



WELCOME

Warsztaty szkoleniowe

Technologia SafetyLon w systemach związanych z bezpieczeństwem funkcjonalnym

Demo funkcji bezpiecznych

Moduł 7.1



Plan prezentacji



1.	Założenia programu demo SFDA – Safe Function Demo Application
2.	Interfejs sieciowy i wejścia/wyjścia obiektowe
3.	Algorytm aplikacji
4.	Połączenia sieciowe i funkcjonalność
5.	Funkcje pomocnicze wejścia/wyjścia



```

#pragma enable_sd_nv_names
#pragma set_node_sd_string "Safe Function Demo App"

msg_tag bind_info(nonbind) sage_mgn;

#pragma set_std_prog_id 81:02:03:05:28:0A:04:08

// INPUT NVs
// Non-safe input NVs
network input sd_string("NonSafe in data 1") SNVT_switch nviData;

// Safe input NVs
//##SL SNVT_switch
network input sd_string("Safe output 1") UNVT_safe_2 nvis_D01;
//##SL SNVT_switch
network input sd_string("Safe output 2") UNVT_safe_2 nvis_D02;

// OUTPUT NVs
// Non-safe output NVs
network output sd_string("NonSafe out data 1") SNVT_switch nvoData;
network output sd_string("SOS+App Version info") SNVT_str_asc nvoAppVer;

// Safe output NVs
//##SL SNVT_switch,1500,100,250
network output sd_string("safe input 1") UNVT_safe_2 nvos_DI1;
//##SL SNVT_switch,1500,100,250
network output sd_string("Safe input 2") UNVT_safe_2 nvos_DI2;

//##SL UNVT_timesync
network output sd_string("Safe out NV time sync") UNVT_safe_4 nvos_Time;
//##SL UNVT_timesync,5000,200,1000
network input sd_string("Safe out NV time sync") UNVT_safe_4 nvis_Time;

```

zmienne standardowe (nie bezpieczne)

zmienne bezpieczne

słowo kluczowe (//##SL)
nazwa SNVT,
max. heartbeat,
min heartbeat,
domyślny heartbeat

