SPC574K72 CAN example

Quick Guide

Version 1.0, March, 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 Suport Package

GPIO
: General Purpose Input Output

EVB
: Evaluation Board

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

- SPC574K72 device
- An evaluation board (SPC57xxMB + MPC5746M_176DS)
- HighTec Development Suite, version: 4.9.3.0

2.2. HW resources

Hardware resources used by this example:

HW unit	channel / output pin	function
M_CAN_1_TX	PA[10]	Core [2] M_CAN transmit output
M_CAN_1_RX	PA[11]	Core [2] M_CAN receive input
default resources from the c-startup example		
GPIO	PA [0]	Core [0] LED control
GPIO	PA [2]	Core [2] LED control
PIT 0	Channel [0]	Core [0] periodic interrupt
PIT 0	Channel [2]	Core [2] periodic interrupt

Tab. 1. HW resources

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.

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
J35	CAN	1-2: 5V_SR to PHY U1 VCC, 3-4: 12V to PHY U1 VBAT
J37	CAN0_TX	2-3: PA[10] to CAN TX

Jumper	PCB Legend	Description
J38	CAN1_RX	2-3: PA[11] to CAN RX

Tab. 2. SPC57xxMB CAN configuration

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

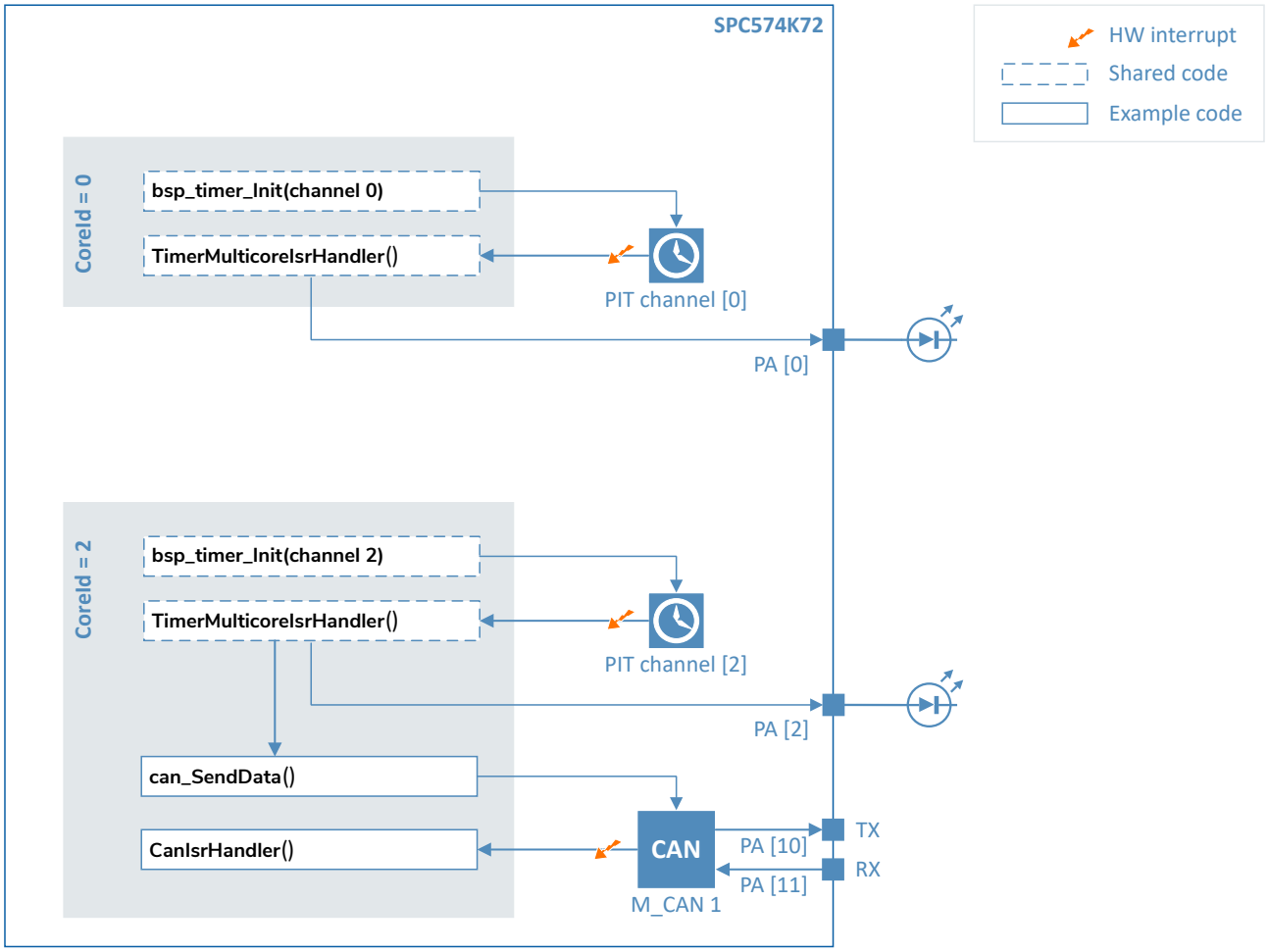

Fig. 1. System block diagram

- The example code runs on a single core. Other cores run the default functionality defined by SPC5xx_c_startup.
- The example initializes M_CAN1 module to MCAN mode 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, .XTD = 0, .BRS = 0, .DLC = 8, .EDL = 1,      \
 .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.
- Each CAN frame transmission toggles GPIO PA[2] to enable a visualization of the transmit event if the pin is connected to the LED jumper on the EVB.

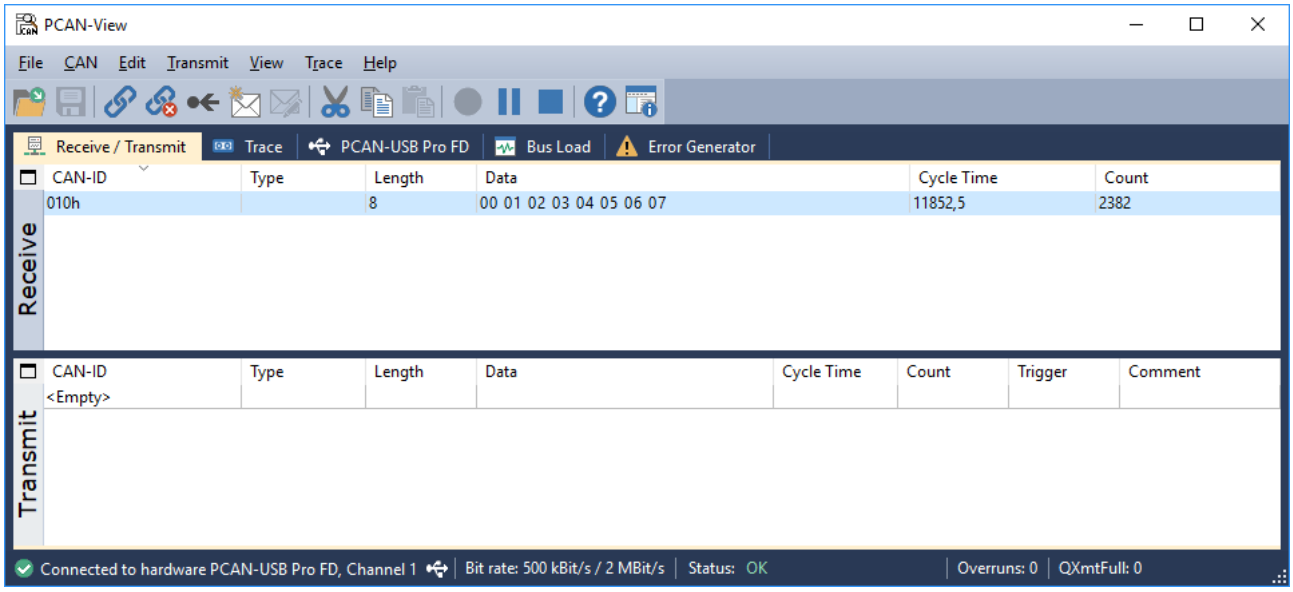

Fig. 2. Received CAN frame on the Host

3.3. CAN frame receive

- The application receives data to FIFO0 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`.

