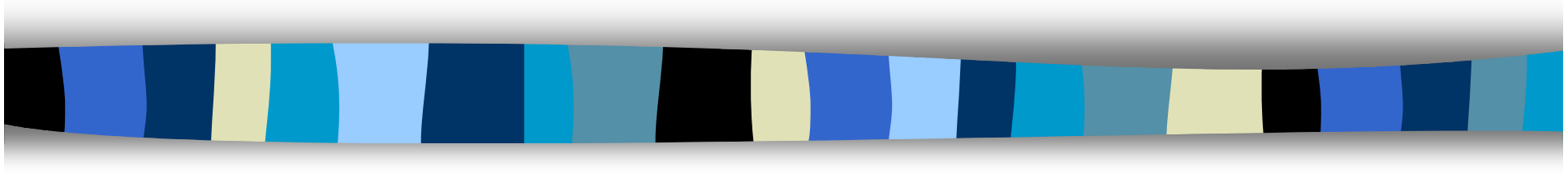
Interfejs CAN



Podstawy interfejsu oraz jego zastosowanie na przykładzie układu Cypress PSoC5LP

Gdańsk, czerwiec 2017

Plan prezentacji

- Podstawowe informacje o standardzie CAN
- Kontroler CAN w układzie PSoC5LP
- Przykład implementacji transmisji pomiędzy zestawami Cypress PSoC5LP

Literatura

[1] 1991, Robert Bosch GmbH, Postfach 30 02 40, D-70442 Stuttgart „CAN Specification Version 2.0 ”, http://www.bosch-semiconductors.de/media/ubk_semiconductors/pdf_1/canliteratur/can2spec.pdf

[2] Ranjith M., dokument AN52701 „PSoC® 3 and PSoC 5LP – Getting Started with Controller Area Network (CAN) ”, www.cypress.com

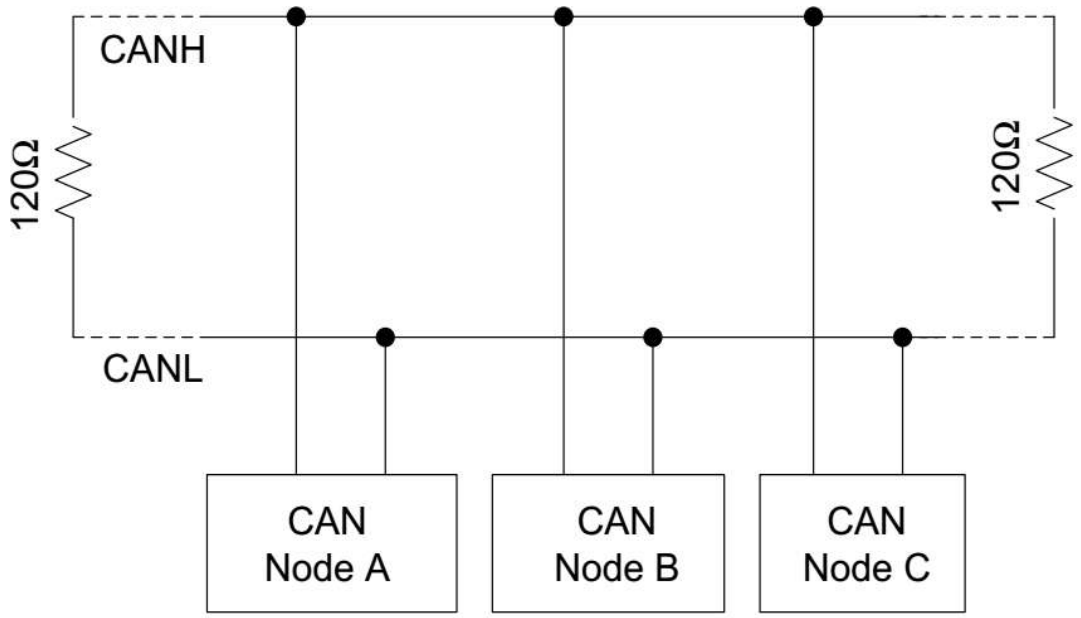
[3] CAN Protocol Tutorial https://www.kvaser.com/can-protocol-tutorial/

[4] TJA1050 DataSheet, www.philips.com

[4] „PSoC® 5LP: CY8C58LP Family Datasheet”, www.cypress.com

Warstwa fizyczna łącza [2, 3]

CAN ma kilka różnych możliwych do zastosowania warstw fizycznych, zależą on od zastosowanego transceivera. Na płytkach prototypowych użyto układu Philips TJA1050 w najpopularniejszym standardzie ISO 11898-2.

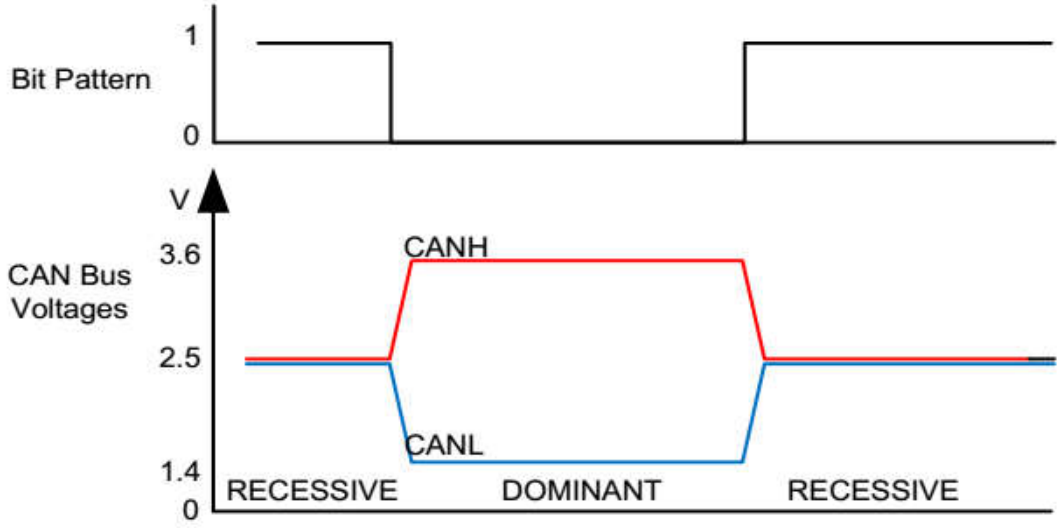


Rys.1. Węzły CAN połączone z siecią CAN [2].

Warstwa fizyczna łącza [2, 3] c.d.

CAN używa linii różnicowych, „1” logiczna jest reprezentowana przez takie same poziomy sygnałów (ok. 2,5V) a „0” logiczne jest reprezentowane przez różnicę napięć pomiędzy liniami CANH i CANL rzędu 1,5V –3V.

Logiczna „1” jest typu „recessive” a logiczne „0” jest typu „dominant”. W przypadku wystawienia przez różne węzły sieci różnych wartości na liniach CAN tworzone jest logiczne AND i wygrywa logiczne „0”. W przypadku braku wysterowania linii CAN pojawia się na nich samoistnie logiczne „1”.



Rys.2. Stany logiczne i odpowiadające im napięcia na liniach CAN w standardzie ISO 11898-2 [2].

Szybkości transmisji CAN ISO 11898-2

- Szybkość transmisji zależy od długości szyny CAN
- Taka zależność spowodowana jest tym, że każdy z transceiverów musi równocześnie nadawać i odbierać stan szyny w celu ciągłego sprawdzania/prowadzenia arbitrażu

Tab.1. Szybkość transmisji w standardzie ISO 11898-2 w zależności od długości przewodów [2].

