INSTYTUT CYBERNETYKI TECHNICZNEJ
POLITECHNIKI WROCŁAWSKIEJ
Raport serii SPR nr 11/2004

Modyfikacja robota mobilnego MK

Robert Szlawski
Marek Wnuk

Słowa kluczowe: robot mobilny, mikrokontroler, sterownik

Wrocław 2004

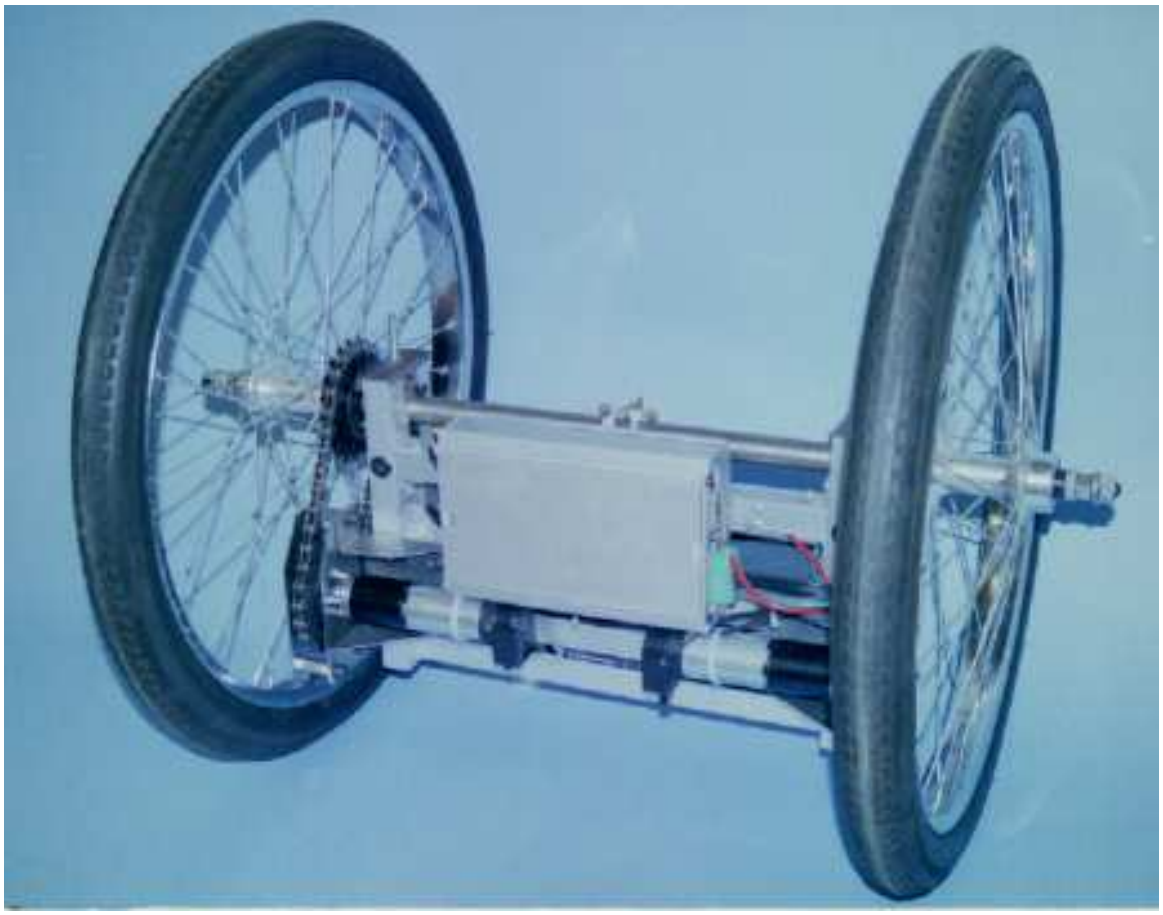
Spis treści

Spis rysunków

Spis tablic

1 Wprowadzenie

Robot mobilny MK [4] jest dwukołowym wózkiem pozwalającym badać elementy napędu i sterowania obiektów nieholonomicznych. Stanowi nie tylko model doświadczalny poprzedzający wykonanie robota kulistego RoBall [7], ale również samodzielne stanowisko do badania algorytmów sterowania obiektem nieholonomicznym z uwarunkowaniami dynamicznymi (przekazywanie napędu z wykorzystaniem grawitacji wymaga uwzględniania parametrów dynamicznych robota przy rozwiązywaniu zadania nawigacji [3]).



Rysunek 1: Widok ogólny modelu wózka

Został on z powodzeniem wykorzystany w pracach badawczych związanych z uruchamianiem i badaniem algorytmów sterowania opartych na linearyzacji dynamicznej [5, 8] i statycznej [9], zaimplementowanych na rzeczywistym obiekcie. Wyniki tych prac wykazały, że głównym utrudnieniem przy implementaowaniu algorytmów sterowania opartych na modelu dynamiki wózka MK jest brak sprzętowego układu zmiennoprzecinkowego w mikrokontrolerze MC68332. Poniżej przedstawiono modernizację sterownika wózka MK rozwiązującą ten problem.