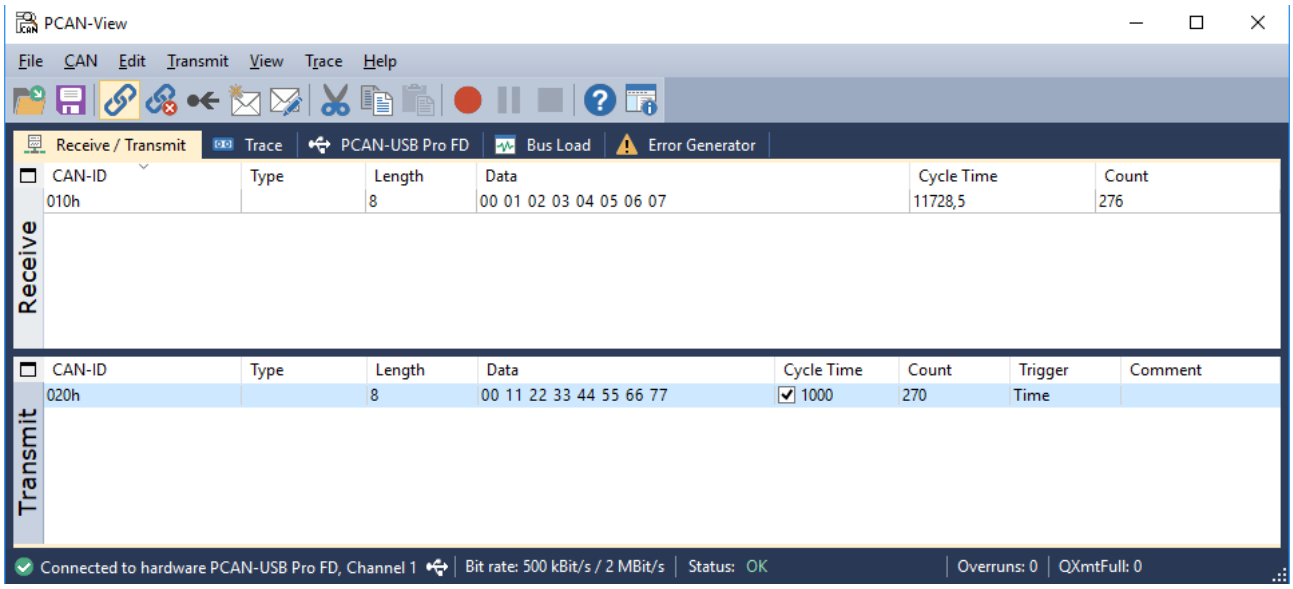

Fig. 3. CAN frame transmitted from the host

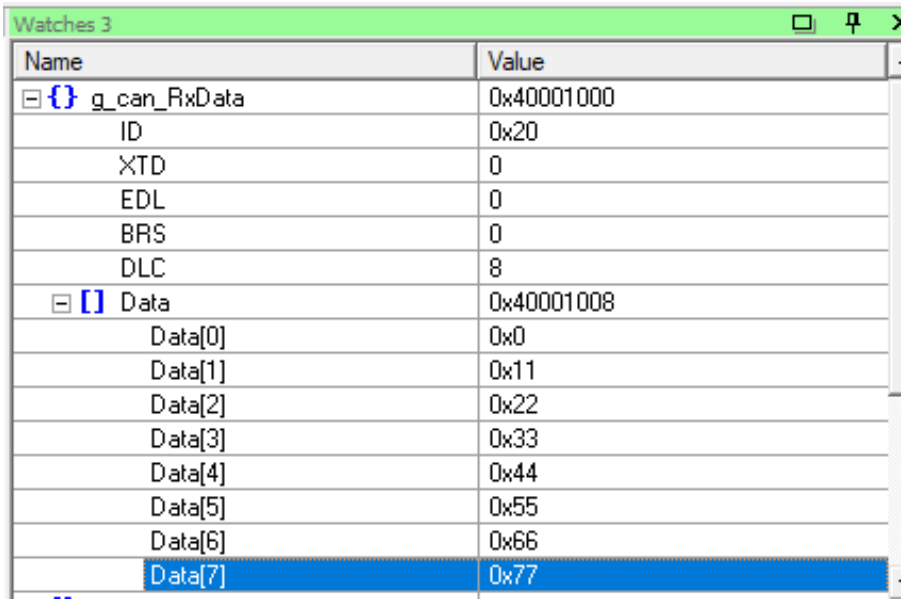

Fig. 4. Received CAN frame

3.4. Application Flow

The UC_CORE_IOP core is responsible for the CAN communication.

