SPC570S50 CAN example

Quick Guide

Version 1.0, February, 2018

Table of Contents

1. Terms & Abbreviations

CAN
: Controller Area Network

t_q
: Time Quantum

PIT
: Periodic Interrupt Timer

IOP
: I/O processor

BSP
: Board Startup Package

GPIO
: General Purpose Input Output

crt0
: 'C' run-time environment initialization code

HAL
: Hardware Abstraction Layer

EVB
: Evaluation Board

PIT
: Periodic Interrupt Timer

NC
: Not Configured

uC
: Microcontroller

ISR
: Interrupt Service Routine

2. Introduction

This example is a small functional project designed to simplify an evaluation phase of the certain peripheral on the microcontroller. It is built on the c-startup example providing necessary low-level functions like a startup code, a minimalistic hardware abstraction or predefined memory partitioning. On top of this low-level implementation, it provides the peripheral example code running on single core.

2.1. Prerequisites

- SPC570S50 device
- An evaluation board (SPC57xxMB + SPC570S50)
- HighTec Development Suite, version: 4.9.3.0

2.2. HW resources

Hardware resources used by this example:

Tab. 1. HW resources

HW unit	channel / output pin	function
FlexCAN0_TX	PC[3]	Core [0] FlexCAN transmit output
FlexCAN0_RX	PC[2]	Core [0] FlexCAN receive input
FlexCAN0	MB[5]	Core [0] Interrupt - Frame available in the FIFO
default resources from the c-startup example		
GPIO	PA [0]	Core [0] LED control
PIT 0	Channel [0]	Core [0] periodic interrupt

By default, the evaluation board does not have CAN interface enabled. To enable the CAN interface, dedicated jumpers on EVB board shall be connected as described in the table below.

Tab. 2. SPC57xxMB CAN configuration

Jumper	PCB Legend	Description
J21	CAN1_EN	1-2: WAKE to GND, 3-4: STB to 5V, 5-6: EN to 5V
J33	CAN_PWR	1-2: 5V_SR to PHY U2 VCC, 3-4: 12V to PHY U2 VBAT

Jumper	PCB Legend	Description
J35	CAN	1-2: 5V_SR to PHY U1 VCC, 3-4: 12V to PHY U1 VBAT
J37	CAN1_TX	2: PC[3] to CAN TX (wire connection)
J38	CAN1_RX	2: PC[2] to CAN RX (wire connection)

3. Example overview

The example shows both parts of the communication, sending and receiving a CAN frame.

CAN Frame transmit

The PIT timer IRS schedules a periodic transmit of CAN frames on the bus.

CAN Frame receive

The example accepts any CAN frame sent by the external CAN node.