2 Modernizacja sprzętu sterownika wózka MK

Sterownik wózka MK zmontowano na płytce o wymiarach 160x100mm i wbudowano do typowej kasetki 3U/6TE, która jest zainstalowana w prowadnicach stanowiących mechaniczne połączenie z korpusem [4]. Wyglad oryginalnego sterownika w kasetce przedstawiono na rys. 2. Jednostka centralna sterownika, moduł EM332 [10], jest umieszczona w dwóch dwurzędowych złączach na płytce uniwersalnej. Sygnały modułu EM332 wykorzystywane w sterowniku wózka MK podano w tab. 1.

Rysunek 2: Widok ogólny płytki sterownika oryginalnego w kasetce

Opisywana modernizacja sprzętu sterownika polega na zamianie jednostki MC68332 na jednostkę MPC555. W tym celu wykorzystano moduł phyCORE–MPC555 firmy PHYTEC [2]. Dostosowanie wyprowadzeń modułu phyCORE do istniejących łączówek dla modułu EM332 [10] wymagało wykonania płytki przejściowej (rys. 3). Widok sterownika po modernizacji przedstawiono na rys. 4.
Schemat płytki przejściowej pokazano na rys. 5, a przyporządkowanie sygnałów modułu phyCORE wykorzystywanym w sterowniku sygnałom modułu EM332 w tab. 4. Płytka przejściowa, oprócz odpowiednich łączówek dla modułu phyCORE i wtyków symulujących moduł EM332, zawiera stabilizator napięcia 3.3V, gniazdo interfejsu BDM, zworki do ustalania trybu pracy modułu phyCORE i konfigurowania sygnałów interfejsów zewnętrznych (w szczególności - TPU). Została ona wykonana w postaci dwustronnego obwodu drukowanego z metalizowanymi otworami przelotowymi. Rozmieszczenie elementów na obu stronach płytki pokazano na rys. 6, a wygląd mozaiki połączeń na rys. 7.
Zworka JP4 zezwala na programowanie wewnętrznej pamięci FLASH mikrokontrolera MPC555 (jest przyłączona do sygnału EPEE). Zworka JP15 pozwala wybrać pamięć, z której moduł phyCORE startuje (wewnętrzna lub zewnętrzna pamięć FLASH). Jest połączona z linią danych D20, która

Rysunek 3: Widok obu stron płytki przejściowej sterownika

Rysunek 4: Widok sterownika po modernizacji

w czasie restartu steruje bitem FLEN w rejestrze konfiguracyjnym mikrokontrolera (*Hard-Reset-*

złącze L				złącze P			
	–	B1	VCC	VCC	C1	D1	GND
GND	A2	B2	RESET	RxD	C2	D2	TxD
	A3	B3		PCS3	C3	D3	PCS2
	A4	B4		PCS1	C4	D4	PCS0
	A5	B5		SCK	C5	D5	MOSI
	A6	B6		MISO	C6	D6	
	A7	B7			C7	D7	
PE6	A8	B8	PE7		C8	D8	
PE5	A9	B9	PE4		C9	D9	
PE2	A10	B10	PE3		C10	D10	
PE1	A11	B11	PE0		C11	D11	
	A12	B12			C12	D12	
	A13	B13			C13	D13	
	A14	B14			C14	D14	
	A15	B15			C15	D15	
	A16	B16		TP1	C16	D16	TP0
	A17	B17		TP3	C17	D17	TP2
	A18	B18		TP5	C18	D18	TP4
	A19	B19		TP7	C19	D19	TP6
	A20	B20		TP9	C20	D20	TP8
	A21	B21		TP11	C21	D21	TP10
	A22	B22		TP13	C22	D22	TP12
	A23	B23	T2CLK	TP15	C23	D23	TP14
GND	A24	B24	VCC	VCC	C24	D24	GND

Tablica 1: Wykorzystywane sygnały modułu EM332

zworka	stan	domyślnie	opis
JP4	rozwarta	X	zakaz programowania wewnętrznej pamięci FLASH, zasilanie FLASH obniżone do 3.3V
	zwarta		zezwolenie na programowanie wewnętrznej pamięci FLASH, zasilanie FLASH - 5V
JP15	rozwarta		wybór zależy od ustawienia J1 na phyCORE ([2])
	1-2		start z wewnętrznej pamięci FLASH
	2-3	X	start z zewnętrznej pamięci FLASH
JP16	rozwarta	X	źródło konfiguracyjnego ustawione przez J5 na phyCORE ([2])
	1-2		konfiguracja z ustawień wewnętrznych (domyślnych)
	2-3		konfiguracja ze źródła zewnętrznego

Tablica 2: Zworki konfiguracyjne dla phyCORE MPC555

styk	sygnał	opis
1	RXD2	dane odbierane RS232C, SCI2
2	TXD2	dane nadawane RS232C, SCI2
3	GND	masa

Tablica 3: Wyprowadzenie dodatkowego portu szeregowego (złącze RS232)