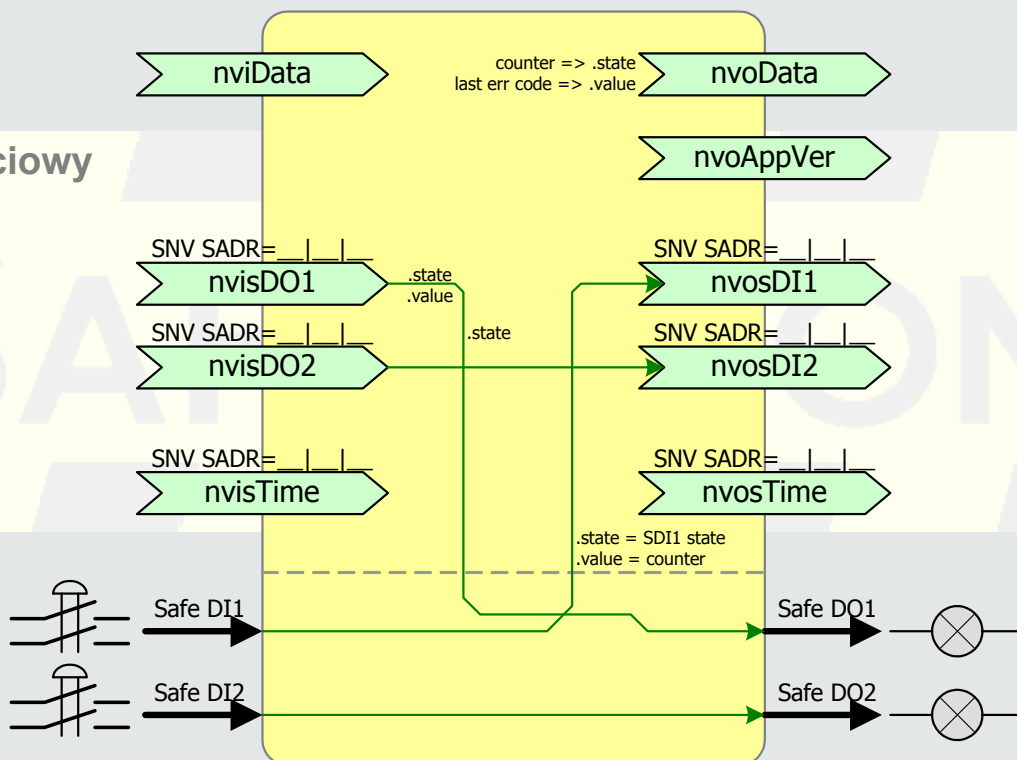


SFDA - wstęp



Interfejs sieciowy

interfejs sprzętowy



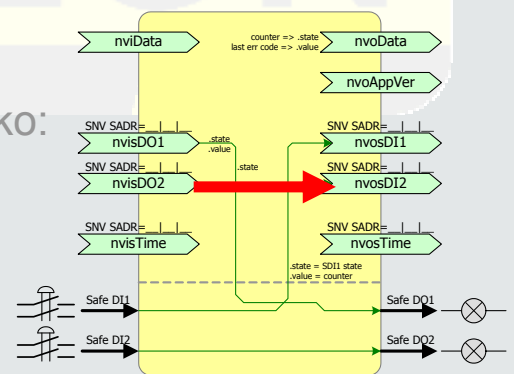


Cele SFDA (1)



– SFDA demonstruje:

- zm. bezp. wejściowa → zm. bez. wyjściowa
 - nvis_DI2 → nvos_DO2
- bezp. wejście → bezp. wyjście
 - Safe DI2 → Safe DO2
- zm. bezp. wejściowa → bezp. wyjście
 - nvis_DI1 → Safe DO1
- bezp. wejście → zm. bezp. wyjściowa
 - Safe DI1 → nvos_DI1
- stany bezpieczne – przerwanie kanału komunikacji
- bezpieczne wejścia mogą być używane jako:
 - przycisk o pojedynczym styku
 - przycisk o podwójnym styku (SIL-3)
 - » z lub bez filtrowania rozbieżności



Slajd nr 5

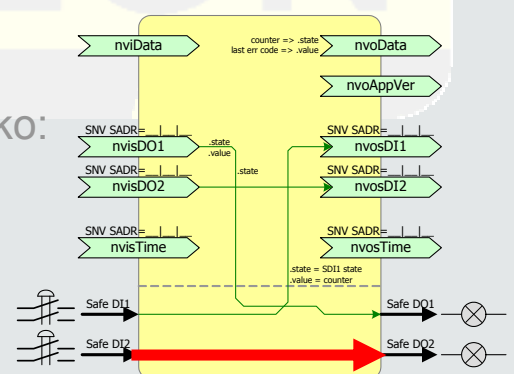


Cele SFDA (2)



– SFDA demonstruje:

- zm. bezp. wejściowa → zm. bez. wyjściowa
 - nvis_DI2 → nvos_DO2
- bezp. wejście → bezp. wyjście
 - Safe DI2 → Safe DO2
- zm. bezp. wejściowa → bezp. wyjście
 - nvis_DI1 → Safe DO1
- bezp. wejście → zm. bezp. wyjściowa
 - Safe DI1 → nvos_DI1
- stany bezpieczne – przerwanie kanału komunikacji
- bezpieczne wejścia mogą być używane jako:
 - przycisk o pojedynczym styku
 - przycisk o podwójnym styku (SIL-3)
 - » z lub bez filtrowania rozbieżności



Slajd nr 6

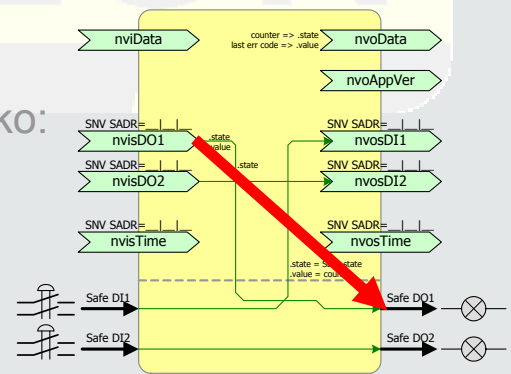


Cele SFDA (3)



– SFDA demonstruje:

- zm. bezp. wejściowa → zm. bez. wyjściowa
 - nvis_DI2 → nvos_DO2
- bezp. wejście → bezp. wyjście
 - Safe DI2 → Safe DO2
- **zm. bezp. wejściowa → bezp. wyjście**
 - **nvis_DI1 → Safe DO1**
- bezp. wejście → zm. bezp. wyjściowa
 - Safe DI1 → nvos_DI1
- stany bezpieczne – przerwanie kanału komunikacji
- bezpieczne wejścia mogą być używane jako:
 - przycisk o pojedynczym styku
 - przycisk o podwójnym styku (SIL-3)
 - » z lub bez filtrowania rozbieżności

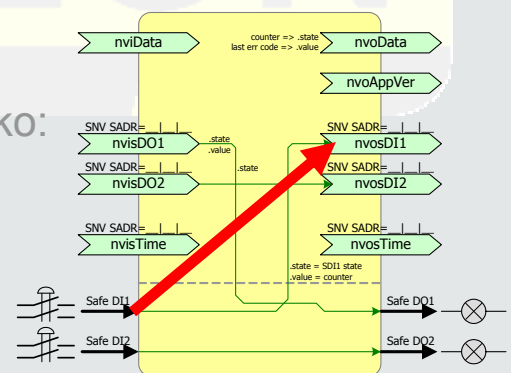


Cele SFDA (4)



– SFDA demonstruje:

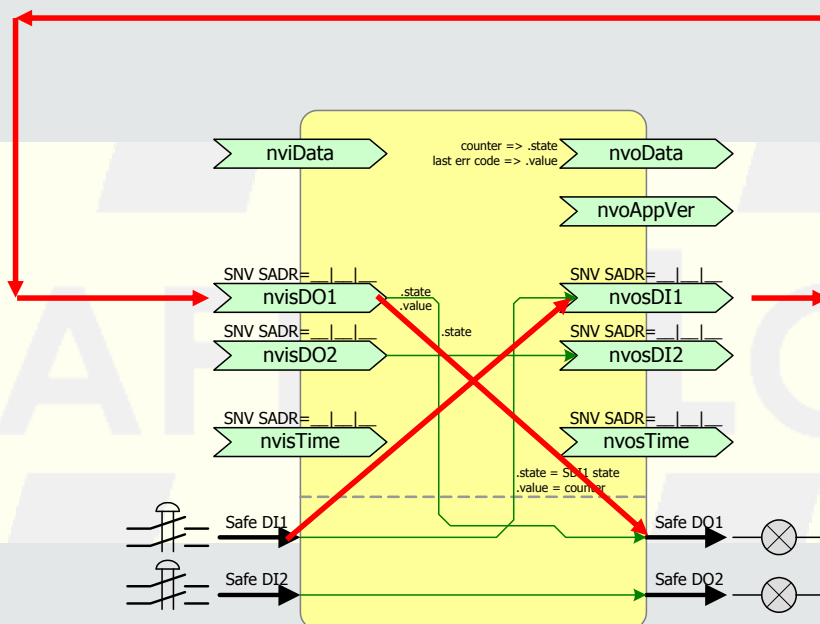
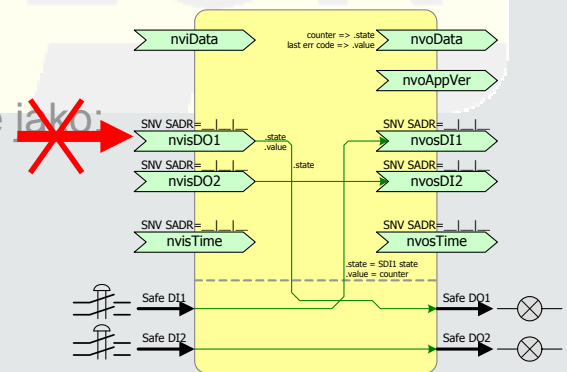
- zm. bezp. wejściowa → zm. bez. wyjściowa
 - nvis_DI2 → nvos_DO2
- bezp. wejście → bezp. wyjście
 - Safe DI2 → Safe DO2
- zm. bezp. wejściowa → bezp. wyjście
 - nvis_DI1 → Safe DO1
- **bezp. wejście → zm. bezp. wyjściowa**
 - **Safe DI1 → nvos_DI1**
- stany bezpieczne – przerwanie kanału komunikacji
- bezpieczne wejścia mogą być używane jako:
 - przycisk o pojedynczym styku
 - przycisk o podwójnym styku (SIL-3)
 - » z lub bez filtrowania rozbieżności