3.1. System Description

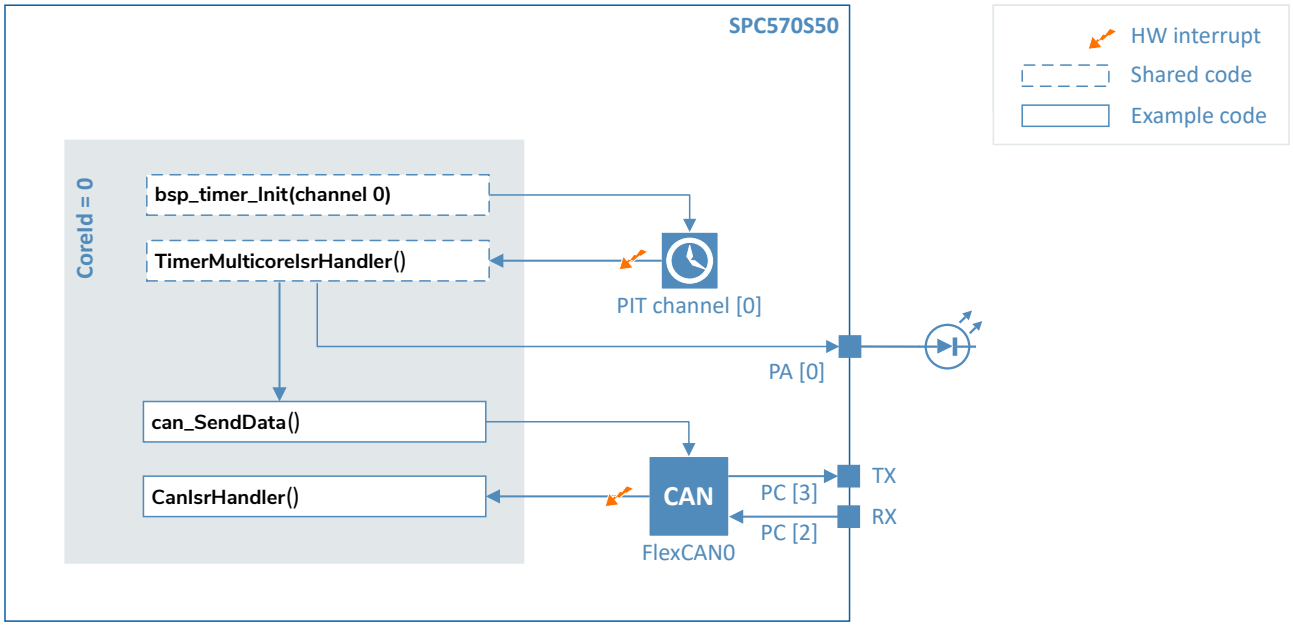

Fig. 1. System block diagram

- The example code runs on a single core.
- The example initializes FlexCAN module with a standard CAN frame format

Code 1. Initial CAN frame for transmission (../app/can.c)

```
CAN_MESSAGE_OBJECT_t g_can_TxData = {.ID = 0x10, .LENGTH = 8, \
.Data[0] = 0, .Data[1] = 1, .Data[2] = 2, .Data[3] = 3,       \
.Data[4] = 4, .Data[5] = 5, .Data[6] = 6, .Data[7] = 7};
```

3.2. CAN frame transmit

- The PIT timer periodically triggers CAN frame transmission.
- The send routine increments data[0] values after each successful transmit.