Configuration-Word). Zworka JP16 steruje wyborem źródła rejestru konfiguracyjnego (wewnętrzne - ustawienia domyślne, lub zewnętrzne). Możliwości ustawienia zworek i ustawienie domyślne podano w tab. 2.
Zworki lutowane TP[15:0] i T2CLK służą do wyboru bloku TPU obsługującego poszczególne kanały.

sygnał EM332	typ	opis	sygnał MPC555
RxD	I	dane odbierane SCI	RXD1_TTL
TxD	IO	dane nadawane SCI	TXD1_TTL
PCS[3:0]	IO	wybór urządzenia QSPI	QGPIO[3:0]
SCK	IO	zegar QSPI	QGPIO6
MOSI	IO	we/wy danych QSPI (*Master Out Slave In*)	QGPIO5
MISO	IO	we/wy danych QSPI (*Master In Slave Out*)	QGPIO4
TP[15:2]	IO	sygnały kanałów TPU	A/B_TPU[15:2]
TP[1:0]	IO	sygnały kanałów TPU	A/B_TPU[1:0] / MPWMSM[1:0]
T2CLK	I	wejście zegara zewnętrznego TPU	A/B_T2CLK
RESET	IO	restart	HRESIN
PE[7:0]	IO	port E	MPIO[7:0]
VSS(GND)	PWR	masa	GND
VDD(+5V)	PWR	zasilanie (+5V)	5V

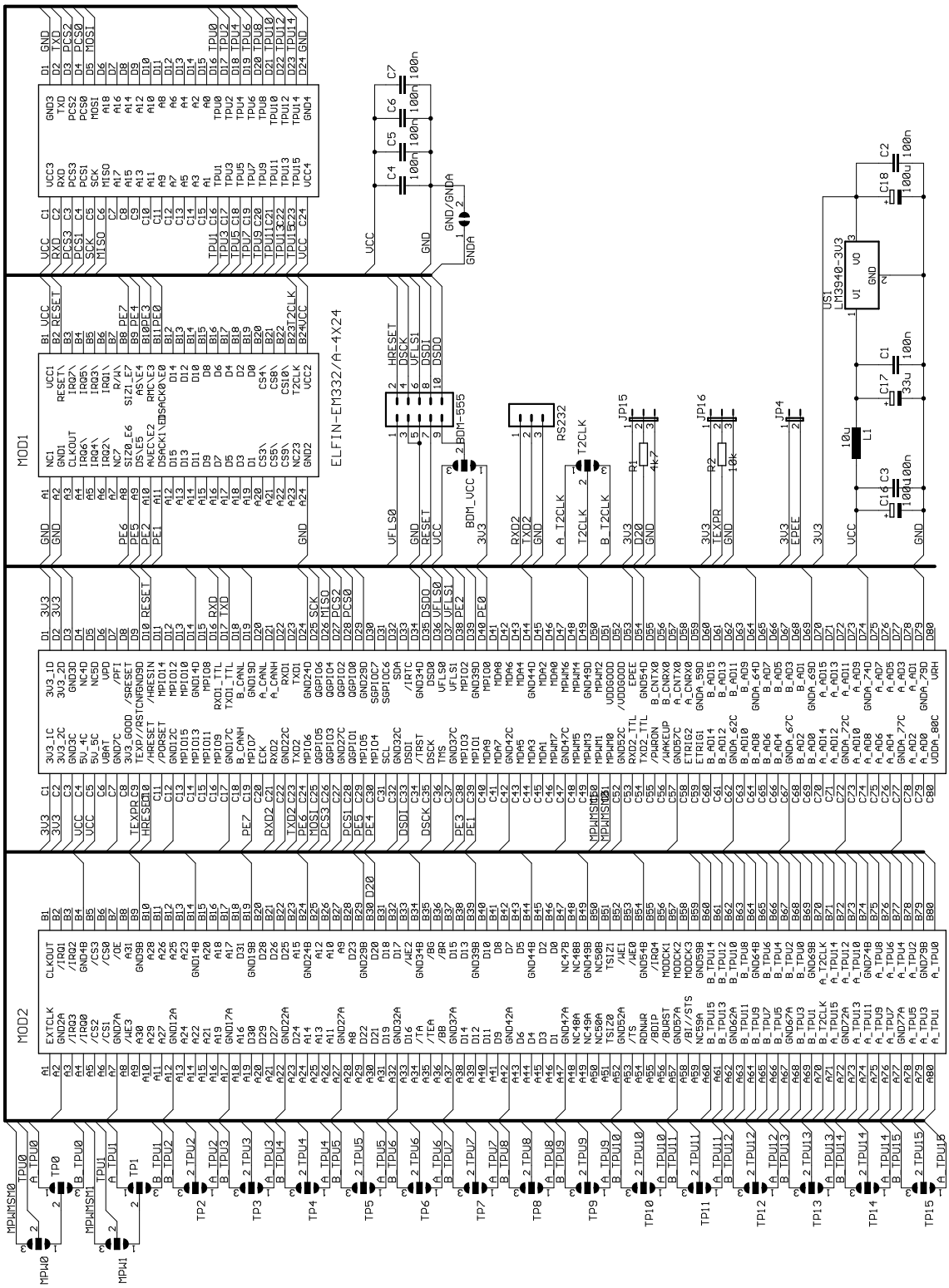
Tablica 4: Połączenia sygnałów EM332 i phyCORE MPC555