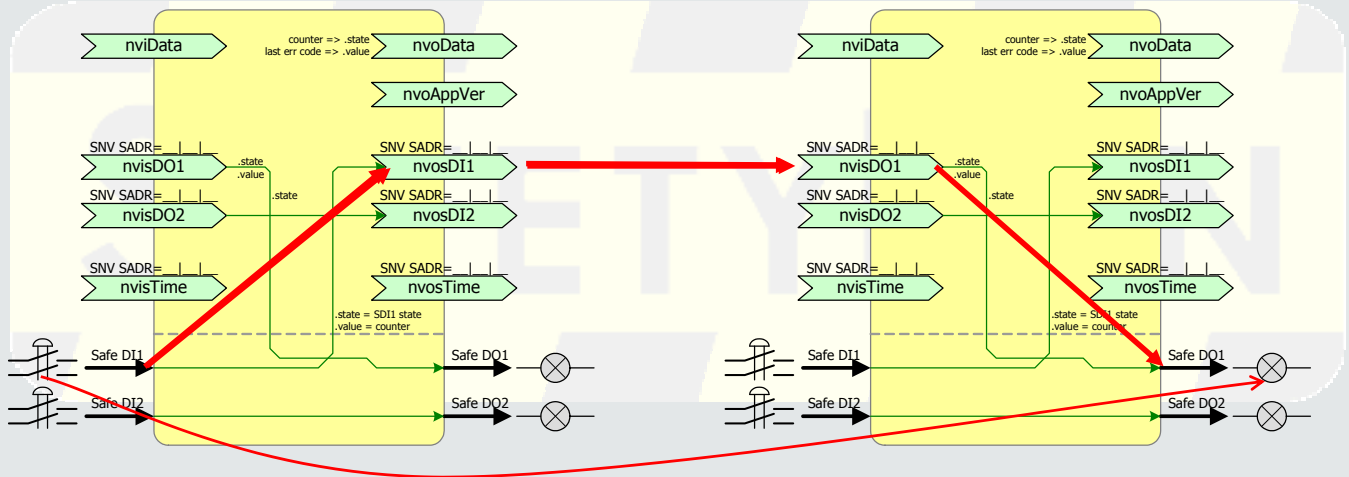
– SFDA demonstruje:

- zm. bezp. wejściowa → zm. bez. wyjściowa
 - nvis_DI2 → nvos_DO2
- bezp. wejście → bezp. wyjście
 - Safe DI2 → Safe DO2
- zm. bezp. wejściowa → bezp. wyjście
 - nvis_DI1 → Safe DO1
- bezp. wejście → zm. bezp. wyjściowa
 - Safe DI1 → nvos_DI1
- **stany bezpieczne – przerwanie kanału komunikacji**
- bezpieczne wejścia mogą być używane jako:
 - przycisk o pojedynczym styku
 - przycisk o podwójnym styku (SIL-3)
 - » z lub bez filtrowania rozbieżności