Baud Rate (in bits per second)	Typical Cable Length (in meters)
1 Mbps	40
500 kbps	100
250 kbps	200
125 kbps	500
10 kbps	6000

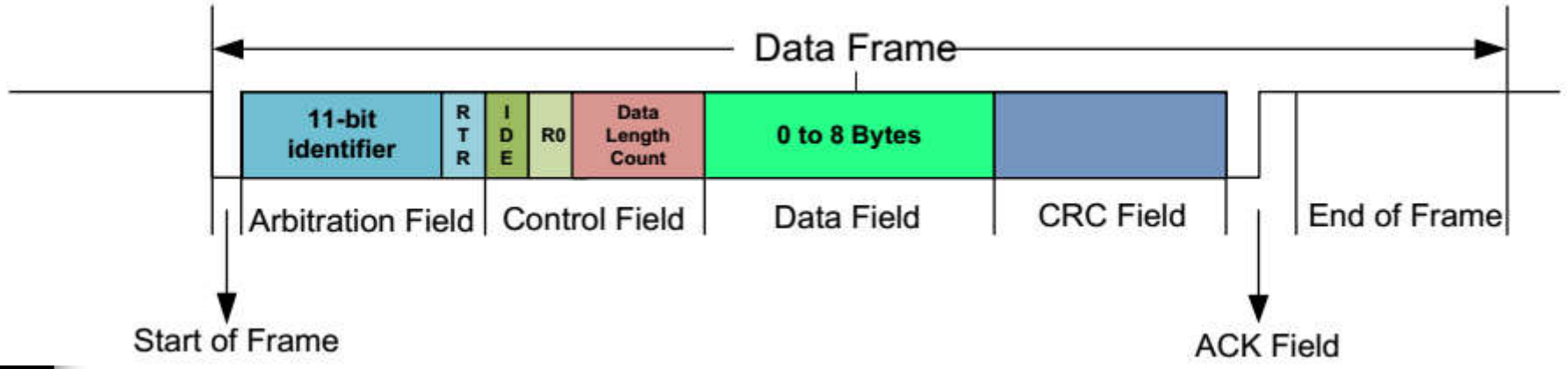
Warstwa transmisyjna [2, 3]

Każdy węzeł może rozpocząć transmisję kiedy szyna CAN jest w stanie bezczynności (jest w stanie „1” logicznej). Przez szynę można przesyłać komunikaty w ustalonych formatach nazwanych ramkami.

CAN definiuje 4 następujące możliwe typy ramek:

- ramka danych (ang. Data Frame) – przenosi dane z nadajnika do odbiornika,
- ramka zdalna (ang. Remote Frame) – nadawana przez węzeł sieci w celu zażądania ramki danych,
- ramka błędu (ang. Error Frame) – nadawana przez dowolny węzeł gdy wykryty zostanie błąd na szynie CAN,
- ramka przepełnienia (ang. Overload Frame) – nadawana w celu uzyskania dodatkowego czasu pomiędzy kolejnymi ramkami.

Ramka danych [2, 3]

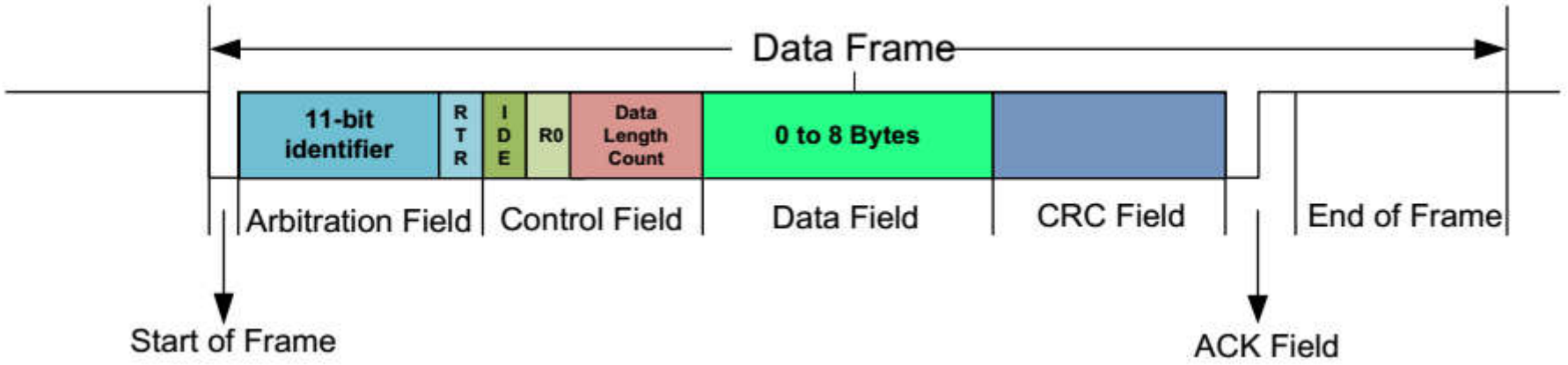


Rys.3. Ramka danych CAN z adresowaniem standardowym, 11-to bitowym [2].

Arbitration Field – pole składające się z 2 części: **Identifier** oraz Remote Transmission Request bit (**RTR**). **Identifier** jest używany w celu opisu znaczenia danych przenoszonych przez ramkę. Każdy węzeł odbiorczy, na podstawie wartości **Identifier** decyduje czy dane są dla niego interesujące. W zależności od długości pola **Identifier** definiowane są komunikaty standardowe (11 bitów) oraz rozszerzone (29 bitów).

RTR = „0” dla ramki danych i „1” dla ramki zdalnej.

Ramka danych [2, 3] c.d.



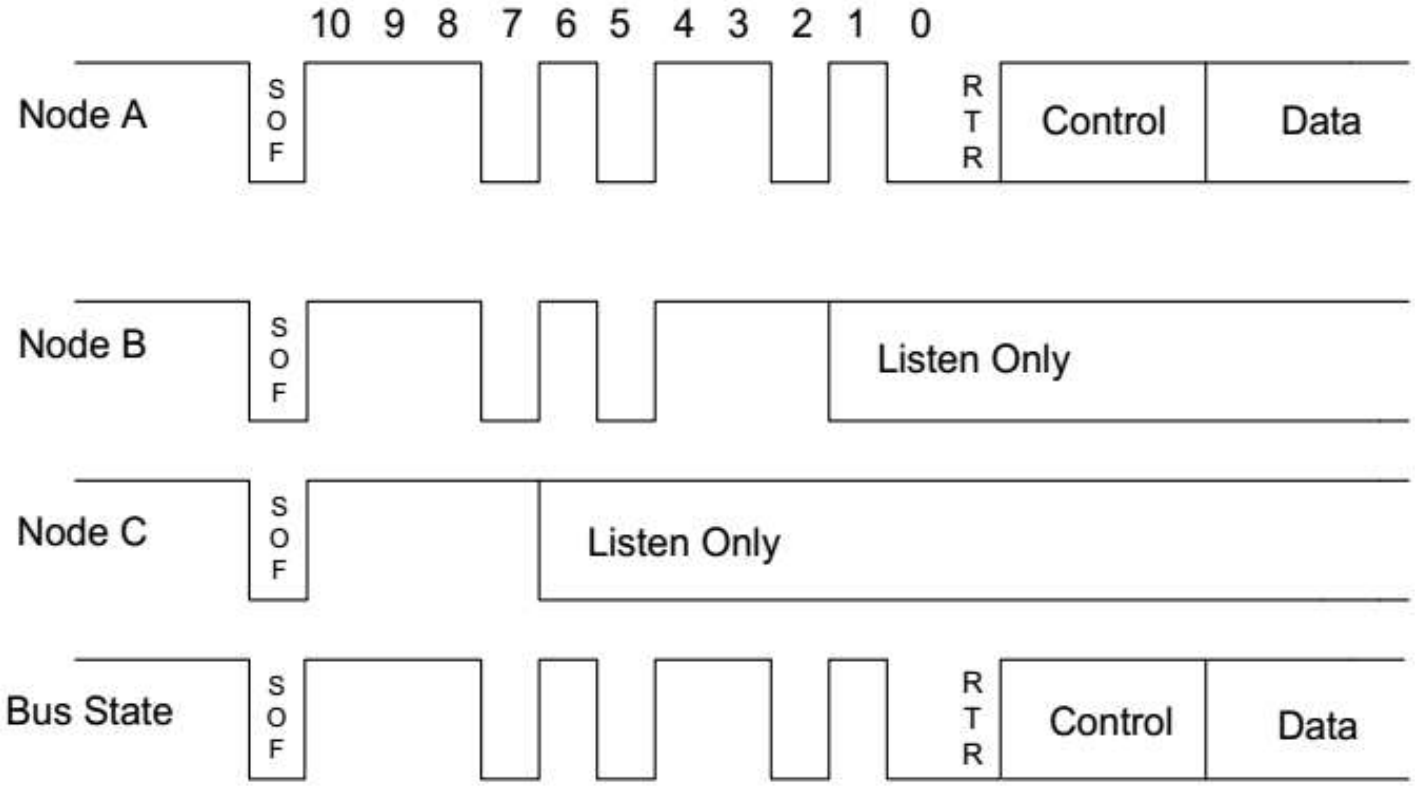
Control Field – pole 6-cio bitowe składające się z 2 bitów do późniejszego zastosowania i 4 bitów określających długość danych.