W MPC555 są dostępne dwa bloki TPU3 (A_TPU - połączenie 1-2 i B_TPU - połączenie 2-3), pomiędzy które można rozdzielić zadania jednego TPU z jednostki MC68332. Warunkiem prawidłowego rozdzielenia jest, by oba kanały QDEC oraz kanał QDVEL dla każdego koła były zlokalizowane w jednym bloku TPU. W przypadku kanałów wejściowych zalecane jest połączenie wszystkich trzech wyprowadzeń zworek (1-2-3) w celu przeniesienia decyzji na poziom oprogramowania. Nieco inaczej jest w przypadku wyjść (PWM). Tu należy zdecydować, czy sterowanie napięciem zasilającym silniki przez PWM ma się odbywać przez TPU (zworka MPWx w pozycji 1-2), czy przez MPWMSMx (zworka w pozycji 2-3). Zworka GND/GNDA pozwala połączyć masę analogową z masą cyfrową modułu phyCORE. Jej pozycja na płytce pośredniej nie ma znaczenia (blok analogowy nie jest wykorzystywany).
Na płytce wyprowadzono złącze transmisji szeregowej RS232 z drugiego modułu SCI. Można je wykorzystać w celach testowych lub uruchomieniowych. Wyprowadzenia podano w tab 3.

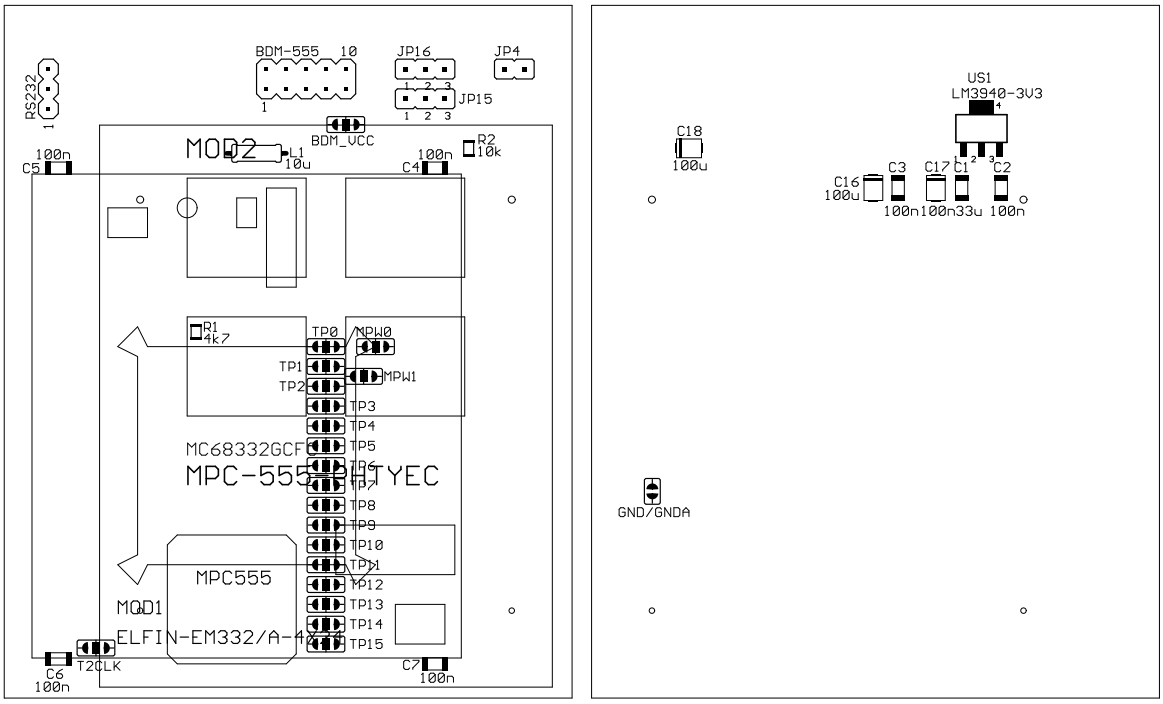
sygnał	styk	styk	sygnał
VFLS0	1	2	HRESET
GND	3	4	DSCK
GND	5	6	VFSL1
RESET	7	8	DSDI
BDM_VCC	9	10	DSDO

Tablica 5: Wyprowadzenie portu uruchomieniowegoo (złącze BDM555)