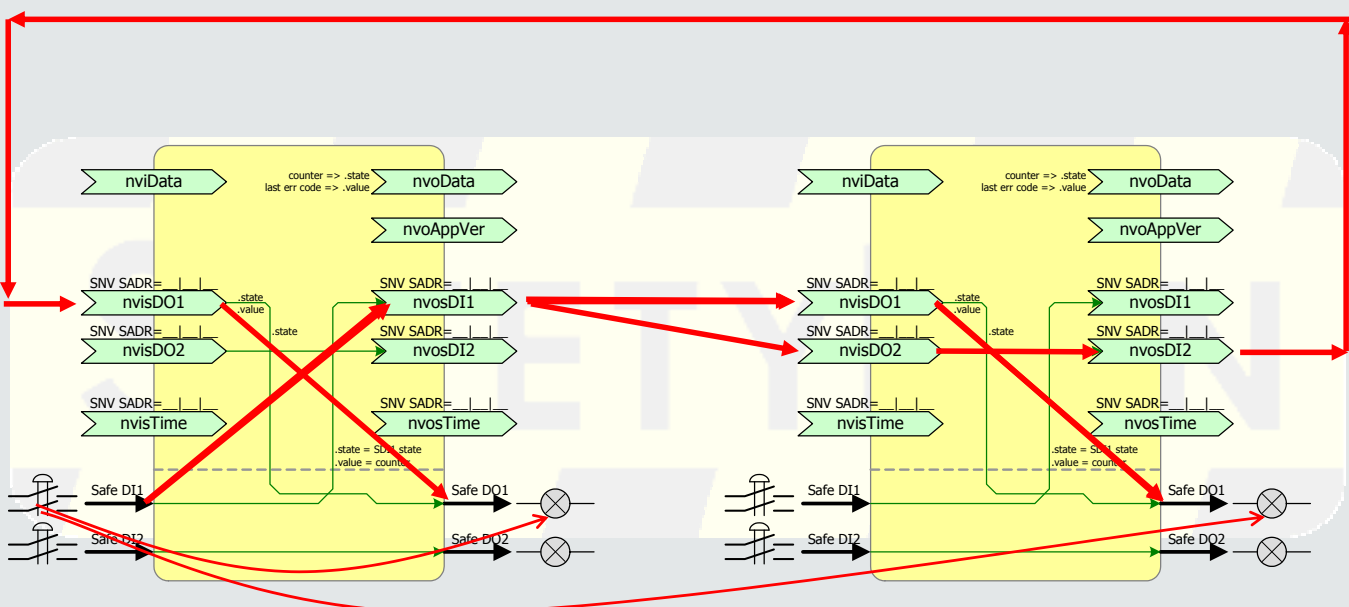




Możliwe połączenia sieciowe (2)



Możliwe połączenia sieciowe (3)





- samo restartujący timer
 - timer_reset(): (re)startuje timer
 - timer_expired()
 - jeśli timer odliczył, restartuje go i zwraca 1
 - w przeciwnym wypadku zwraca 0

funkcje SOS

kod aplikacji

```
clock_sam7t lTriggerLast[MAX_TIMER];

// timer implementation
void timer_reset( int nTimer ){
    if( nTimer<0 || nTimer>=MAX_TIMER ) // check if nTimer is in range
        return;
    lTriggerLast[nTimer] = sapi_get_time(); // remember time when timer is reset
}

int timer_expired( int nTimer, long msec ){
    clock_sam7t period = msec/10; // sapi_get_time returns time with 10ms
    resolution
    clock_sam7t now = sapi_get_time();
    if( nTimer<0 || nTimer>=MAX_TIMER )
        return 0;
    if( now>=(lTriggerLast[nTimer]+period) ){ // check if timer expired
        lTriggerLast[nTimer] = now; // reset timer (it's self-resetting timer)
        return 1; // return "timer expired and resetted"
    }
    return 0; // return "timer still counting"
}
}
```



- Wejścia bezpieczne:
 - odczyt wejścia o pojedynczym styku (get_sinuput_1)
 - odczyt wejścia z podwójnych stykach (SIL3)
 - bez eliminacji rozbieżności (get_sinuput)
 - z eliminacją rozbieżności (get_sinuput_filter)
- Wyjścia bezpieczne
 - ustawianie stanu bezpiecznego wyjścia (set_soutput)

```
// Sets safe output iOutNum to Value
void set_soutput( int iOutNum, int value ){
    if( iOutNum<0 || iOutNum>1 ) // check if iOutNum is in right range
        return;
    if( value!=0 ){
        sapi_output_set( iOutNum );
    } else {
        sapi_output_clear( iOutNum );
    }
}
}
```



Funkcje pomocnicze – wejścia bezpieczne (1)



```
int get_sinput( int iSInNum ){
    int in0, in1;
    if( iSInNum<0 || iSInNum>1 ) // check if iSInNum is in valid range
        return 0x10 | 4; // return error code
    in0 = sapi_input_get( 2*iSInNum );
    in1 = sapi_input_get( 2*iSInNum+1 );
    if( in0!=in1 ){
        return 0x10 | ( in1 << 1 ) | in0;
    }
    return in0;
}

int get_sinput_1( int iSInNum ){
    if( iSInNum<0 || iSInNum>1 ) // check if iSInNum is in valid range
        return 0x10 | 4; // return error code
    return sapi_input_get( 2*iSInNum );
}
```