CRC Field – pole sprawdzenia poprawności danych przez wyliczenie sumy kontrolnej (cyclic redundancy check).

ACK Field – pole używane przez węzły odbiorcze do potwierdzenia poprawnego odbioru ramki.

Ramka zdalna różni się od ramki danych wartością bitu **RTR** oraz zerową długością pola danych.

Arbitraż na szynie CAN [2, 3]



Rys.4. Arbitraż na szynie CAN. Ponieważ „0” logiczne dominuje na linii CAN węzeł nadający „0” wygrywa arbitraż i przejmuje szynę, węzeł przegrywający przechodzi do stanu nasłuchiwania [2].

Detekcja i zarządzanie wykrytymi błędami [2].

Węzeł CAN może wykryć następujące błędy:

- błąd bitu – błąd wykrywany w nadajniku w momencie wykrycia niezgodności między nadawanym bitem a stanem linii CAN, wykrywanie tego błędu następuje w nadajniku,
- błąd ramki – błąd jest wykrywany jeśli dane nie są zgodne z formatem jednej z 4 możliwych ramek, wykrywanie tego błędu następuje w odbiorniku,
- błąd kolejności (ang. stuff error) – nadajnik w trakcie nadawania danych wykrywa ciąg 5 kolejnych identycznych wartości, jeśli coś takiego występuje wówczas automatycznie wstawiany jest bit o przeciwnej wartości, taka czynność jest nazywana w j. ang. bit stuffing, błąd jest wykryty jeśli zostaje wykrytych 6 identycznych kolejnych wartości, wykrywanie tego błędu następuje w odbiorniku,
- błąd CRC – jest wykryty jeśli suma kontrolna CRC jest niezgodna, wykrywanie tego błędu następuje w odbiorniku,
- błąd potwierdzenia – jest wykryty jeśli węzeł nadający nie wykryje potwierdzenia przez inne węzły w postaci dominującego zera, wykrywanie tego błędu następuje w nadajniku.

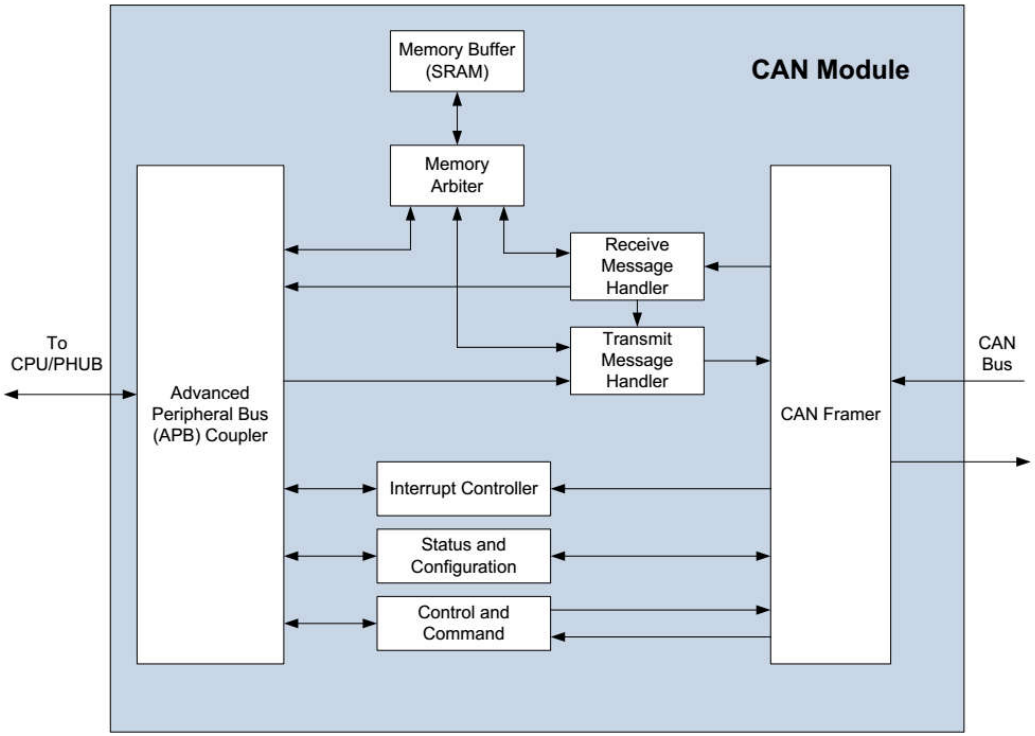
Detekcja i zarządzanie wykrytymi błędami [2] c.d.

Działanie po wykryciu błędu:

- Jeśli zostanie wykryty błąd wówczas odbiornik generuje ramkę błędu a nadajnik ponawia transmisję ramki danych aż do osiągnięcia prawidłowej transmisji.
- W przypadku gdy wadliwy węzeł się zawiesił i ciągle generuje ramki błędu wówczas jego możliwości nadawcze są usuwane. Każdy węzeł ma liczniki błędów dla błędów nadawczych i odbiorczych. W zależności od stanu liczników błędów każdy z węzłów CAN może być w 3 różnych stanach:
  - Error Active State: liczniki błędów nadawczych i odbiorczych mają wartości poniżej 128, taki węzeł pracuje normalnie,
  - Error Passive State: liczniki błędów nadawczych i odbiorczych mają wartości równe lub większe niż 128, taki węzeł może ale nie musi pracować normalnie,
  - Bus Off State: licznik błędów nadawczych jest równy lub większy niż 256, węzeł w tym stanie jest wyłączony z komunikacji i nie wpływa na stany na szynie CAN.

CAN w układzie PSoC5LP [2]

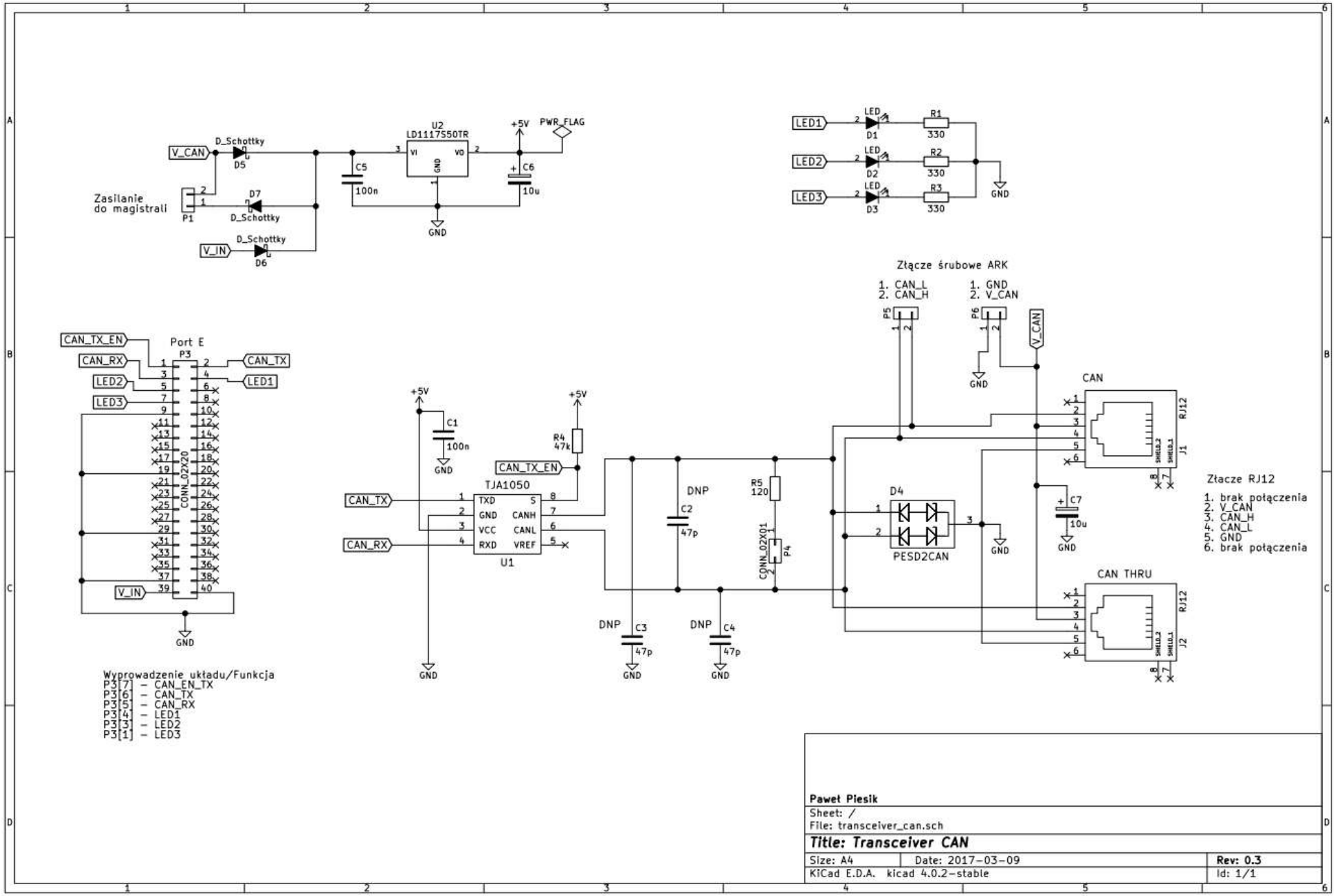
Kontroler CAN w układzie PSoC5LP zapewnia warstwę logiczną, do poprawnej pracy wymagany jest zewnętrzny transceiver np. TJA1050 (NXP), SN65HVD1050-EP (TI) lub inny.