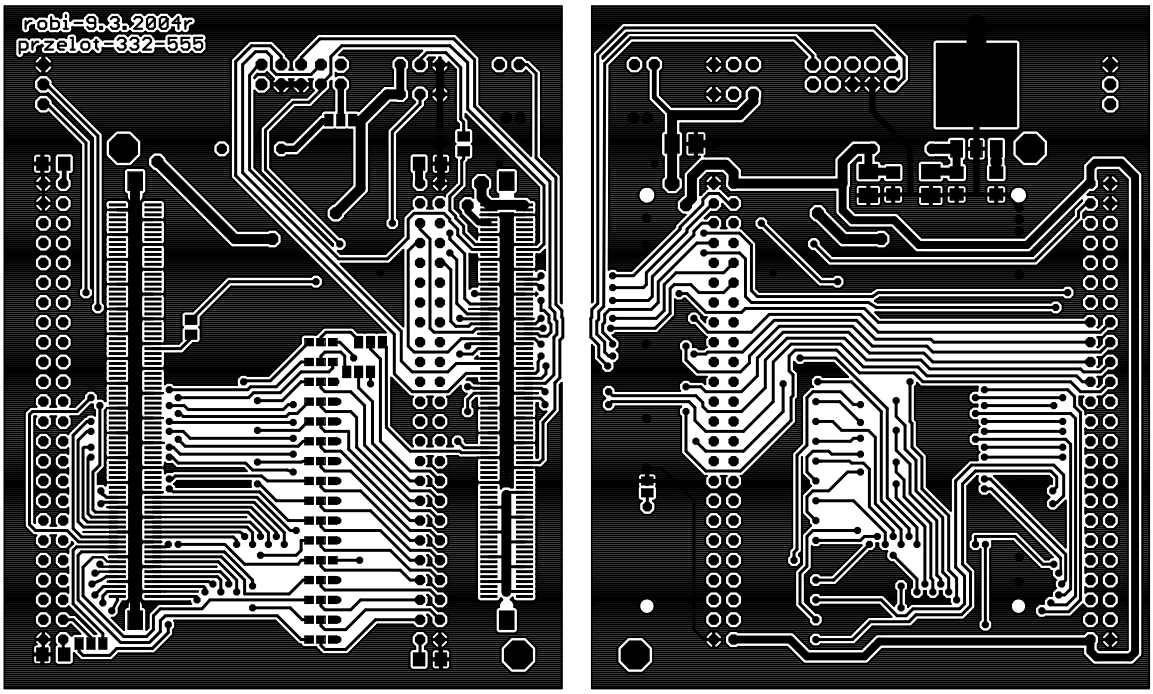
Sygnały interfejsu uruchomieniowego BDM (*Background Debug Mode*) umieszczono na złączu BDM555. Zestawienie sygnałów podano w tab. 5. Zasilanie dla interfejsu BDM wybiera się zworką lutowaną BDM_VCC (1-2 - 3.3V, 2-3 - 5V). W przypadku korzystania z interfejsu zgodnego z Wiggler firmy Macraigor Systems należy wybrać zasilanie 5V (2-3).