Funkcje pomocnicze – wejścia bezpieczne (2)



```
struct {
    bool bDiscrepancy;
    int iLastState;
} SIN[2];

int get_sinput_filter( int iSInNum ){
    int in0, in1; // the variables that hold safe input states
    if( iSInNum<0 || iSInNum>1 ){ // check if iSInNum is in valid range
        return 0x10 | 4; // return error code
    }
    in0 = sapi_input_get( 2*iSInNum );
    in1 = sapi_input_get( 2*iSInNum+1 );
    if( in0==in1 ){ // no discrepancy
        SIN[iSInNum].bDiscrepancy = false; // there's no discrepancy
        SIN[iSInNum].iLastState = in0; // value of safe inputs is = in0 = in1
        return in0;
    }
    // else - we have discrepancy
    if( SIN[iSInNum].bDiscrepancy==false ){ // its discovered for the first time
        timer_reset( iSInNum ); // start discrepancy timer
        SIN[iSInNum].bDiscrepancy=true; // timer started
        return SIN[iSInNum].iLastState; // return "last good input state"
    } else {
        if( timer_expired( iSInNum, SIN_DISCREPANCY_TIME ) ){
            SIN[iSInNum].iLastState = 0x10 | ( in1 << 1 ) | in0; // "last state"
            // indicates discrepancy
        }
    }
    return SIN[iSInNum].iLastState; // return "last good input state"
}
```



- Inicjalizacja aplikacji:
 - wartości zmiennych bezpiecznych inicjalizowane na „wartości bezpieczne”
 - inicjalizacja zmiennych globalnych do stanów bezpiecznych

```
////////////////////////////////////  
//  
// called before first application() cycle  
// initializes all nvo, nvos and global variables  
  
void application_init( void ){  
    memset( &sapi_nvos_DI1,      0, sizeof( sapi_nvos_DI1 ) );  
    memset( &sapi_nvos_DI2,      0, sizeof( sapi_nvos_DI2 ) );  
    g_nvisD01_SS = 0;  
    g_nvisD02_SS = 0;  
}
```



- Algorytm aplikacji zawiera 7 kroków:
 - jeśli nvis_DO1 jest w stanie bezpiecznym, DO1 mruga – priorytet nad innymi sterowaniami
 - jeśli nvis_DO2 jest w stanie bezpiecznym, DO2 mruga – priorytet nad innymi sterowaniami
 - zmiany DI1 są wysyłane na zmiennej nvos_DI1, change_counter jest zwiększany przy każdej zmianie DI1
 - DI2 steruje DO2, zmiana DI2 zeruje change_counter,
 - nvis_DO1.state steruje DO1
 - aktualizacje nvis_DO2 są przepisywane do nvos_DI2
 - nvoData.state = change_counter
nvoData.value = ostatni kod błędu – błąd wejść bezpiecznych (rozbieżność)
nviData.value = wybór trybu odczytu wejścia bezpiecznego (0/0,5/1,0/1,5 = get_sinput/get_sinput_1/get_sinput_filter/błąd aplikacji)



- sapi_nvi_safe_state sprawdza spodziewany czas odbioru heartbeatów
 - demonstracja zerwania komunikacji na kanale standardowym (nie bezpiecznym)

```
if( sapi_nvi_safe_state( SAPI_NVIDX_NVIS_DO1 ) ){
    g_nvisDO1_SS = 1; // safe state was detected and safe output blinks
    if( timer_expired( TIdx_DI1_SS, SS_BLINK_TIME ) )
        g_nvisDO1_SS_toggle = (g_nvisDO1_SS_toggle==1) ? 0 : 1;
    set_soutput( 0, g_nvisDO1_SS_toggle );
} else {
    if( g_nvisDO1_SS != 0 ){ // previously we've blinked the outputs, so "repair it"
        set_soutput( 0, sapi_nvos_DI1.state==1 ); // safe output 0 = nvis_DO1.state
        g_nvisDO1_SS = 0; // safe state of nvis_DO1 has ended
    }
}
```