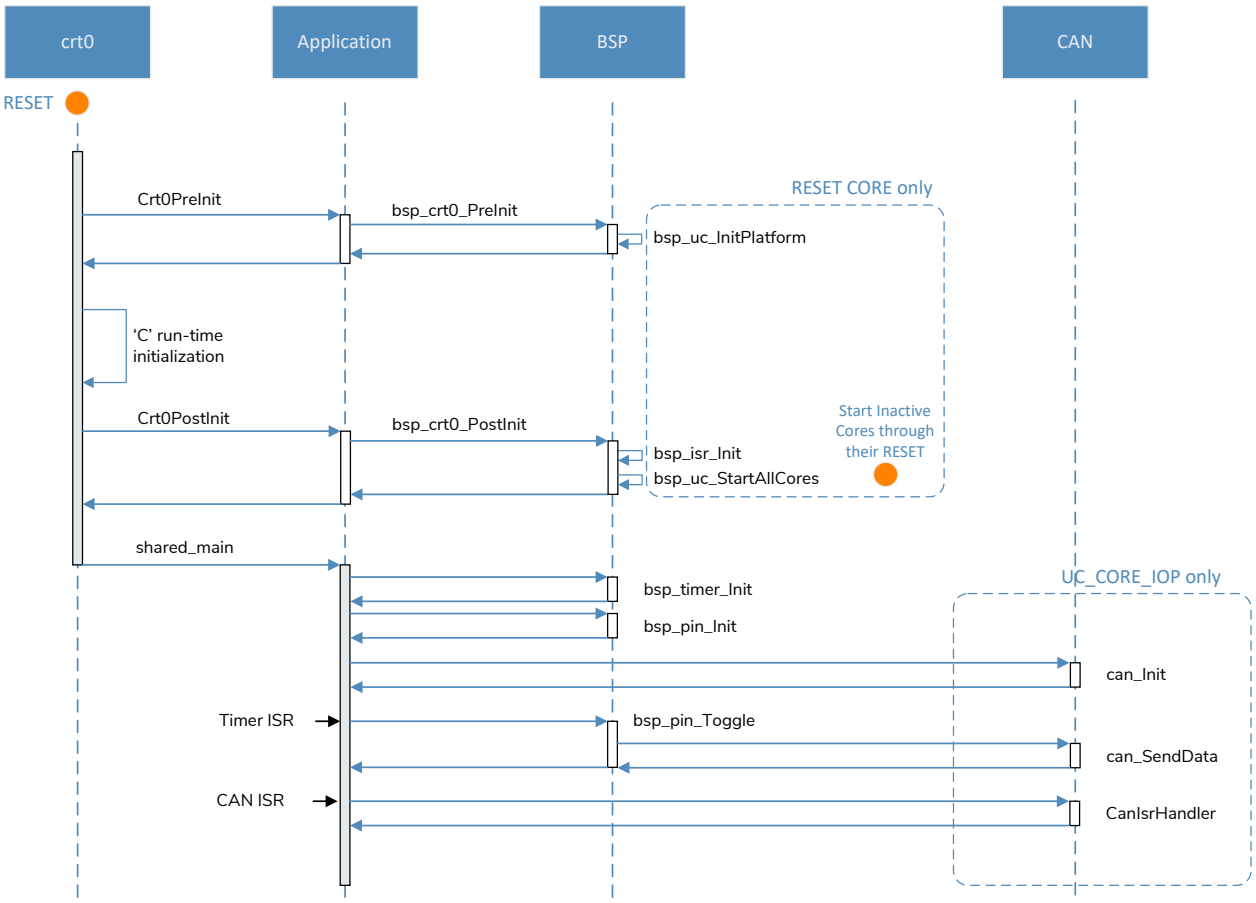

Fig. 5. Simplified individual core execution flow

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 'RxFIFO new message event' to the interrupt line `M_CAN_INT` corresponding to the interrupt vector and sets the interrupt vector priority.

