Custom Status Line Setup Guide

This guide explains how to set up a custom color-coded status line for Claude Code that displays:

- Working Directory (Cyan)

- Model Name (Yellow)

- Context Window Usage with visual meter + percentage (Green/Orange/Red)

- Current Git Repository and Branch (Magenta)

Windows Setup

Windows uses PowerShell (built-in, no additional dependencies required).

Step 1: Create the PowerShell Script

Create a file named statusline.ps1 in your .claude folder:

C:\Users\<YourUsername>\.claude\statusline.ps1

PowerShell Script (statusline.ps1):

```
# Claude Code Status Line - Windows PowerShell
# Color-coded KPIs with visual context meter

$esc = [char]27

# Color definitions (ANSI)
$cyan    = "$esc[96m"      # Working directory
$yellow  = "$esc[93m"      # Model
$magenta = "$esc[95m"      # Git repo/branch
$green   = "$esc[92m"      # Context low
$orange  = "$esc[38;5;208m" # Context medium
$red     = "$esc[91m"      # Context high
$dim     = "$esc[90m"      # Separators/labels
$reset   = "$esc[0m"

# Read JSON input
$input_json = [Console]::In.ReadToEnd() | ConvertFrom-Json

# Working Directory
$full_dir = $input_json.workspace.current_dir
$dir = Split-Path -Leaf $full_dir

# Model
$model = $input_json.model.display_name

# Context Window with visual meter
$context_display = ''
```

```
$usage = $input_json.context_window.current_usage
if ($usage) {
    $current = $usage.input_tokens + $usage.cache_creation_input_tokens +
$usage.cache_read_input_tokens
    $size = $input_json.context_window.context_window_size
    if ($size -gt 0) {
        $pct = [int](($current * 100) / $size)

        # Choose color based on usage level
        if ($pct -lt 50) {
            $ctx_color = $green
        } elseif ($pct -lt 80) {
            $ctx_color = $orange
        } else {
            $ctx_color = $red
        }

        # Build visual meter [========  ] style
        $meter_width = 10
        $filled = [int](($pct * $meter_width) / 100)
        if ($filled -gt $meter_width) { $filled = $meter_width }
        $empty = $meter_width - $filled
        $meter_bar = ('=' * $filled) + (' ' * $empty)

        $context_display = "${dim}CONTEXT WINDOW ${ctx_color}[${meter_bar}] ${pct}%${reset}"
    }
}

# Git Repo/Branch
$git_display = ''
try {
    $branch = git -C $full_dir --no-optional-locks branch --show-current 2>$null
    if ($branch) {
        $repo_name = git -C $full_dir --no-optional-locks remote get-url origin 2>$null
        if ($repo_name) {
            $repo_name = [System.IO.Path]::GetFileNameWithoutExtension($repo_name.Split('/')[-1])
            $git_display = "${magenta}${repo_name}:${branch}${reset}"
        } else {
            $git_display = "${magenta}${branch}${reset}"
        }
    }
} catch {}

# Build output with separators
$sep = "${dim} | ${reset}"
$parts = @()
$parts += "${cyan}${dir}${reset}"
$parts += "${yellow}${model}${reset}"
if ($context_display) { $parts += $context_display }
if ($git_display) { $parts += $git_display }

$output = $parts -join $sep
Write-Host $output -NoNewline
```

Step 2: Configure settings.json

Edit or create the file:

C:\Users\<YourUsername>\.claude\settings.json

Add the following configuration (replace <YourUsername> with your actual username):

```
{
  "statusLine": {
    "type": "command",
    "command": "powershell -NoProfile -ExecutionPolicy Bypass -File 
C:\\Users\\<YourUsername>\\.claude\\statusline.ps1"
  }
}
```

Mac / Linux Setup

Step 1: Install jq

jq is a command-line JSON processor required for parsing the status data.