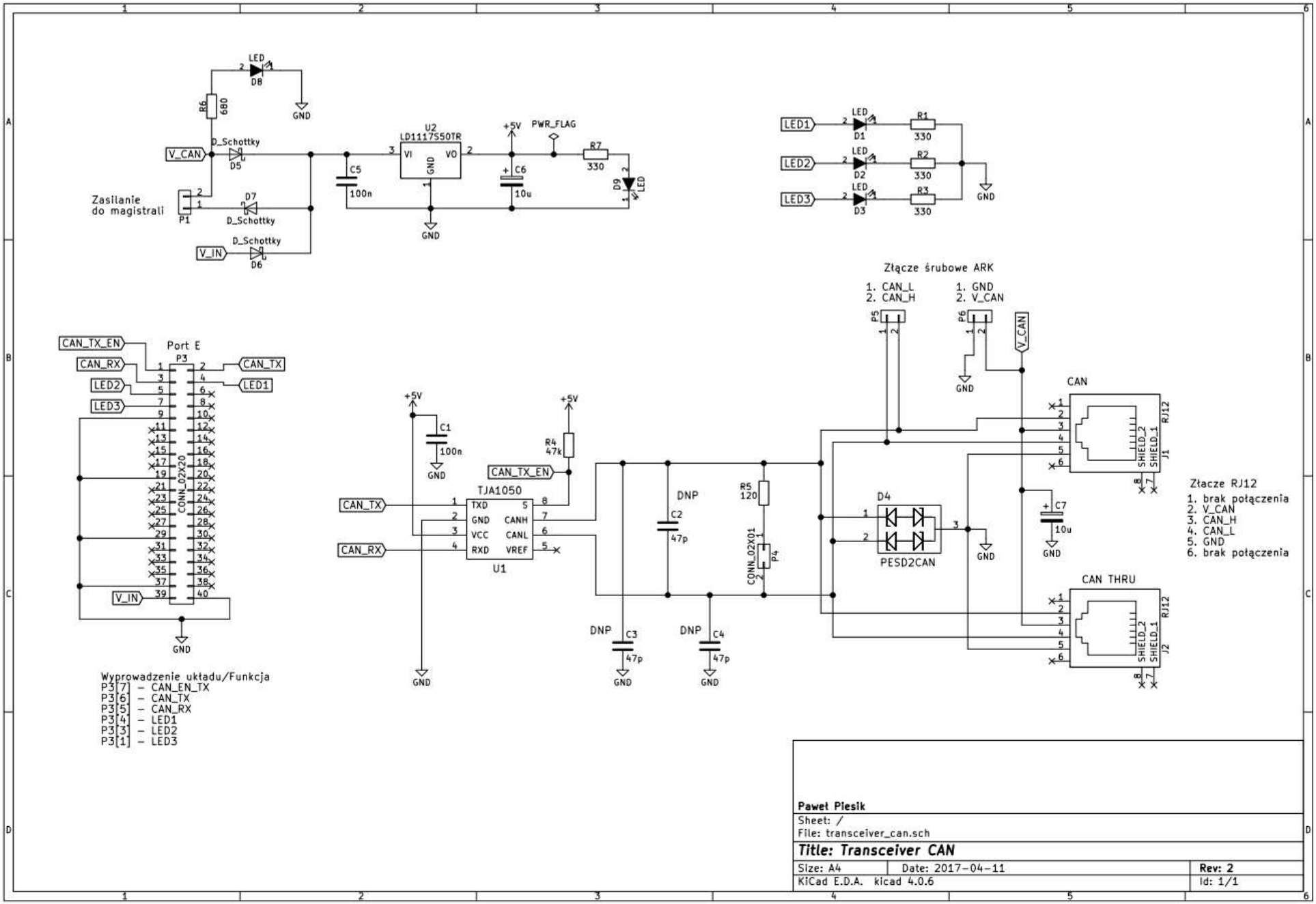
Rys.5. Blokowy schemat kontrolera CAN w układzie PSoC5LP [2].

Blok CAN w IDE PSoC Creator

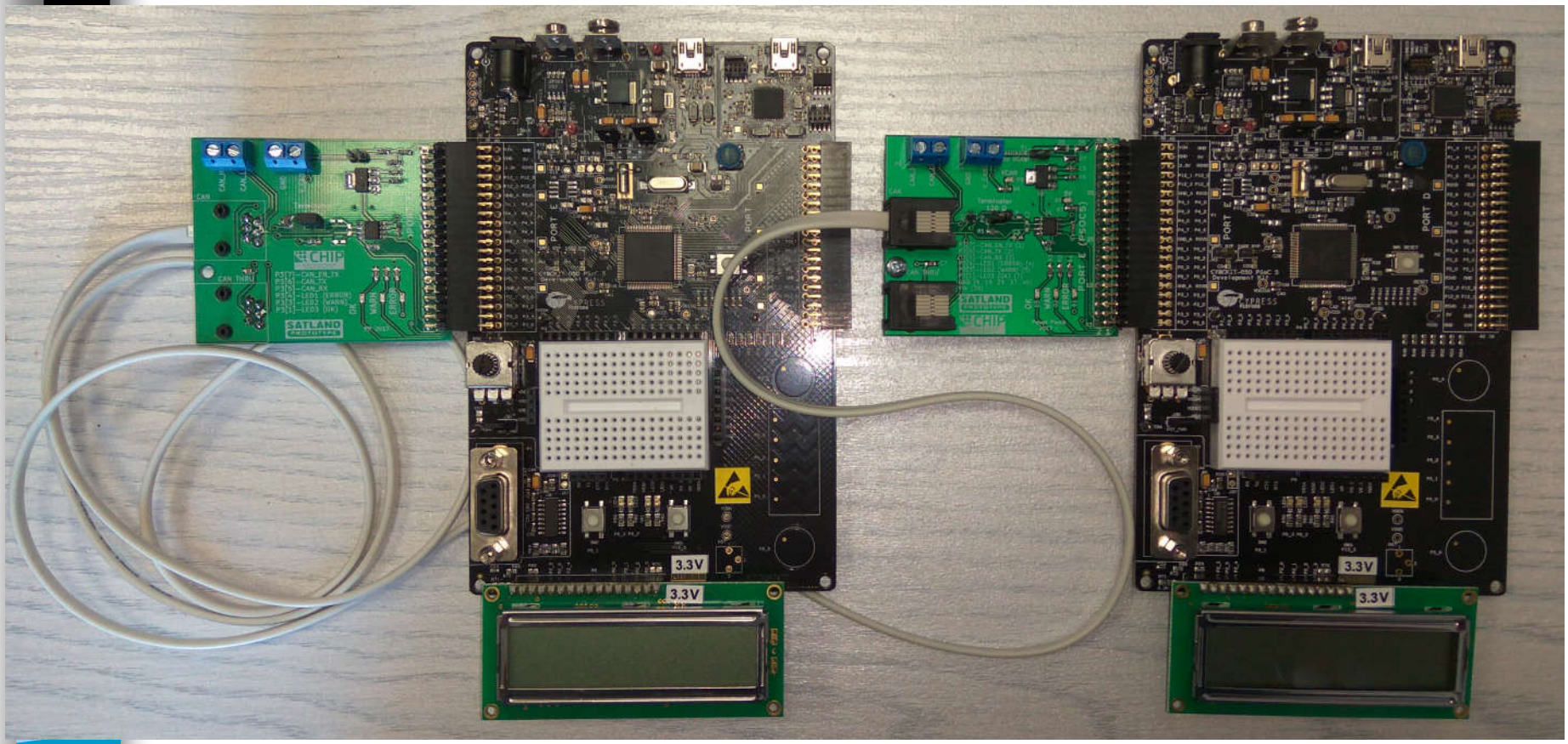
- Sprzętowy blok CAN ma 3 lub 2 wyprowadzenia (wyjście TX_En jest opcjonalne)
- Blok CAN może pracować w trybie Full CAN:
  - parametry bloku CAN są ustawiane w GUI,
  - wymaga minimalnej ilości programowania,
  - realizuje sprzętowe filtrowanie wiadomości,
  - może odbierać wiadomości wyłącznie o pojedynczym identyfikatorze,
- oraz w trybie Basic CAN:
  - parametry bloku CAN są ustawiane programowo,
  - CPU jest używane do filtrowania wiadomości, każda odebrana wiadomość generuje przerwanie,
  - może obsłużyć wiadomości w pewnym zakresie identyfikatorów.