Code 2. /app/can.c

```
bsp_isr_RegisterHandler(UC_CAN_ISR_INT0, 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

- No timestamp used
- Rx FIFO 0 new message interrupt
- Rx FIFO 0 in overwrite mode
- f_{CAN} = 40MHz (derived from PLL0)
- 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

Parameter	Value
Bit Rate [kbit/s]	500
Baud Rate Prescaler (NBRP)	5
Synchronization (NSJW) [t_q]	1
Phase Segment 1 (NTSEG1) [t_q]	16 (14)
Phase Segment 2 (NTSEG2) [t_q]	4 (3)
Sample point [%]	85
t_q [ns]	(NBRP + 1) * (1/f_{CAN}) → 100
Total Number of time quanta	20

Tab. 3. CAN1 protocol configuration

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

Code 3. /app/can.c

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

    /* Configuring the time stamp and timeout registers */
    MCAN_1.TSCV.R = 0; /* Time stamp is not used */
    MCAN_1.TOCC.R = 0; /* Timeout counter is not used */

    /* Configuring the interrupt registers */
    MCAN_1.IE.B.RF0NE =  1;      /* Rx FIFO 0 New Message Interrupt Enable */
    MCAN_1.ILS.B.RF0NL = 0;      /* RxFIFO 0 new message is assigned to interrupt line 0 */
    MCAN_1.ILE.B.EINT0 = 1;      /* Enable interrupt from line 0 */

    MCAN_1.GFC.R = 0;            /* The standard and Extended frames are accepted in Rx
FIFO_0*/
    MCAN_1.XIDAM.R = 0x1FFFFFFF; /* Extended ID mask, AND mask is not active */

    /* Writing the start addresses of all the M_CAN Buffers */
    MCAN_1.RXF0C.R = CAN_RX_FIFO_OFFSET;
    MCAN_1.RXF0C.B.F0S = 1;      /* One Rx FIFO 0 element */
    MCAN_1.RXF0C.B.F0WM = 0;     /* Watermark interrupt disabled */
    MCAN_1.RXF0C.B.F0OM = 1;     /* FIFO 0 overwrite mode*/

    MCAN_1.RXF1C.B.F1S = 0;      /* NO Rx FIFO 1 element */

    MCAN_1.SIDFC.R = 0x0;        /* No standard Message ID filter */
    MCAN_1.XIDFC.R = 0x0;        /* No extended Message ID filter */

    MCAN_1.TXBC.R = CAN_TX_FIFO_OFFSET;
    MCAN_1.TXBC.B.NDTB = 1;      /* One Dedicated Tx Buffers */
    MCAN_1.TXBC.B.TFQS = 0;      /* No Tx Buffers used for Tx FIFO*/
    MCAN_1.TXBC.B.TFQM = 0;      /* Tx FIFO operation */

    MCAN_1.TXEFC.R = 0x0;        /* Tx events disabled */

    /* Writing to the RAM for filter configuration */
    /* No filter configuration */

    //MCAN_1.SIDFC.B.FLSSA = (CAN_STD_FILTER_OFFSET >> 2);
    MCAN_1.SIDFC.R = CAN_STD_FILTER_OFFSET;
    MCAN_1.SIDFC.B.LSS   = 1;

    /* M_CAN Baud Rate configuration*/
    MCAN_1.NBTP.R = CAN_BTP_CFG;

    CCCU.CCFG.R = 0x444; /* Bypass Clock Calibration on CAN unit */

...
}
```

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

```
1 ...
2 /* External Loop Back mode enable(1) or disable(0)
3  * See RM Chapter - External Loop Back mode */
4 #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` 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: Release Notes

Version	Date	Changes to the previous version
1.0	March 2018	Initial version based on SPC574K72 c-startup example V1.1

HighTec EDV-Systeme GmbH
Feldmannstrasse 98, D-66119 Saarbrücken
info@hightec-rt.com
+49-681-92613-16
www.hightec-rt.com