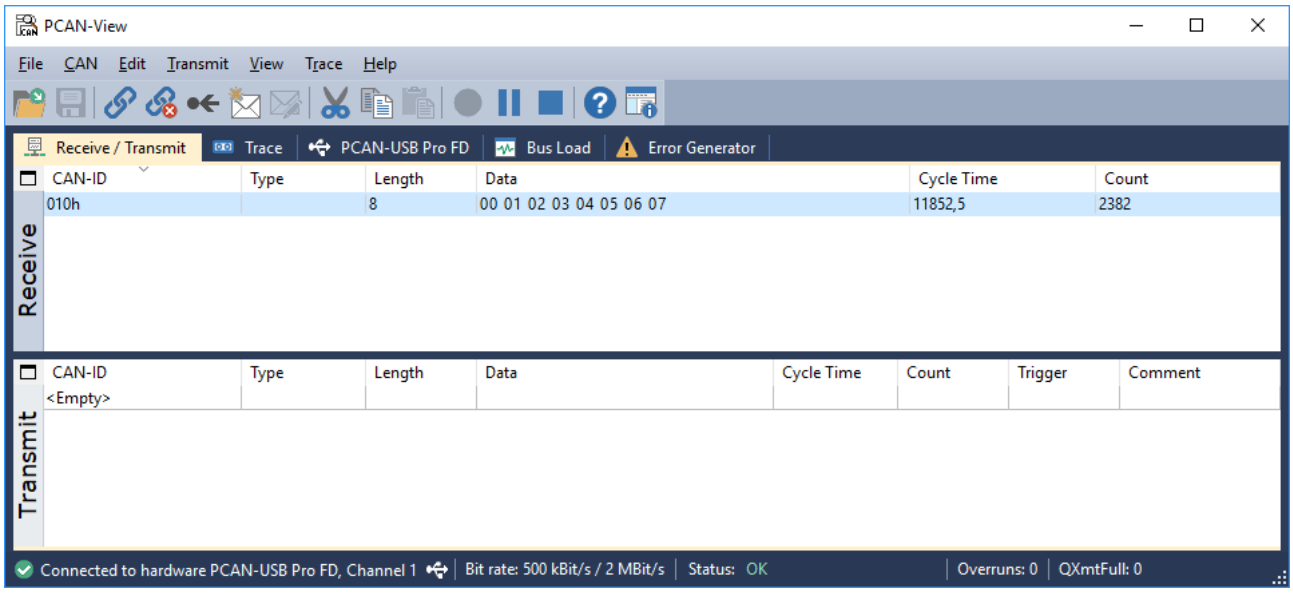

Fig. 2. Received CAN frame on the Host

3.3. CAN frame receive

- The application receives data to FIFO buffer of the CAN module.
- An event `new data in FIFO` triggers an interrupt.
- An interrupt service routine moves data from FIFO to a global variable `g_can_RxData`.