Wejście bezpieczne → zmienna sieciowa



- odczyt stanu wejścia bezpiecznego
- sprawdzenie, czy stan wejścia uległ zmianie
- zapis wartości wejścia do wyjściowej zmiennej bezpiecznej
 - sapi_nvo_data_write()
 - sapi_nvo_propagate()

```
DI1 = get_sinput( 0 ); // get_sinput_1( 0 ) or get_sinput_filter( 0 )
if( DI1!=prev_DI1 ){ // this is new value of 1st safe input
    prev_DI1 = DI1; // remember this value for further processing
    // update and propagate nvos_DI1 value
    sapi_nvos_DI1.state = DI1;
    sapi_nvos_DI1.value = cntr_DI1; // value shows how many times safe DI1 was triggered
    sapi_nvo_data_write( SAPI_NVIDX_NVOS_DI1, &sapi_nvos_DI1, sizeof( sapi_nvos_DI1 ) );
    sapi_nvo_propagate( SAPI_NVIDX_NVOS_DI1 );
}
```



Wejście bezp. → wyjście bezp.



- odczyt stanu wejścia bezpiecznego
- sprawdzenie, czy stan wejścia uległ zmianie
- zapis stanu wejścia bezpiecznego na wyjście bezpieczne

```
DI1 = get_sinput( 1 ); // get_sinput_1( 1 ) or get_sinput_filter( 1 )
if( DI1!=prev_DI2 ){ // this is new value of safe input 2
    prev_DI2 = DI1;
    cntr_DI1 = 0; // reset change_counter
    // if there's Safe State of nvis_D02, then Safe State drives 1st safe digital output
    // Otherwise we can control the safe output
    if( g_nvisD02_SS==0 )
        set_soutput( 1, DI1==1 );
}
```



Zmienna wej. → zmienna wyj.



- sprawdzenie czy pojawiła się aktualizacja
 - odczyt zmiennej
 - sprawdzenie czy długość jest poprawna
- jeśli powyższe ok., sterowanie wyjściem bezpiecznym

```
if( sapi_nvi_update_received( SAPI_NVIDX_NVIS_D01 ) ){
    sapi_nv_data_read( SAPI_NVIDX_NVIS_D01, &sapi_nvis_D01, &len );
    if( len!=sizeof(sapi_nvis_D01) ){
        // signal error condition
    } else {
        // if nvis_D01 update is properly received ...
        if( g_nvisD01_SS==0 ) // ... and there's no safe state of this nvis ...
            set_soutput( 0, sapi_nvis_D01.state==1 ); // ... then set 1st safe output
    }
}
```



- sprawdzenie czy pojawiła się aktualizacja
 - odczyt zmiennej
 - sprawdzenie czy długość jest poprawna
- jeśli powyższe ok., zapis do wyjściowej zmiennej bezpiecznej

```
if( sapi_nvi_update_received( SAPI_NVIDX_NVIS_D02 ) ){
    sapi_nv_data_read( SAPI_NVIDX_NVIS_D02, &sapi_nvis_D02, &len );
    if( len!=sizeof(sapi_nvis_D02) ){
        // signal error condition
    } else {
        // if nvis_D02 update is properly received ...
        // ... then copy the nvis_D02 to nvos_DI2 and write/propagate
        memcpy( &sapi_nvos_DI2, &sapi_nvis_D02, sizeof( sapi_nvos_DI2 ) );
        sapi_nvo_data_write( SAPI_NVIDX_NVOS_DI2, &sapi_nvos_DI2, sizeof( sapi_nvos_DI2 ) );
        sapi_nvo_propagate( SAPI_NVIDX_NVOS_DI2 );
    }
}
```



5.	Wprowadzenie do środowiska projektowego
5.1.	Metodyka prac przy projektowaniu aplikacji
5.2.	Środowisko szkoleniowe
P2	Przerwa 2
5.3.	Interfejs programowy aplikacji (API)
6.	Integracja sieci SafetyLon
7.	Przykładowe aplikacje
7.1.	Demo funkcji bezpiecznych
7.2.	Obsługa przycisku awaryjnego
7.3.	Element logiczny
8.	Wytyczne dot. zarządzania bezpieczeństwem funkcjonalnym