Mac (using Homebrew):

```
brew install jq
```

Ubuntu/Debian:

```
sudo apt install jq
```

Fedora/RHEL:

```
sudo dnf install jq
```

Step 2: Create the Bash Script

Create a file named statusline.sh in your .claude folder:

`~/.claude/statusline.sh`

Bash Script (statusline.sh):

```
#!/bin/bash
# Claude Code Status Line - Mac/Linux
# Color-coded KPIs with visual context meter

# Color definitions (ANSI)
CYAN="\033[96m"
YELLOW="\033[93m"
MAGENTA="\033[95m"
GREEN="\033[92m"
ORANGE="\033[38;5;208m"
RED="\033[91m"
DIM="\033[90m"
RESET="\033[0m"

# Read JSON from stdin
INPUT=$(cat)

# Parse JSON with jq
DIR=$(basename "$(echo "$INPUT" | jq -r '.workspace.current_dir')")
MODEL=$(echo "$INPUT" | jq -r '.model.display_name')
CURRENT_TOKENS=$(echo "$INPUT" | jq -r '
  .context_window.current_usage |
  if . then (.input_tokens + .cache_creation_input_tokens + .cache_read_input_tokens) else 0 end
')
CONTEXT_SIZE=$(echo "$INPUT" | jq -r '.context_window.context_window_size // 0')
FULL_DIR=$(echo "$INPUT" | jq -r '.workspace.current_dir')

# Build context meter
CTX_DISPLAY=""
if [ "$CONTEXT_SIZE" -gt 0 ] 2>/dev/null; then
    PCT=$((CURRENT_TOKENS * 100 / CONTEXT_SIZE))
    # Choose color based on usage
    if [ "$PCT" -lt 50 ]; then
        CTX_COLOR="$GREEN"
    elif [ "$PCT" -lt 80 ]; then
        CTX_COLOR="$ORANGE"
    else
        CTX_COLOR="$RED"
    fi

    # Build visual meter
    FILLED=$((PCT / 10))
    [ "$FILLED" -gt 10 ] && FILLED=10
    EMPTY=$((10 - FILLED))
    METER=$(printf '=%.0s' $(seq 1 $FILLED 2>/dev/null) | tr -d '\n')
    SPACES=$(printf ' %.0s' $(seq 1 $EMPTY 2>/dev/null) | tr -d '\n')

    CTX_DISPLAY="${DIM}CONTEXT WINDOW ${CTX_COLOR}[${METER}${SPACES}] ${PCT}%${RESET}"
fi

# Git repo/branch
GIT_DISPLAY=""
if [ -d "$FULL_DIR" ]; then
    BRANCH=$(git -C "$FULL_DIR" --no-optional-locks branch --show-current 2>/dev/null)
    if [ -n "$BRANCH" ]; then
        REPO=$(basename "$(git -C "$FULL_DIR" --no-optional-locks remote get-url origin
2>/dev/null)" 2>/dev/null | sed 's/.git$//')
        if [ -n "$REPO" ]; then
            GIT_DISPLAY="${MAGENTA}${REPO}:${BRANCH}${RESET}"
        else
            GIT_DISPLAY="${MAGENTA}${BRANCH}${RESET}"
        fi
    fi
fi

# Build output
SEP="${DIM} | ${RESET}"
OUTPUT="${CYAN}${DIR}${RESET}${SEP}${YELLOW}${MODEL}${RESET}"
[ -n "$CTX_DISPLAY" ] && OUTPUT="${OUTPUT}${SEP}${CTX_DISPLAY}"
[ -n "$GIT_DISPLAY" ] && OUTPUT="${OUTPUT}${SEP}${GIT_DISPLAY}"

printf "%b" "$OUTPUT"
```

Step 3: Make the Script Executable

```
chmod +x ~/.claude/statusline.sh
```

Step 4: Configure settings.json

Edit or create the file:

~/.claude/settings.json

```
{
  "statusLine": {
    "type": "command",
    "command": "bash ~/.claude/statusline.sh"
  }
}
```

Color Reference

KPI	Color	Description
Working Directory	Cyan	Current folder name
Model	Yellow	Claude model being used
Context Meter	Green/Orange/Red	Green <50%, Orange 50-80%, Red >80%
Git Repo:Branch	Magenta	Repository name and current branch

Example Output

```
REPO | Claude Opus 4.5 | CTX [======    ] 62% | my-project:main
```

Troubleshooting

1. If colors do not display, ensure your terminal supports ANSI colors

2. On Windows, use Windows Terminal (not legacy cmd.exe) for best results

3. On Mac/Linux, verify jq is installed by running: jq --version

4. Restart Claude Code after making changes to settings.json

5. Ensure the script path in settings.json matches your actual file location