Rys.6. Schemat transceivera CAN z układem TJA1050 przystosowanego do płytki CY8CKIT-050, wer.1.



Rys.7. Schemat transceivera CAN z układem TJA1050 przystosowanego do płytki CY8CKIT-050, wer.2.

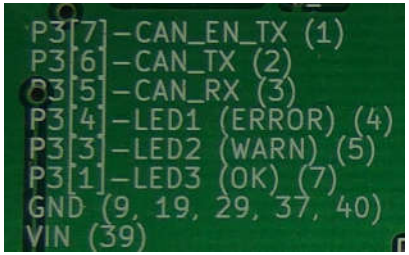


Rys.8. Połączenie 2 płytek CY8CKIT-050 przy użyciu transceiverów CAN w wer.1 i wer.2.

Uwagi do połączenia płytek CY8CKIT-050 poprzez transceiver CAN wer.1 i 2

- Obie wersje transceiverów są ze sobą kompatybilne i mogą być używane razem. Różnice między nimi to:
  - odwrócony montaż złącz RJ12,
  - dodatkowe diody LED sygnalizujące obecność zasilania (w wer.2).
- Płytki transceiverów należy włożyć do złącza PORT E na płytce ewaluacyjnej CY8CKIT-050 PSoC5LP.
- Oba porty RJ12 są ze sobą połączone równolegle i przewód połączeniowy można wpiąć w dowolne z gniazd. Zdwojona liczba gniazd umożliwia łatwe utworzenie sieci z większą ilością węzłów.
- W transceiverach umieszczonych na końcach szyny CAN należy umieścić zworę na złączu P4 (Terminator 120 Ω) a pozostałe węzły powinny mieć te wyprowadzenia rozwarte.
- Wymagane połączenia pinów projektu w IDE PSOC Creator przedstawione są na poniższych dwóch rysunkach:

Połączenia na PCB:



Wymagane połączenia w IDE PSOC Creator:

	Name	Port
	\LCD_Char_1:LCDPort[6:0]\	P2[6:0]
	RX_1	P3[5]
	TX_1	P3[6]
	Tx_En_1	P3[7]

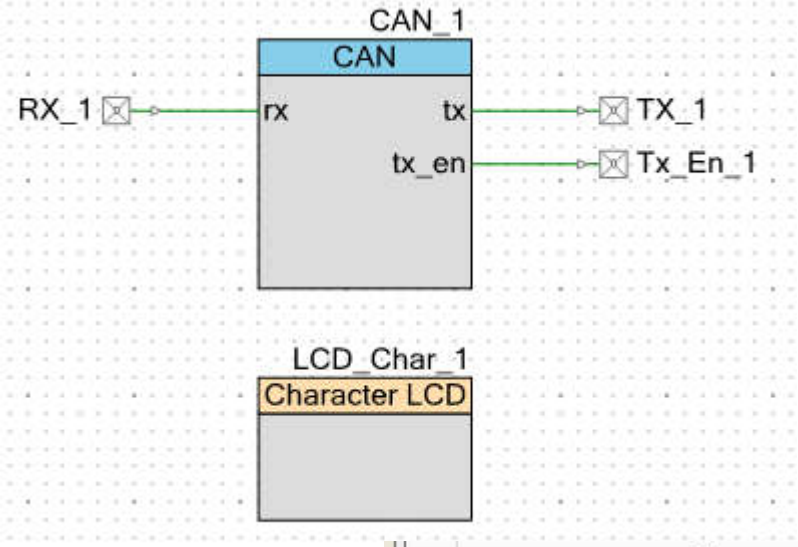
Przykład prostego projektu użycia CAN [2]

- Na jednej płytce nadawane będą kolejne liczby (jednobajtowe) a druga płytka będzie te liczbę odbierać i wyświetlać na wyświetlaczu znakowym LCD.
- Oba bloki CAN (nadawczy i odbiorczy) będą użyte w trybie Full CAN (uproszczone programowanie z filtrowaniem sprzętowym).

Projekt „Nadajnik” – krok po kroku

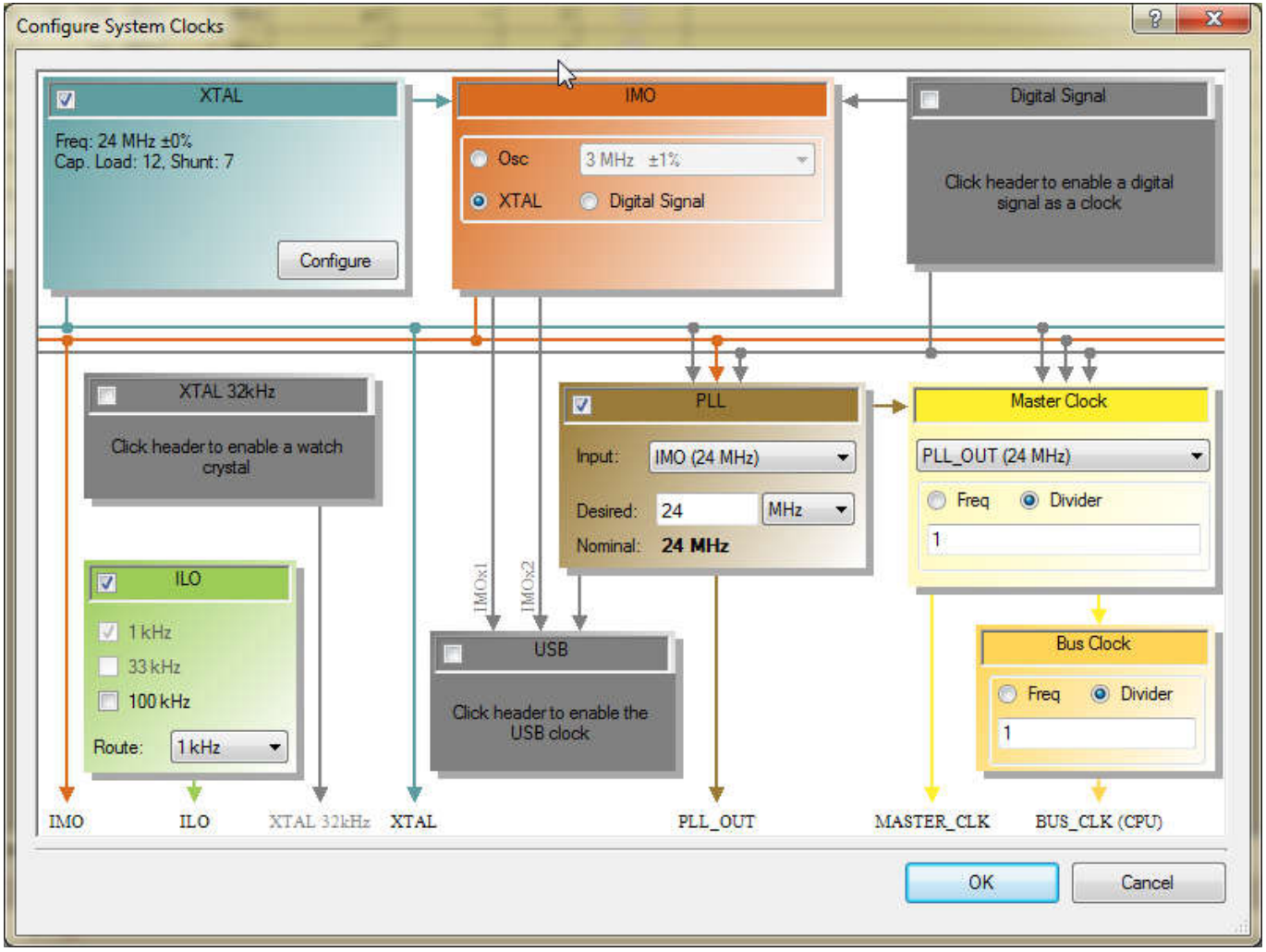
- Tworzymy nowy projekt w IDE PSOC Creator, wybieramy Menu: File -> New -> Project i podajemy nazwę „Nadajnik”.
- Na schemacie umieszczamy bloki „CAN Controller Macro” oraz „Character LCD”.
- Dodajemy wyjście cyfrowe i nadajemy nazwę Tx_En_1.
- Wykonujemy przypisania wyprowadzeń układu PSoC5LP.

