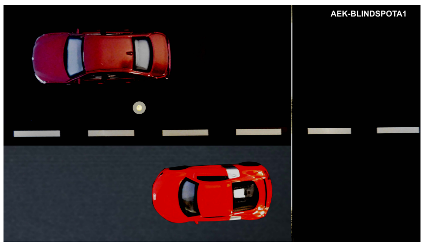
AEKD-BLINDSPOTA1

Data brief

Blind-spot detection simulation kit



Product summary	
Blind-spot detection simulation kit	AEKD-BLINDSPOTA1
Firmware for AEKD-BLINDSPOTB1	STSW-BLINDSPOT
Development environment	SPC5-Studio
community.st.com/autodevkit	AutoDevKit
AutoDevKit library plugin for SPC5-STUDIO	STSW-AUTODEVKIT
SPC5-UDESTK Debugging Software for Windows	SPC5-UDESTK-SW
MCU discovery board for SPC5 Chorus 4M automotive microcontroller with CAN transceivers	AEK-MCU-C4MLIT1
Digitally controlled LED driver board for automotive lighting applications	AEK-LED-21DISM1
VN7050AS evaluation board based on VIPower M0-7 technology	EV-VN7050AS
Applications	Chassis and Safety

Features

- Conveyor belt with DC gearbox motor
- Hall sensor for magnetic field detection
- 1 car model 1:43 fixed
- 1 car model 1:43 with internal magnets to be placed on the conveyor belt
- High brightness LED turns on when two cars are near each other
- Part of the AutoDevKit™ initiative
- RoHS and WEEE compliant

Description

The AEKD-BLINDSPOTA1 blind-spot kit simulates the detection of motor a vehicle in the regions around a car that drivers typically cannot see directly or with mirrors. It works by interpreting magnetic field variations through a Hall sensor mounted on one car model that remains stationary and magnets mounted on another car model that moves on the conveyor belt driven by a 12 V_{DC} gearbox motor with 40 RPM and high torque.

The AEKD-BLINDSPOTA1 hardware assembly system is used with the AEKD-BLINDSPOTB1 kit of boards to drive the blind-spot demo loads.

The goal of the training kit is to help developers build effective firmware for blind-spot systems using the SPC5-Studio integrated development environment with the AutoDevKit plugin extension.

1 Overview

Blind-spot detection is implemented in modern transportation vehicles to prevent accidents in common overtaking scenarios and other dangerous circumstances involving vehicles occupying areas that cannot seen directly by the drivers of other vehicles. These warning systems detect the presence of vehicles in blind-spot zones and alert the driver.

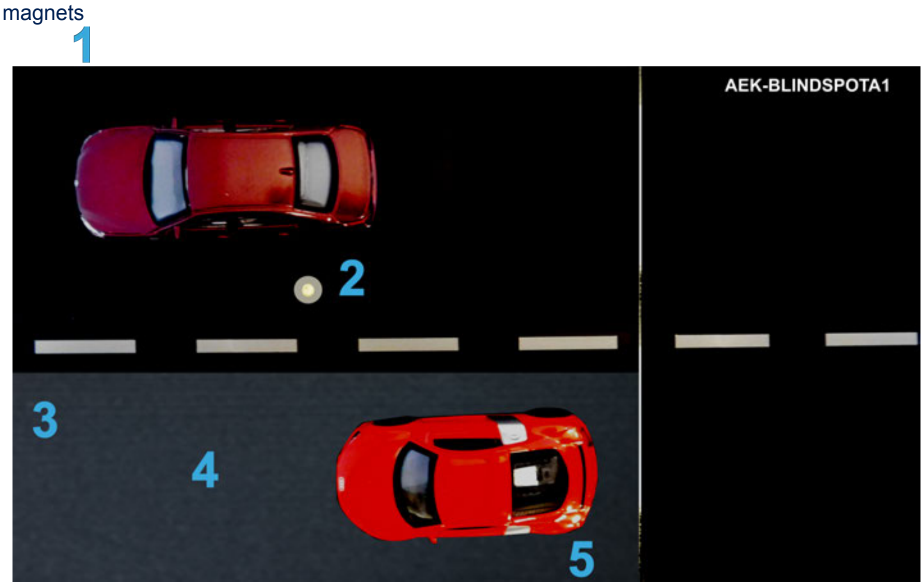
In this simulation kit, the system consists of a plexiglass structure to simulate the loads and a set of boards to drive the application (AEKD-BLINDSPOTB1). Two cars are placed on the demo: one is stationary, while another car with embedded magnets moves on the conveyor belt driven by a gear-DC motor. A Hall sensor under the belt measures the magnetic field around it and a LED near the stationary car indicates when the blind-spot warning signal is triggered.

