WPF: Time Control 12 Hour Format

Time Control 12 Hour Format

I've spent a bit of time scouring the internet for a WPF Time Control that supports 12-hour format. There are several examples out there that are in 24-hour format, but I haven't seen any for 12-hour format, so I wrote one. I began with some code located here and then modified it extensively to work for my situation. I didn't need to display seconds, however it would be pretty easy to add them.
This control supports a wide range of keyboard input. You can enter the value for Hour, Minute, andPart of Day by either using the up/down arrow keys or by typing in numbers and letters. If you have any questions/suggestions/praise/opinions please feel free to leave a comment!

XAML:

```
<UserControl x:Class="CGS.TimeControl"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:local="clr-namespace:CGS"
             Height="Auto" Width="Auto" x:Name="UserControl">
    <UserControl.Resources>
        <local:MinuteSecondToStringConverter x:Key="minuteSecondConverter" />
    </UserControl.Resources>

    <Border BorderBrush="Black" BorderThickness="1" CornerRadius="1">
        <Grid x:Name="LayoutRoot" Width="Auto" Height="Auto" Margin="2"
Background="White">
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="0.2*"/>
                <ColumnDefinition Width="0.05*"/>
                <ColumnDefinition Width="0.2*"/>
                <ColumnDefinition Width="0.05*"/>
                <ColumnDefinition Width="0.2*"/>
            </Grid.ColumnDefinitions>

            <Grid x:Name="hour" Focusable="True" KeyDown="Down"
GotFocus="Grid_GotFocus" LostFocus="Grid_LostFocus" MouseDown="Grid_MouseDown">
                <TextBlock TextWrapping="Wrap" Text="{Binding Path=Hours,
ElementName=UserControl, Mode=Default}"
                 TextAlignment="Center" VerticalAlignment="Center" FontSize="{Binding
ElementName=UserControl, Path=FontSize}"/>
            </Grid>

            <Grid  Grid.Column="1">
                <TextBlock  x:Name="sep1" TextWrapping="Wrap"
VerticalAlignment="Center" Background="{x:Null}"
                 FontSize="{Binding ElementName=UserControl, Path=FontSize}" Text=":"
TextAlignment="Center"/>
            </Grid>
```

```
            <Grid  Grid.Column="2" x:Name="min" Focusable="True" KeyDown="Down"
GotFocus="Grid_GotFocus" LostFocus="Grid_LostFocus" MouseDown="Grid_MouseDown">
                <TextBlock  TextWrapping="Wrap" Text="{Binding Path=Minutes,
ElementName=UserControl, Mode=Default, Converter={StaticResource
minuteSecondConverter}}"
                 TextAlignment="Center" VerticalAlignment="Center" FontSize="{Binding
ElementName=UserControl, Path=FontSize}"/>
            </Grid>

            <Grid  Grid.Column="4" Name="half" Focusable="True" KeyDown="Down"
GotFocus="Grid_GotFocus" LostFocus="Grid_LostFocus" MouseDown="Grid_MouseDown">
                <TextBlock TextWrapping="Wrap" Text="{Binding Path=DayHalf,
ElementName=UserControl, Mode=Default}"
                 TextAlignment="Center" VerticalAlignment="Center" FontSize="{Binding
ElementName=UserControl, Path=FontSize}"/>
            </Grid>

        </Grid>
    </Border>
</UserControl>
```