Widok schematu i przypisań projektu „Nadajnik”

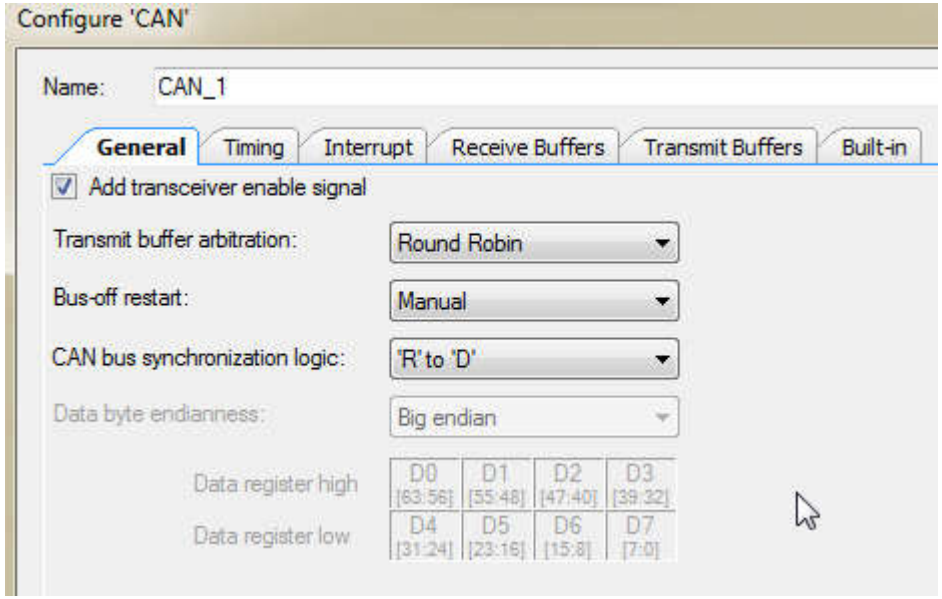


	Name	Port	Pin	Lock
	\LCD_Char_1:LCDPort[6:0]\	P2[6:0]	2, 1, 99…95	☑
	RX_1	P3[5]	49	☑
	TX_1	P3[6]	51	☑
	Tx_En_1	P3[7]	52	☑

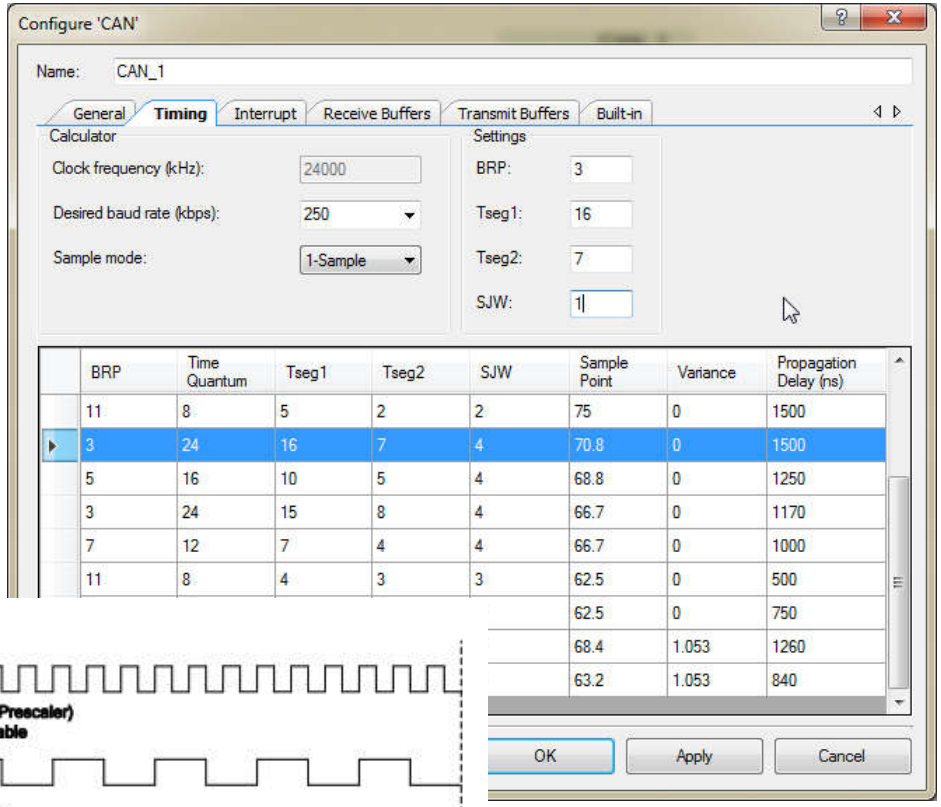
Konfiguracja zegarów jak na rys.