The application is controlled by:

- a 32-bit automotive microcontroller (SPC58EC80E5)
- a VIPower actuator board (EV-VN7050AS) with a high side driver used to drive the conveyor belt geared DC motor
- a LED driver board (AEK-LED-21DISM1) drives the warning LED

Figure 1. **AEKD-BLINDSPOTA1 hardware assembly**

1. Side for board connection
2. LED
3. DC motor
4. Hall sensor
5. Car with magnets



1.1 Block diagram

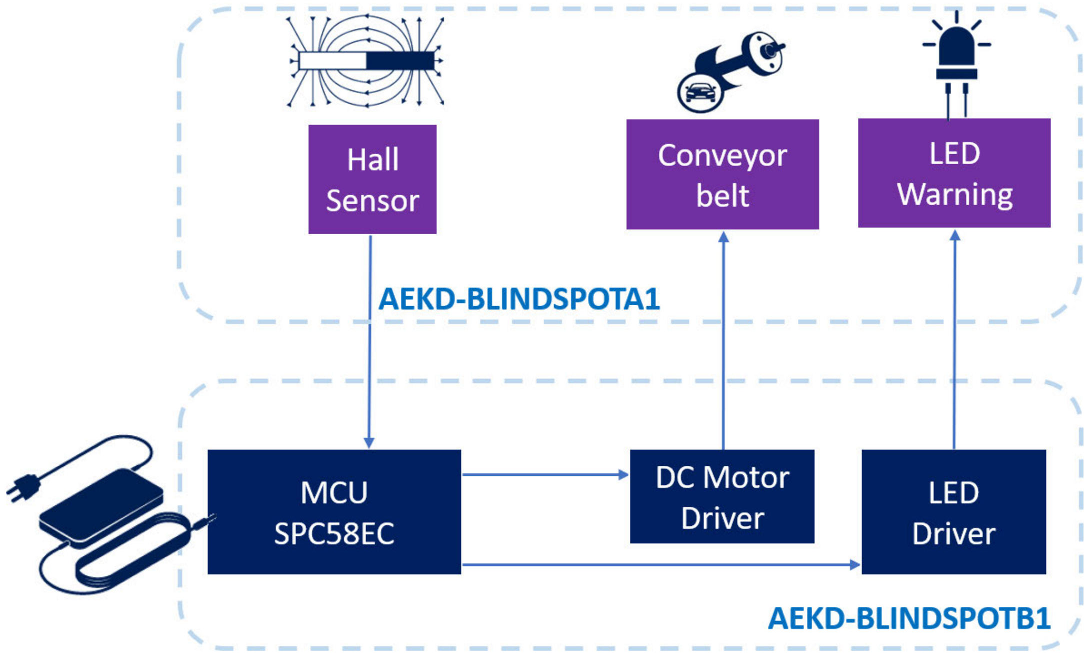


Figure 2. AEKD-BLINDSPOTA1 plus AEKD-BLINDSPOTB1 blind-spot detection kit component block diagram

1.2 How to create a blind-spot detection application using AutoDevKit

Step 1. Create a new project and select the MCU platform to use, i.e. SPC58 Chorus 4 Mb

Step 2. Add SPC5 Init Component for clock configurator, interrupt management and an operating system abstraction layer together with bootstrap files

Step 3. Add AutoDevKit Init Component for Board View Editor

Step 4. Add low-level drivers for MCU peripherals management

Step 5. Pick AutoDevKit components according to the blind-spot warning system to build: LED driver, DC motor driver for the conveyor belt and the Hall sensor capture component

Step 6. Configure the added AutoDevKit components

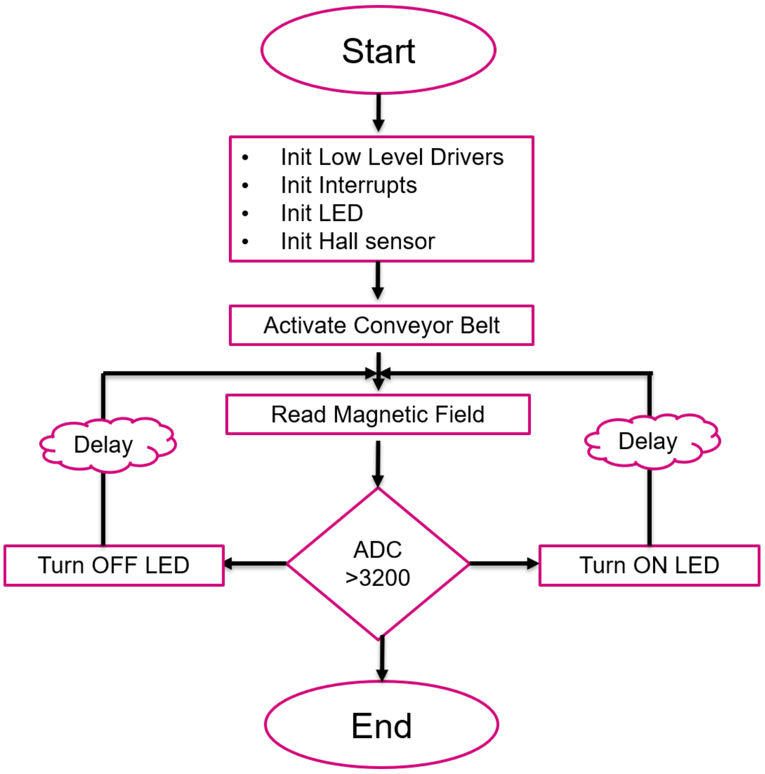
Step 7. Run automatic pin allocation

Step 8. Verify the assigned pins in the PinMap Editor

Step 9. Connect your boards using the Board View

Step 10. Write the `main()` function using AutoDevKit with high-level APIs

Figure 3. `main()` function operation block diagram



Step 11. Compile the code

Step 12. Download the code using PLS UDE

Step 13. Debug your application

Revision history

Table 1. **Document revision history**

Date	Version	Changes
13-Feb-2020	1	Initial release.
04-Mar-2020	2	Minor text changes.