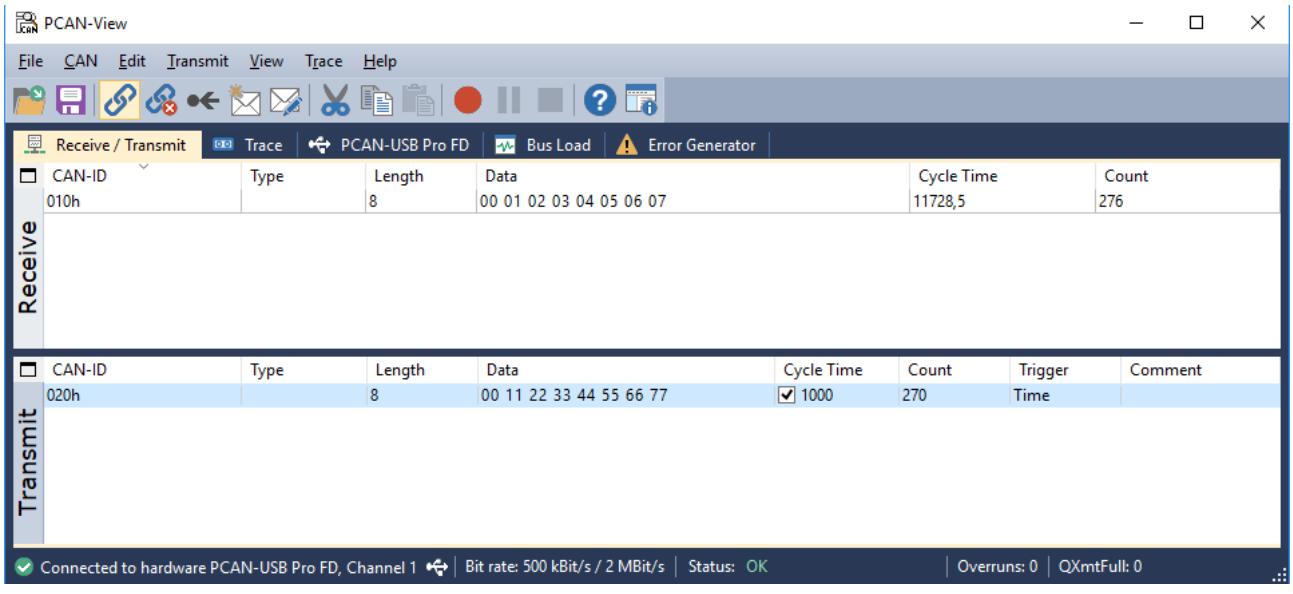

Fig. 3. CAN frame transmitted from the host

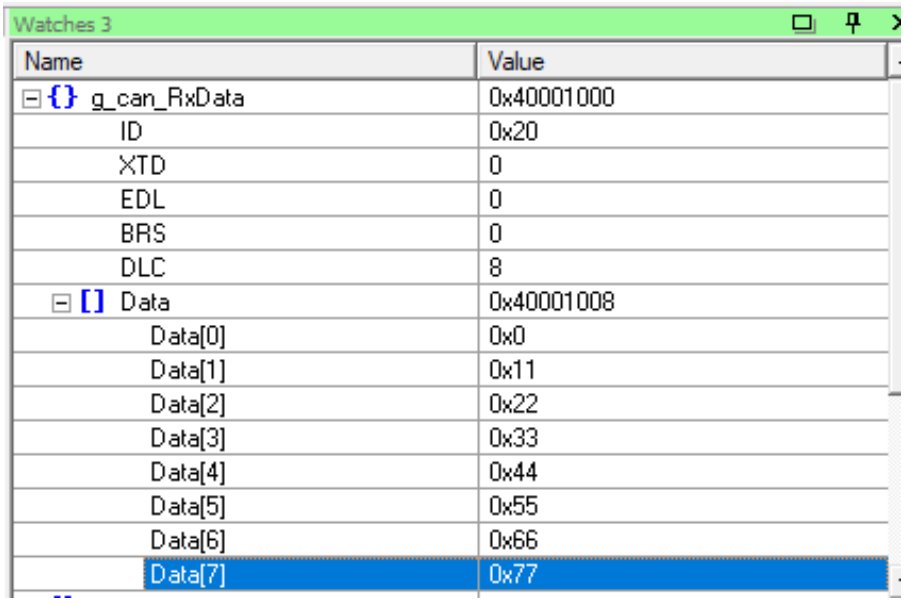

Fig. 4. Received CAN frame

3.4. Application Flow

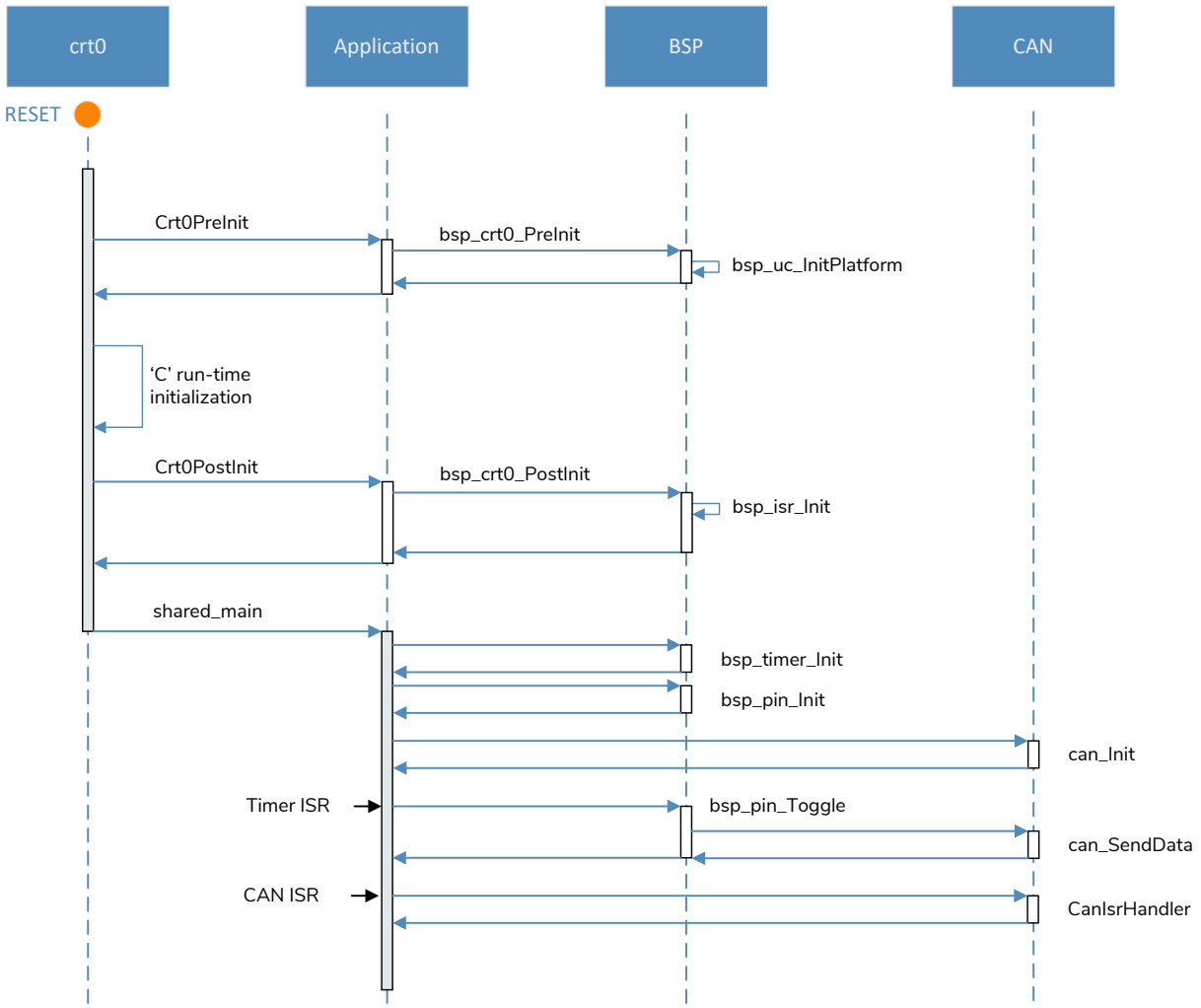

Fig. 5. Simplified individual core execution flow

3.5. Interrupt Handling

Implemented interrupt handling uses low-level BSP functions, for example to register each interrupt service routine (`bsp_isr_RegisterHandler`).

The CAN initialization function assigns the `'FlexCAN_0_MB4_7'` interrupt vector and sets the interrupt vector priority.

Code 2. /app/can.c

```
bsp_isr_RegisterHandler(UC_CAN0_ISR_MB4_7, 4, CanIsrHandler);
```

The example uses the SW interrupt mode.

3.6. CAN module configuration

The CAN configuration sets these parameters:

- MCAN mode with standard CAN frame
- 8 byte TX/RX buffer
- Rx FIFO new message interrupt
- Rx FIFO in overwrite mode
- f_{CAN} = 40MHz
- Bitrate 500kbaud

The CAN bit rate is calculated from the input clock frequency (f_{CAN}), CAN clock Pre-scaler, and both phase segment size.

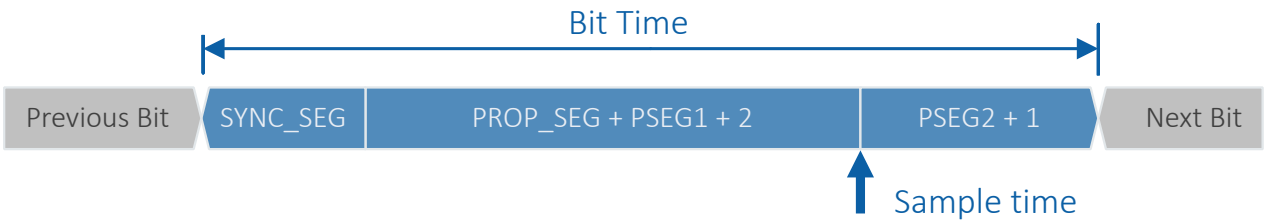

Fig. 6. Time segments of a CAN-Bit

Tab. 3. CAN1 protocol configuration

Parameter	Value
Bit Rate [kbit/s]	500
Prescaler Division Factor (PRESDIV)	5
Synchronization (RJW) [t_q]	3
Phase Segment 1 (PSEG1) [t_q]	8 (7)
Phase Segment 2 (PSEG2) [t_q]	4 (3)
Propagation Segment (PROPSEG) [t_q]	3 (2)
SYNC_SEG [t_q]	1
Sample point [%]	75
t_q [ns]	(NBRP) * ($1/f_{CAN}$) = 625
Total Number of time quanta	16

For more configuration details see `can_Init()` function snippet below.

