings on ATtiny817/ATtiny816/ATtiny814/ATtiny417, which from the 20MHz CLK_MAIN and a division factor of 6, give a CLK_PER of 3.33MHz. The default dead time is set up to 500 timer counts. As 16-bit Timer/Counter Type B (TCB) is set up with a clock division factor of 1, this corresponds to 150µs.
4. **CCL BLDC Fan Hardware Suggestions**

The accompanying code includes support for adjusting the PWM duty cycle with the on-board button of the ATtiny817 Xplained Mini. Other than this there is no specific hardware provided or described for this application note. For this reason, and the fact that BLDC motors and their control circuits come in many different varieties, two specific fan hardware layouts that will fit the described device setup are outlined below.

**Two Phase BLDC Fan with Dual Control Input**

As the setup described in this application note is designed to take one Hall sensor input and output two PWM control signals, a simple approach would be to use a two phase BLDC fan motor with a single Hall sensor output and one driver element connected to each phase as illustrated in the figure below. The drivers and Hall sensor should be connected to the device as indicated.

*Figure 4-1. Outline of a Two Phase BLDC Fan Motor with Drivers*

For this configuration no shoot-through will occur if both drivers are activated, thus the dead time can in fact be reduced to zero or removed.

**One Phase BLDC Fan with full H-bridge Control**

In addition to the more basic configuration illustrated above, the described application is also suited for controlling a single phase BLDC fan motor with a full H-bridge driver circuit by connecting the device as outlined in the figure below. For this configuration shoot-through will occur if both PWM signals are active, thus dead time is needed when switching between what output the PWM pulses are sent to.
Figure 4-2. Outline of a One Phase BLDC Fan Motor and a Full H-bridge

- Hall sensor input
- AVR
- PWM A
- One Phase BLDC
- PWM B
- Hall

Atmel AVR42778: Core Independent Brushless DC Fan Control Using Configurable Custom Logic on ATtiny817 [APPLICATION NOTE]
5. **CCL BLDC Fan Further Development**

This application note presents a basic setup for core independent BLDC fan operation that can act as a starting point for adding more advanced features and/or control. Some examples could be:

- **Temperature based fan speed control:** This could be implemented by connecting a temperature sensor via the ADC and set the PWM duty cycle based on the measured value. The duty cycle could for instance be set from a predefined look-up table or by a control algorithm.

- **Closed loop speed control:** This could be implemented by estimating the fan speed from the Hall sensor signal switching rate and using the value as input to a control algorithm.

- **Stall detection:** This could be implemented by detecting lack of Hall sensor signal switching.

- **Analog reference input:** This could be implemented by connecting a potentiometer to an ADC input and use the output value as the set point for a speed or temperature controller, or to simply set the PWM duty cycle directly.

- **Serial communication:** By using the SPI module, remote monitoring and control can be added. This could be very useful if the fan is to be integrated in a larger system with some higher level control scheme.
6. **Get the Device Datasheet**


Document/file: Atmel ATtiny817/ATtiny816/ATtiny814/ATtiny417 Datasheet (summary, complete)(.pdf)

- There are two versions:
  - Complete version (includes all peripheral descriptions and electrical characteristics)
  - Summary version
7. Get the ATtiny817 Xplained Mini Evaluation Kit

Figure 7-1. ATtiny817 Xplained Mini Kit


Get the kit: http://www.atmel.com/tools/attiny817-xmini.aspx#buy

Document/file:
- ATtiny817 Xplained Mini User Guide (.pdf)

Key features:
- ATtiny817 microcontroller
- One yellow user LED
- One mechanical button
- Two QTouch® buttons
- mEDBG
  - Auto-ID for board identification in Atmel Studio
  - One green board status LED
  - Programming and Debugging
  - Virtual COM port (CDC)
• USB powered
• ATtiny817 power sources:
  – 5.0V from USB
  – 3.3V regulator
  – External voltage
• Arduino shield compatible footprints

The ATtiny817 Xplained Mini User Guide covers how to power the kit, the detailed information on board components, extension interface and the hardware guide.
8. **Get Atmel Studio 7.0**


   **Document/file:**
   - Atmel Studio 7.0 (build 1006) Installer - Full (.exe)

   Atmel Studio 7.0 or later is the preferred IDE for developing and debugging firmware for ATtiny817/ATtiny816/ATtiny814/ATtiny417.
9. **Get Source Code from Atmel START**

The example code is available through Atmel START, which is a web-based tool that enables configuration of application code through a graphical user interface. The code can be downloaded for both Atmel Studio 7.0 and IAR™ IDE via the **Examples**-link below, or the **BROWSE EXAMPLES** button on the Atmel START front page.

**Web page:** [http://start.atmel.com/](http://start.atmel.com/)

**Documentation:** [http://start.atmel.com/static/help/index.html](http://start.atmel.com/static/help/index.html)

**Examples:** [http://start.atmel.com/#examples](http://start.atmel.com/#examples)

In the Examples-browser, search for: AVR42778 BLDC Fan Control (press **User Guide** in Atmel START for detailed requirements for the example project).

Double-click the downloaded .atzip file and the project will be imported to Atmel Studio 7.0.

For information on how to import the project in IAR, press the **Documentation**-link above, select ‘Atmel Start Output in External Tools’ and 'IAR Embedded Workbench'.

"Atmel AVR42778: Core Independent Brushless DC Fan Control Using Configurable Custom Logic on ATtiny817 [APPLICATION NOTE]"
## Terms and Abbreviations

**Table 10-1. Dictionary**

<table>
<thead>
<tr>
<th>Phrase/Abbreviation</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLDC</td>
<td>Brushless Direct Current</td>
</tr>
<tr>
<td>PWM</td>
<td>Pulse Width Modulation</td>
</tr>
<tr>
<td>Dead time</td>
<td>Buffer time between enabling transistors connected to different voltages to avoid short circuiting the two voltage levels</td>
</tr>
<tr>
<td>EMI</td>
<td>Electromagnetic Interference</td>
</tr>
<tr>
<td>Duty cycle</td>
<td>Used to describe the duration of the on-period of a pulse width modulated signal</td>
</tr>
<tr>
<td>Hall effect sensor</td>
<td>Magnetic sensor commonly used for indicating the angular position of the rotor in a brushless DC motor in order to correctly apply current to the motor coils</td>
</tr>
<tr>
<td>Commutation</td>
<td>The act of applying current to the correct coils of an electrical motor in order to make it turn</td>
</tr>
<tr>
<td>Steady-state</td>
<td>The condition of a dynamic system when its states (like speed, current, etc.) have stabilized</td>
</tr>
<tr>
<td>Open loop control</td>
<td>Term used about a control scheme where information about the system state or states that is to be controlled is not fed back to the controller to create a closed control loop that adjusts the controller output automatically based on the feedback</td>
</tr>
<tr>
<td>PCB</td>
<td>Printed Circuit Board</td>
</tr>
<tr>
<td>I/O</td>
<td>Input/Output. Used as collective term for input and output signals to/from the device.</td>
</tr>
<tr>
<td>Shoot-through</td>
<td>Term commonly used about short circuiting a transistor between supply voltage and ground, potentially damaging the transistor</td>
</tr>
</tbody>
</table>
# 11. Revision History

<table>
<thead>
<tr>
<th>Doc. Rev.</th>
<th>Date</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>42778B</td>
<td>11/2016</td>
<td>Minor correction in chapter &quot;Device Specific Implementation Details&quot;</td>
</tr>
<tr>
<td>42778A</td>
<td>10/2016</td>
<td>Initial document release</td>
</tr>
</tbody>
</table>