Rysunek 5: Schemat płytki przejściowej



Rysunek 6: Rozmieszczenie elementów



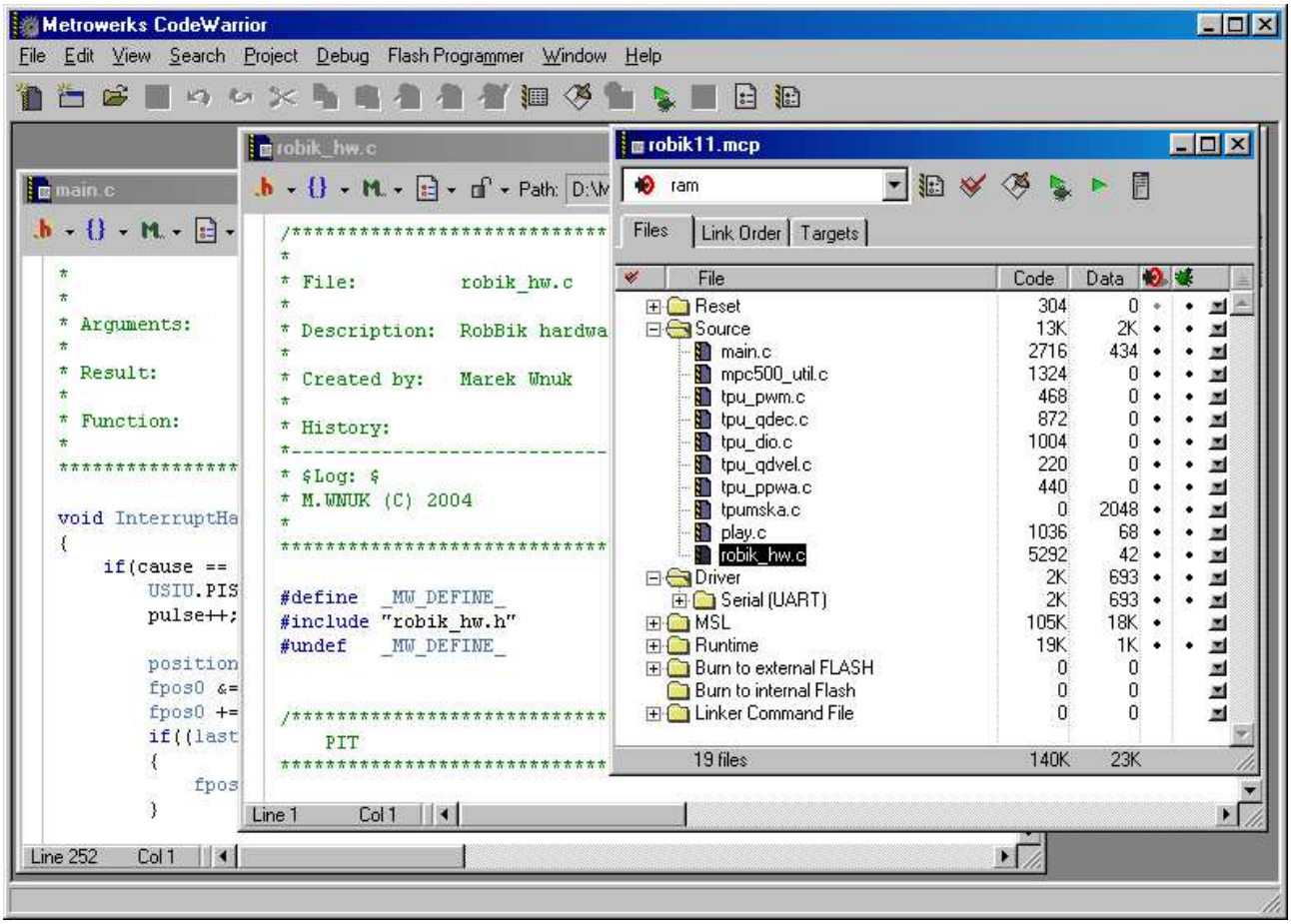
Rysunek 7: Mozaika obwodu drukowanego

3 Oprogramowanie dla nowego sterownika

Zmiana jednostki centralnej EM332 z mikrokontrolerem MC68332 na phyCORE z mikrokontrolerem MPC555 spowodowała konieczność przygotowania nowego oprogramowania dla sterownika wózka MK. Konieczne było również użycie innego środowiska programistycznego (uruchomieniowego) do przygotowywania programów.

3.1 Oprogramowanie uruchomieniowe dla MPC555

Jako środowisko uruchomieniowe dla oprogramowania nowego sterownika wózka MK wybrano pakiet *CodeWarrior for PowerPC Embedded Systems, Release 6.5* firmy Metrowerks. Wybór ten, poza względami merytorycznymi, był podyktowany faktem posiadania w Laboratorium Robotyki ICT PWr licencjonowanej instalacji tego pakietu zakupionej ze środków KBN w celu realizacji projektu RoBall. [1]



Rysunek 8: CW IDE z widocznym oknem zarządzania projektem

Centralnym narzędziem pakietu jest środowisko zintegrowane (IDE - *Integrated Development Environment*). Widok okna IDE przedstawiono na rys. 8. Dostarcza ono następujących usług:

[1] CodeWarrior jest pakietem komercyjnym o znacznej cenie (w wersji pełnej ok. $5000). W najbliższej przyszłości przewidziana jest migracja do środowiska publicznie dostępnego (GNU) w systemie Linux (*make*, *gcc*, *gdb*, *ddd* z obsługa BDM).

- rozbudowane zarządzanie projektem,
- łatwy w obsłudze edytor kodu źródłowego,
- zintegrowany debugger,
- przeglądarka różnych formatów plików źródłowych i wynikowych,
- zaawansowane kompilatory C, C++, EC++, asembler i linker dla PowerPC.