Code 3. /app/can.c

```
void can_Init(void)
{
...

    /* Initialize the Module configuration register */
    FLEXCAN_0.MCR.B.FEN = 1; /* FIFO enabled */
    FLEXCAN_0.MCR.B.IDAM = 0; /* FIFO, One full ID (standard or extended) per
                                 filter element*/
    FLEXCAN_0.MCR.B.BCC = 1; /* Individual Rx masking and queue feature are
                                enabled*/
    /* Set individual mask register for ID filtering for FIFO, hits all IDs */
    FLEXCAN_0.RXIMR[0].R = 0;
    FLEXCAN_0.RXIMR[1].R = 0;
    FLEXCAN_0.RXIMR[2].R = 0;
    FLEXCAN_0.RXIMR[3].R = 0;
    FLEXCAN_0.RXIMR[4].R = 0;
    FLEXCAN_0.RXIMR[5].R = 0;
    FLEXCAN_0.RXIMR[6].R = 0;
    FLEXCAN_0.RXIMR[7].R = 0;

    /* Initialize the Control register */
    /* Baud rate initialization */
    FLEXCAN_0.CTRL.B.PRESDIV = CAN_CTRL_PRESDIV;
    FLEXCAN_0.CTRL.B.PSEG1 = CAN_CTRL_PSEG1;
    FLEXCAN_0.CTRL.B.PSEG2 = CAN_CTRL_PSEG2;
    FLEXCAN_0.CTRL.B.PROPSEG = CAN_CTRL_PROPSEG;
    FLEXCAN_0.CTRL.B.RJW = CAN_CTRL_RJW;

...

    FLEXCAN_0.IMASK1.B.BUF5M = 1; /* MB5 completed reception or frames
                                     available in the FIFO. Interrupt
                                     is enabled */
    FLEXCAN_0.MB[0].MB_CS.R = 0x04000000; /* MB CODE: 0100b EMPTY: MB is active
                                             and empty */

    FLEXCAN_0.MCR.B.MDIS = 0; /* Enable the FlexCAN module */
    FLEXCAN_0.MCR.B.HALT = 0; /* No Freeze Mode request */
...

}
```

3.7. CAN External Loop Back Test mode

The default example configuration needs an external device (CAN node) for message frame acknowledge and to data visualization. Therefore, the example provides the user with an option to switch the CAN module easily into the test mode. For more information about the "External Loop Back mode" see Microcontroller's reference manual.

Code 4. /app/can.h

```
...
/* External Loop Back mode enable(1) or disable(0)
 * See RM Chapter - External Loop Back mode */
#define CAN_LOOP_BACK_TEST    0
```

4. Example Import & Build

Follow these steps to import an example project to the HighTec IDE environment:

1. From menu **File** → **Import** → **General** choose an option `Existing Projects into Workspace`
2. Browse for your project location
3. Select project
4. Leave the `copy to the workspace option` check box empty
5. Click Finish

Activate the project from menu **Project** → **Set Active Project**.

Build the project from the menu **Project** → **Build Project**.

The output binary file is located under the `_irom_build` folder.

Appendix A: Document References

[1] "SPC5x c-startup - 'C' run-time initialization", HighTec EDV Systeme GmbH, 2018

[2] "SPC5x c-startup - Linker file template", HighTec EDV Systeme GmbH, 2018

[3] "SPC5x c-startup - Hardware Abstraction Layer", HighTec EDV Systeme GmbH, 2018

Appendix B: Document history

Version	Date	Changes to the previous version
1.0	February 2018	Initial version