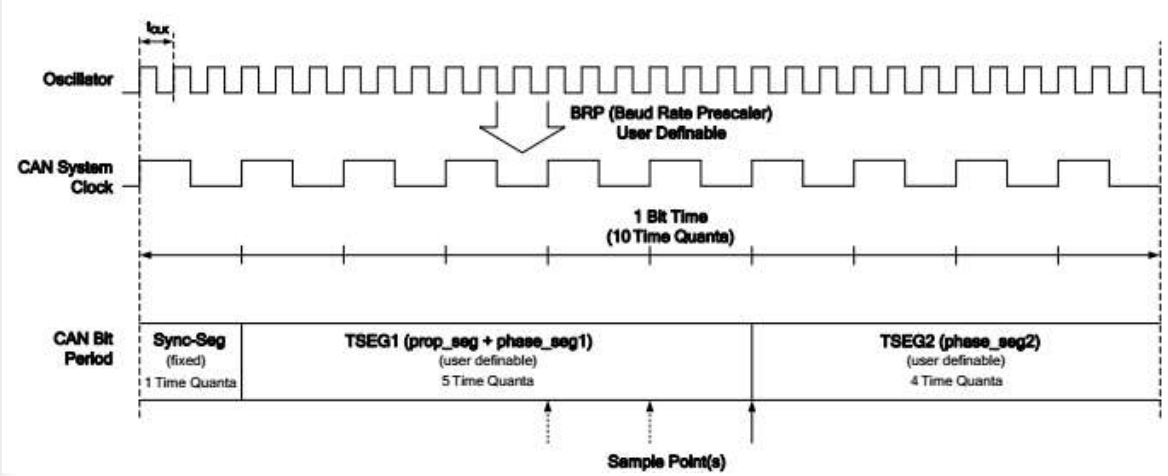


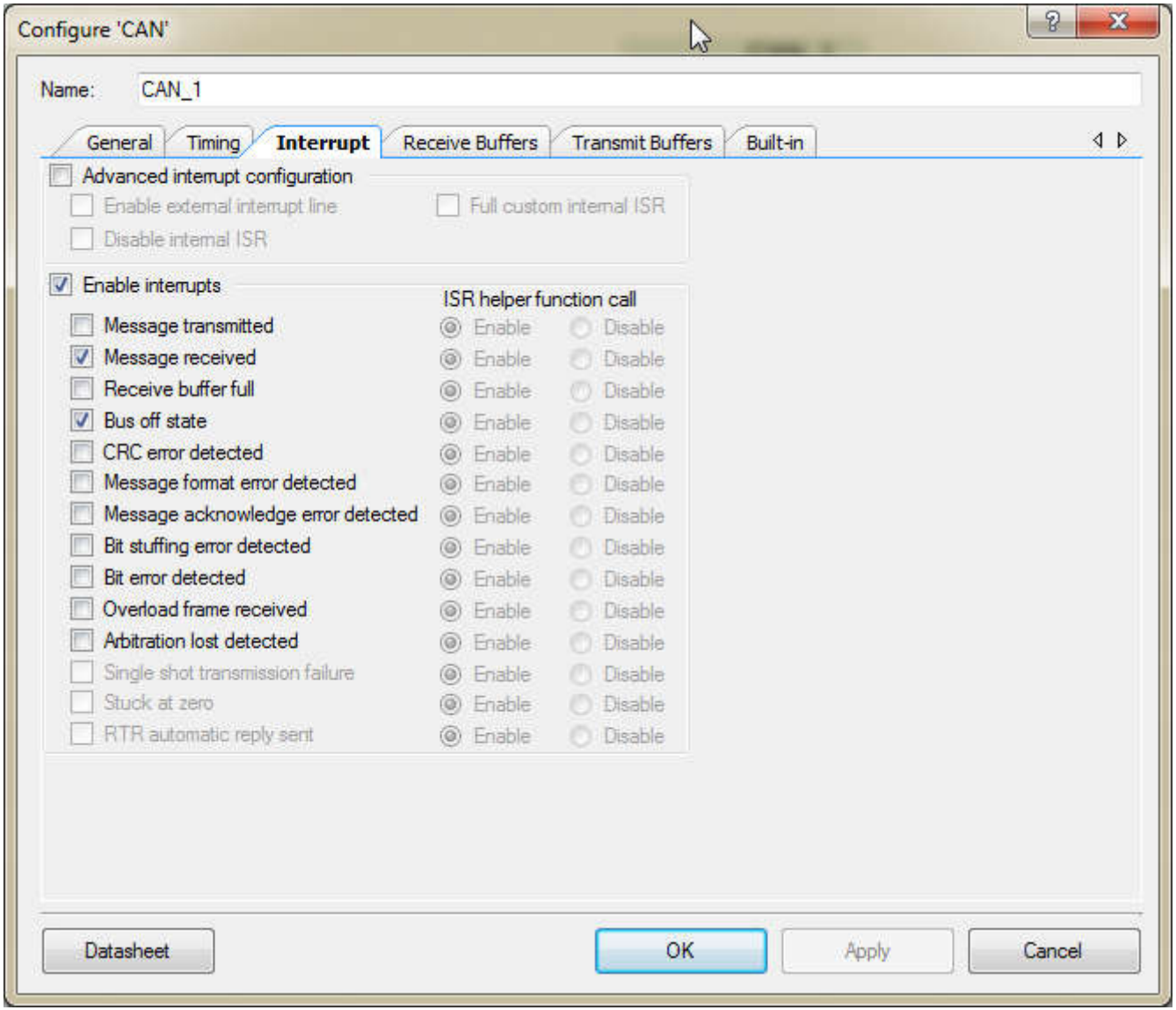
Konfiguracja bloku CAN – jak na kolejnych slajdach



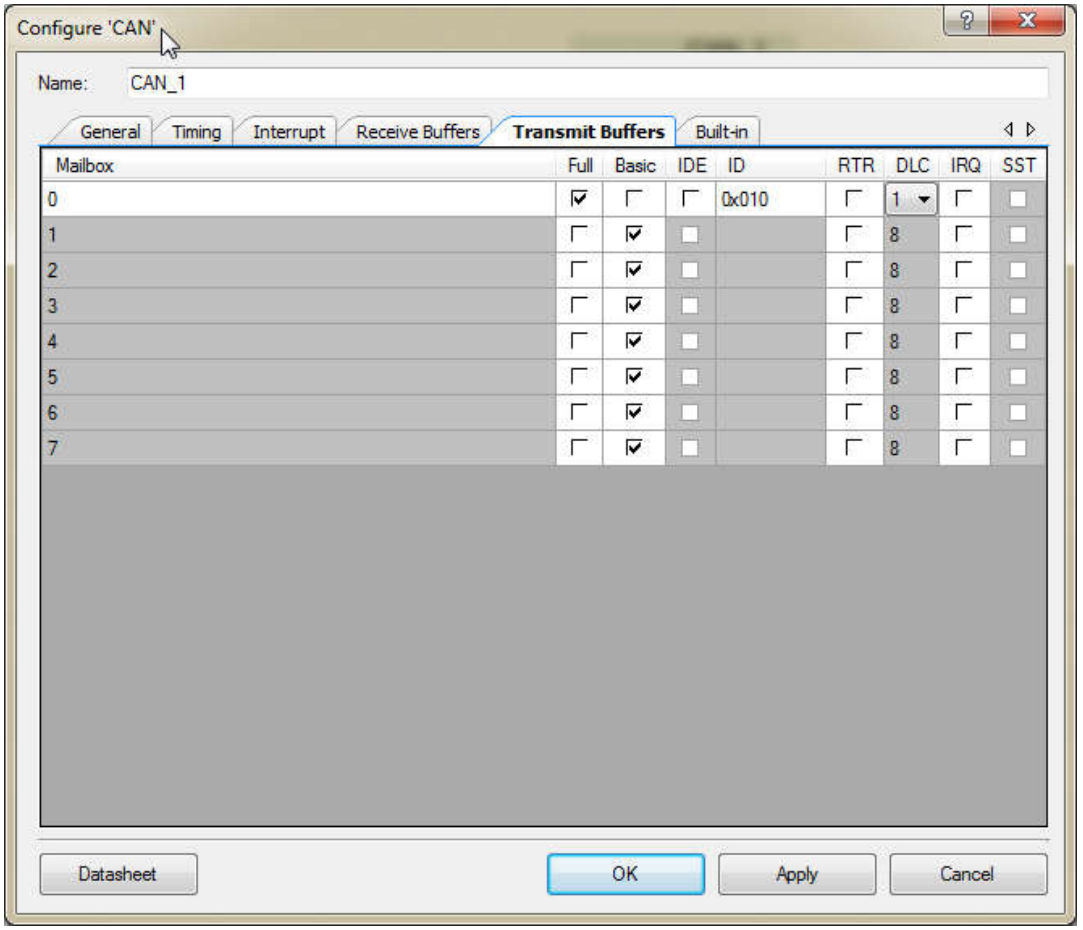
Po wybraniu prędkości 250kps należy kliknąć podwójnie na linię zaznaczoną poniżej strzałką a następnie zmienić wartość SJW na 1. Szczegółowe informacje dotyczące parametrów czasowych można znaleźć w dokumentacji bloku CAN.






Configure 'CAN'
Name:
CAN_1
General
Timing
Interrupt
Receive Buffers
Transmit Buffers
Built-in
Advanced interrupt configuration
Enable external interrupt line
Full custom internal ISR
Disable internal ISR
Enable interrupts
ISR helper function call
Message transmitted
Message received
Receive buffer full
Bus off state
CRC error detected
Message format error detected
Message acknowledge error detected
Bit stuffing error detected
Bit error detected
Overload frame received
Arbitration lost detected
Single shot transmission failure
Stuck at zero
RTR automatic reply sent
Enable
Disable
Datasheet
OK
Apply
Cancel

Zakładkę **Recevie Buffers** pomijamy a w **Transmit Buffers** wypełniamy jak poniżej. **DLC** oznacza liczbę bajtów wiadomości, **ID** oznacza identyfikator wiadomości.