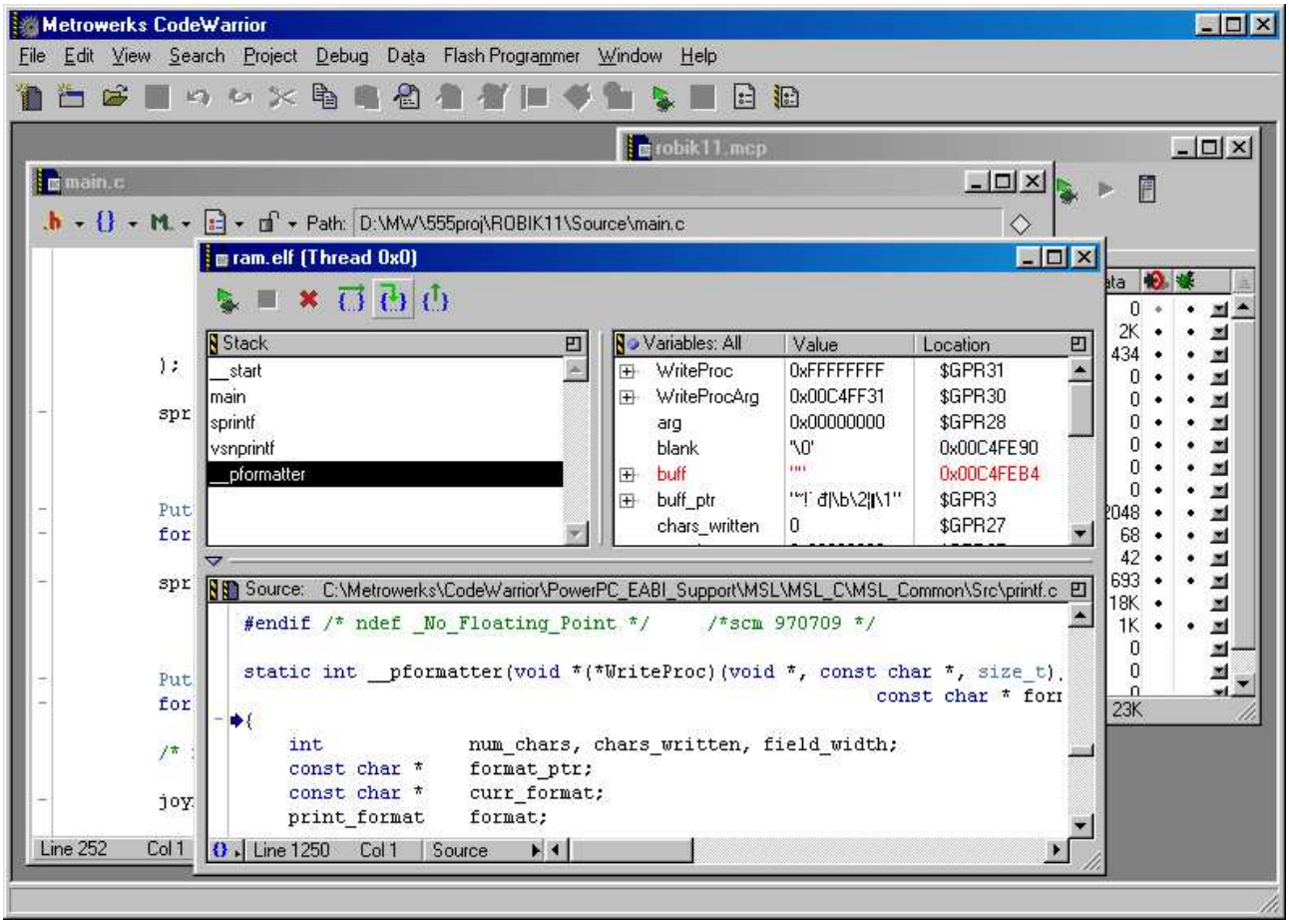
CW IDE umożliwia łatwe i elastyczne wykonywanie podstawowych prac programistycznych:

- Tworzenie i poprawianie kodów źródłowych programów. Edytor zapewnia kolorowe podświetlanie składni, szybki dostęp do plików zródłowych i dołączanych, pracę w trybie wielu okien itp. Przeglądarka oferuje dialogowe okna pomagające w operowaniu procedurami, strukturami danych, zmiennymi, klasami i plikami.
- Zarządzanie wieloma konfiguracjami docelowymi (ang. *targets*). Pozwala to na przechowywanie w projekcie wielu ustawień konfiguracji plików źródłowych, kompilatora i linkera. Przykładem mogą być konfiguracje przeznaczone do pracy w pamięci RAM, ROM i do programowania pamięci FLASH ze wspólnych źródeł.
- Tworzenie oprogramowania dla różnych procesorów w tym samym środowisku. Dostępne są kompilatory, asemblery i linkery dla wielu różnych platform (np. dla mikrokontrolerów MC9S12, czy procesorów sygnałowych DSP5683xx).
- Porównywanie plików obejmujące podkatalogi i otwarte okna edytora.

Narzędzia zawarte w pakiecie obejmują również obsługę wbudowanych wersji systemu Linux (*Embedded Linux Application Development*) z narzędziami GNU (kompilator, asembler, linker, archiwizator) przeniesionymi do środowiska Windows z wykorzystaniem biblioteki *Cygwin.DLL*.
Debugger symboliczny (*CodeWarrior Debugger*) pozwala uruchamiać programy nie tylko na wbudowanym symulatorze PowerPC, ale również na różnych modułach sprzętowych oferowanych przez inne firmy. W naszym przypadku takim modulem jest phyCORE-MPC555 firmy Phytec. Do komunikacji z modułem docelowym wykorzystywane są różne interfejsy wbsługujące złącze BDM lub JTAG mikrokontrolera MPC555 w systemie docelowym oraz port równoległy, szeregowy, lub Ethernet w komputerze roboczym. W naszym przypadku wykorzystano interfejs równoległy (zgodny z Wiggler firmy Macraigor Systems). Widok okna debuggera w srodowisku CW IDE przedstawiono na rys. 9.
Debugger pozwala:

- przeglądać program w postaci kodu źródłowego, zdeasemblowanej zawartości pamięci, lub jednocześnie w obu formach,
- uruchamiać program krok po kroku na poziomie kodu źródłowego lub języka maszynowego,
- ustawiać pułapki zarówno dla kodu w pamięci RAM, jak i ROM (FLASH),
- przeglądać zawartość zmiennych, komórek pamięci i rejestrów przez wskazanie ich w oknie kodu wskaźnikiem myszy,
- modyfikować przez wybór myszą i wprowadzenie nowej wartości.

Więcej informacji o opisanym środowisku można znaleźć na stronie:
<http://www.metrowerks.com/MW/Develop/Embedded/PowerPC/Default.htm>.