Code:

```
using System;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Data;

namespace CGS
{
    /// <summary>
    /// Interaction logic for TimeControl.xaml
    /// </summary>
    public partial class TimeControl : UserControl
    {
        const string amText = "am";
        const string pmText = "pm";

        static SolidColorBrush brBlue = new SolidColorBrush(Colors.LightBlue);
        static SolidColorBrush brWhite = new SolidColorBrush(Colors.White);

        DateTime _lastKeyDown;

        public TimeControl()
        {
            InitializeComponent();

            _lastKeyDown = DateTime.Now;
        }

        public TimeSpan Value
        {
            get { return (TimeSpan)GetValue(ValueProperty); }
            set { SetValue(ValueProperty, value); }
        }
        public static readonly DependencyProperty ValueProperty =
        DependencyProperty.Register("Value", typeof(TimeSpan), typeof(TimeControl),
        new UIPropertyMetadata(DateTime.Now.TimeOfDay, new
PropertyChangedCallback(OnValueChanged)));
```

```
        private static void OnValueChanged(DependencyObject obj,
DependencyPropertyChangedEventArgs e)
        {
            TimeControl control = obj as TimeControl;

            TimeSpan newTime = ((TimeSpan)e.NewValue);

            int timehours = newTime.Hours;
            int hours = timehours % 12;
            hours = (hours > 0) ? hours : 12;

            control._hours = newTime.Hours;
            control.Hours = hours;
            control.Minutes = ((TimeSpan)e.NewValue).Minutes;
            control.DayHalf = ((timehours - 12) >= 0) ? pmText : amText;

        }

        private int _hours;
        public int Hours
        {
            get { return (int)GetValue(HoursProperty); }
            set { SetValue(HoursProperty, value); }
        }
        public static readonly DependencyProperty HoursProperty =
        DependencyProperty.Register("Hours", typeof(int), typeof(TimeControl),
        new UIPropertyMetadata(0));

        public int Minutes
        {
            get { return (int)GetValue(MinutesProperty); }
            set { SetValue(MinutesProperty, value); }
        }
        public static readonly DependencyProperty MinutesProperty =
        DependencyProperty.Register("Minutes", typeof(int), typeof(TimeControl),
        new UIPropertyMetadata(0));


        public string DayHalf
        {
            get { return (string)GetValue(DayHalfProperty); }
            set { SetValue(DayHalfProperty, value); }
        }
        public static readonly DependencyProperty DayHalfProperty =
        DependencyProperty.Register("DayHalf", typeof(string), typeof(TimeControl),
        new UIPropertyMetadata(amText));

        private void Down(object sender, KeyEventArgs args)
        {
            bool updateValue = false;

            if (args.Key == Key.Up || args.Key == Key.Down)
            {
                switch (((Grid)sender).Name)
                {
                    case "min":
                        if (args.Key == Key.Up)
                            if (this.Minutes + 1 > 59)
                            {
                                this.Minutes = 0;
                                goto case "hour";
                            }
                            else
```

```
                            {
                                this.Minutes++;
                            }
                        if (args.Key == Key.Down)
                            if (this.Minutes - 1 < 0)
                            {
                                this.Minutes = 59;
                                goto case "hour";
                            }
                            else
                            {
                                this.Minutes--;
                            }
                        break;

                    case "hour":
                        if (args.Key == Key.Up)
                            this._hours = (_hours + 1 > 23) ? 0 : _hours + 1;
                        if (args.Key == Key.Down)
                            this._hours = (_hours - 1 < 0) ? 23 : _hours - 1;
                        break;

                    case "half":
                        this.DayHalf = (this.DayHalf == amText) ? pmText : amText;

                        int timeHours = this.Hours;
                        timeHours = (timeHours == 12) ? 0 : timeHours;
                        timeHours += (this.DayHalf == amText) ? 0 : 12;

                        _hours = timeHours;
                        break;
                }

                updateValue = true;

                args.Handled = true;
            }
            else if ((args.Key >= Key.D0 && args.Key <= Key.D9) || (args.Key >=
Key.NumPad0 && args.Key <= Key.NumPad9))
            {
                int keyValue = (int)args.Key;
                int number = 0;

                number = keyValue - ((args.Key >= Key.D0 && args.Key <= Key.D9) ?
                                        (int)Key.D0 :
                                        (int)Key.NumPad0
                                    );

                bool attemptAdd = (DateTime.Now - _lastKeyDown).TotalSeconds < 1.5;

                switch (((Grid)sender).Name)
                {
                    case "min":
                        if (attemptAdd)
                        {
                            number += this.Minutes * 10;

                            if (number < 0 || number >= 60)
                            {
                                number -= this.Minutes * 10;
                            }
                        }
```

```
                    this.Minutes = number;
                    break;

                case "hour":
                    if (attemptAdd)
                    {
                        number += this.Hours * 10;

                        if (number < 0 || number >= 13)
                        {
                            number -= this.Hours * 10;
                        }
                    }

                    number = (number == 12) ? 0 : number;
                    number += (this.DayHalf == amText) ? 0 : 12;

                    _hours = number;
                    break;

                default:
                    break;
            }

            updateValue = true;

            args.Handled = true;
        }
        else if (args.Key == Key.A || args.Key == Key.P)
        {
            if (((Grid)sender).Name == "half")
            {
                this.DayHalf = (args.Key == Key.A) ? amText : pmText;

                updateValue = true;
            }
        }

        if (updateValue)
        {
            this.Value = new TimeSpan(_hours, this.Minutes, 0);
        }

        _lastKeyDown = DateTime.Now;
    }

    private void Grid_GotFocus(object sender, RoutedEventArgs e)
    {
        var grd = sender as Grid;

        grd.Background = brBlue;
    }

    private void Grid_LostFocus(object sender, RoutedEventArgs e)
    {
        var grd = sender as Grid;

        grd.Background = brWhite;
    }

    private void Grid_MouseDown(object sender, MouseButtonEventArgs e)
    {
        var grd = sender as Grid;
```

```
            grd.Focus();
        }
    }


    public class MinuteSecondToStringConverter : IValueConverter
    {
        #region IValueConverter Members

        public object Convert(object value, Type targetType, object parameter,
System.Globalization.CultureInfo culture)
        {
            if (value != null)
            {
                if (value is int)
                {
                    return ((int)value).ToString("00");
                }
            }

            return string.Empty;
        }

        public object ConvertBack(object value, Type targetType, object parameter,
System.Globalization.CultureInfo culture)
        {
            if (value != null)
            {
                if (value is string)
                {
                    int number;
                    if (int.TryParse(value as string, out number))
                    {
                        return number;
                    }
                }
            }

            return 0;
        }

        #endregion
    }
}
```

And here's an example of how to use it:

```
<cgs:TimeControl x:Name="timeControl" Margin="15,0,0,0" Height="25" Width="70"
                                 xmlns:cgs="clr-namespace:CGS"/>

// Set the time of the control
this.timeControl.Value = DateTime.Now.TimeOfDay;

// Get the time from the control
TimeSpan time = this.timeControl.Value;



Website Link: http://mel-green.com/2010/03/wpf-time-control-12-hour-format/
```