Projekt „Nadajnik” - oprogramowanie

- Na początek należy wygenerować biblioteki (Shift-F6).
- Następnie należy uzupełnić plik „main.c” do postaci jak poniżej:

```
#include "project.h"
uint8 Tx_Data = 0;
int main(void)
{
    CyGlobalIntEnable; /* Enable global interrupts. */
    /* Place your initialization/startup code here (e.g. MyInst_Start()) */
    CAN_1_Start();
    LCD_Char_1_Start();
    CyGlobalIntEnable;

    for(;;)
    {
        /* Place your application code here. */
        LCD_Char_1_ClearDisplay();
        LCD_Char_1_Position(0,0);
        LCD_Char_1_PrintNumber(Tx_Data);
        CAN_1_SendMsg0();
        Tx_Data++;
        CyDelay(500);
    }
}
```

Projekt „Nadajnik” – oprogramowanie c.d.

• Należy otworzyć plik „CAN_1_TX_RX_func.c”, odnaleźć fragmenty i uzupełnić jak pokazano poniżej:

```
/* `#START TX_RX_FUNCTION` */

extern uint8 Tx_Data; // <= linia dodana

/* `#END` */
```

oraz w funkcji `uint8 CAN_1_SendMsg0(void)`:

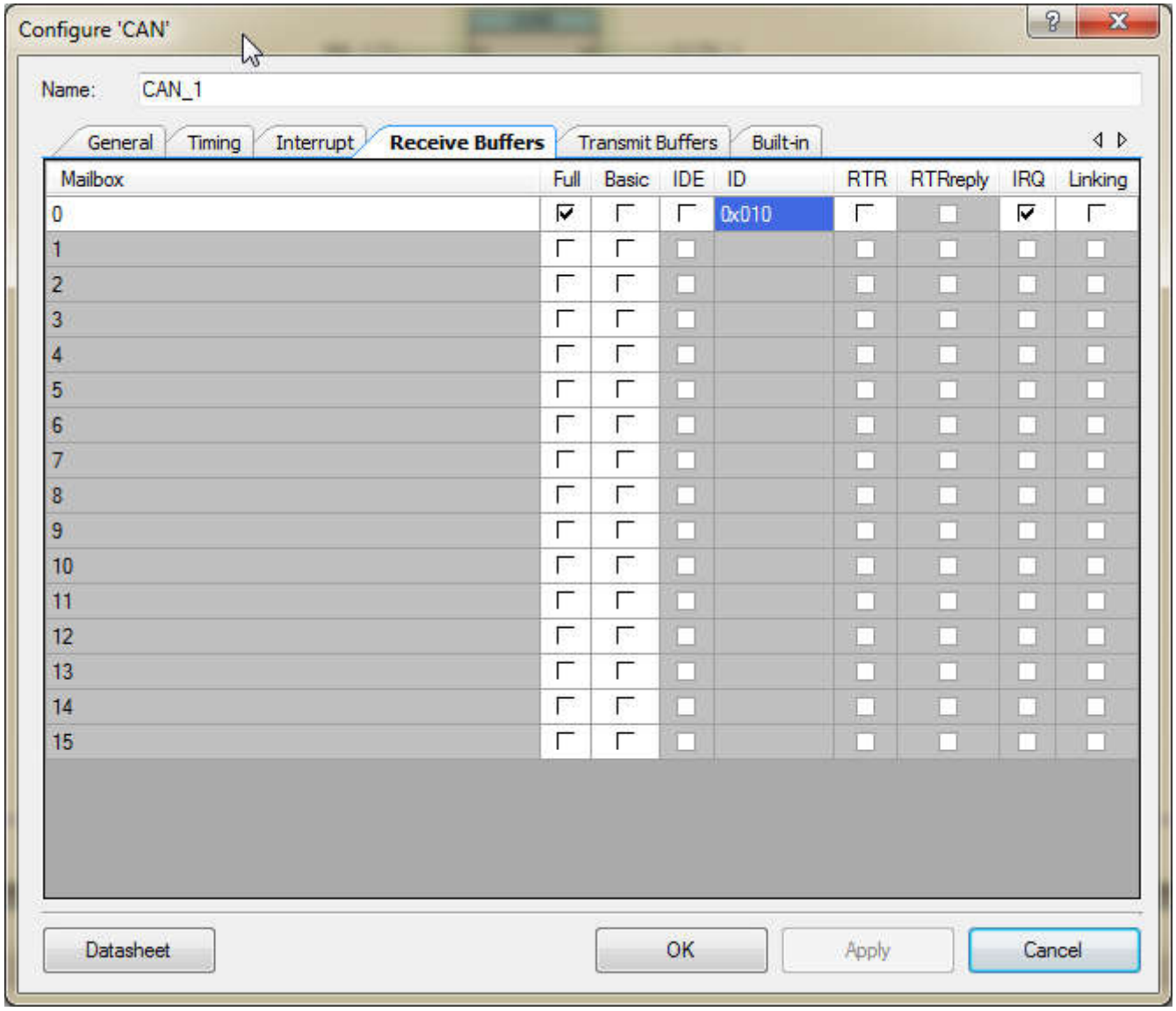
```
/* `#START MESSAGE_0_TRASMITTED` */

CAN_1_TX_DATA_BYTE1(0) = Tx_Data; // <= linia dodana

/* `#END` */
```

Projekt „Odbiornik” – sprzęt

- Schemat identyczny jak „Nadajnik”.
- Konfiguracja zegarów identyczna.
- Konfiguracja bloku CAN identyczna jak w projekcie „Nadajnik” z tą różnicą, że zakładka „Transmit Buffers” zostaje nienaruszona a zakładka „Receive Buffers” jak na rys. na kolejnym slajdzie.


Configure 'CAN'
Name: CAN_1
General
Timing
Interrupt
Receive Buffers
Transmit Buffers
Built-in
Mailbox
Full
Basic
IDE
ID
RTR
RTRreply
IRQ
Linking
0
0x010
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Datasheet
OK
Apply
Cancel

Projekt „Odbiornik” - oprogramowanie

- Na początek należy wygenerować biblioteki (Shift-F6).
- Następnie należy uzupełnić plik „main.c” do postaci jak poniżej:

```
#include "project.h"
uint8 Rx_Data, Rx_Data_Old;
int main(void)
{
    CyGlobalIntEnable; /* Enable global interrupts. */
    CAN_1_Start();
    LCD_Char_1_Start();
    CyGlobalIntEnable;
    /* Place your initialization/startup code here (e.g. MyInst_Start()) */

    for(;;)
    {
        /* Place your application code here. */
        //LCD_Char_1_ClearDisplay();
        //LCD_Char_1_Position(0,0);
        if (Rx_Data != Rx_Data_Old)
        {
            Rx_Data_Old = Rx_Data;
            LCD_Char_1_ClearDisplay();
            LCD_Char_1_Position(0,0);
            LCD_Char_1_PrintNumber(Rx_Data);
        }
        //LCD_Char_1_PrintNumber(Rx_Data);
        //CyDelay(100);
    }
}
```

Projekt „Odbiornik” – oprogramowanie c.d.

• Należy otworzyć plik „CAN_1_TX_RX_func.c”, odnaleźć fragmenty i uzupełnić jak pokazano poniżej:

```
/* `#START TX_RX_FUNCTION` */

extern uint8 Rx_Data; // <= linia dodana

/* `#END` */
```

oraz w funkcji `void CAN_1_ReceiveMsg0(void):`

```
/* `#START MESSAGE_0_RECEIVED` */

Rx_Data = CAN_1_RX_DATA_BYTE1(0); // <= linia dodana

/* `#END` */
```

Testowanie

- Należy wgrać projekt „Nadajnik” na jedną z płytek a „Odbiornik” na drugą.
- Na obu płytkach powinny na wyświetlaczu pojawiać się kolejno nadawane liczby.