Rysunek 9: CW IDE z widocznym oknem debuggera

3.2 Oprogramowanie sterownika MPC555 dla wózka MK

Definicje stałych, makrodefinicje i struktury danych związane ze sprzętem wózka **MK** umieszczono w pliku *cart_hw.h*.
Niektóre parametry są wydzielone na początku pliku, aby ułatwić użytkownikowi konfigurowanie oprogramowania sterownika:

```
#define TPSEC           1000    // requested number of ticks per second
#define PWM_FRQ         1000    // requested PWM frequency [Hz]

#define USE_MPWM        0       // set to 1 if MPWM in use, 0 otherwise
#define USE_TPU_B       0       // set to 1 if ch2 uses TPU_B, 0 otherwise

#define JOY_ZONE        50      // dead zone for joystick neutral position

#define ADXL_PERIODS    1       // requested nbr of accumulated PPWA periods
```

Parametr `TPSEC` oznacza częstotliwość przerwań cyklicznych. Decyduje on o szybkości pracy ste-

rownika (procedura użytkownika będzie wykonywana TPSEC razy na sekundę).
Wybór częstotliwości generowanego sygnału PWM (używanego przy napięciowym sterowaniu silników) jest możliwy przez zmienianie parametru PWM_FRQ, który oznacza częstotliwość tego przebiegu. Sygnał PWM może być generowany za pomocą modułu MPWM mikrokontrolera MPC555 (USE_PWM ustawione na 1) lub przez funkcję PWM w TPU3 (USE_PWM ustawione na 0). Ustawienie tego parametru musi być zgodne z wybranymi połączeniami zworek MPWx na płytce sterownika.
Parametr JOY_ZONE określa szerokość martwej strefy wokół położenia neutralnego joystick-a.
Odczyt akcelerometrów przy pomocy funkcji PPWA TPU3 może się odbywać z prostą filtracją (przez uśrednianie pewnej ilości kolejnych okresów). Ilość uśrednianych okresów podaje parametr ADXL_PERIODS.
W celu ułatwienia programowej obsługi portów równoległych użytych do sterowania diodami LED i kierunkiem obrotów silników zdefiniowano odpowiednie struktury danych i makrodefinicje pozwalające czytelnie odwoływać się do bitów portu z poziomu programu napisanego w języku C:

```
struct PortE{
    VUINT16:8;
    VUINT16 SW:1;
    VUINT16:1;
    VUINT16 L_Y:1;
    VUINT16:1;
    VUINT16 L_G:1;
    VUINT16 L_R:1;
    VUINT16 DIR1:1;
    VUINT16 DIR0:1;
};

#define PORT_E  (*(struct PortE *) 0x00306100)

struct PortEDir{
    VUINT16:8;
    VUINT16 SW:1;
    VUINT16 E6:1;
    VUINT16 L_Y:1;
    VUINT16 E4:1;
    VUINT16 L_G:1;
    VUINT16 L_R:1;
    VUINT16 DIR1:1;
    VUINT16 DIR0:1;
};

#define PE_DIR  (*(struct PortEDir *) 0x00306102)

#define InitLeds()  {MPIOSMDDR=0x00af;\
                     PORT_E.L_G=1;\
                     PORT_E.L_Y=1;\
                     PORT_E.L_R=1;}
```

```
#define LED_GREEN   PORT_E.L_G
#define LED_YELLOW  PORT_E.L_Y
#define LED_RED     PORT_E.L_R

#define SWITCH      PORT_E.SW
```

Poszczególne bity portu MPIOSMDR w MPC555 są zdefiniowane jako pola bitowe w strukturze `struct PortE`. Odpowiednie makrodefinicje odwołują się do tych pól w obszarze przestrzeni adresowej zdefiniowanej przez `PORT_E` jako wyłuskanie stałego wskaźnika o wartości równej adresowi MPIOSMDR w MPC555. Analogicznie określone są pola bitowe rejestru kierunku tego portu - MPIOSMDDR (`struct PortEDir`, `PE_DIR`).
Obsługę TPU ułatwiają następujące definicje stałych i makrodefinicje:

```
#define TRAM_BASE 0x302000

#define XTAL 20000000
#define TCR1CK  ((XTAL)/4)  // TPU time base TCR1
#define PWM_PER ((TCR1CK+(PWM_FRQ>>1))/PWM_FRQ)

/* TPU_A/B channels usage */

#define PWMCHAN0    1       // TPU channel for PWM ch0
#define PWMCHAN1    0       // TPU channel for PWM ch1

#define QDECCHAN0   4       // TPU channel for QDEC ch0
#define QDVELCHAN0  10      // TPU channel for QDVEL ch0

#define QDECCHAN1   2       // TPU channel for QDEC ch1
#define QDVELCHAN1  9       // TPU channel for QDVEL ch1

#define PPWACHX     6       // TPU channel for PPWA ACCX
#define PPWACHY     7       // TPU channel for PPWA ACCY
#define PERCHAN     8       // TPU channel for PPWA period
```

Parametr `XTAL` oznacza częstotliwość rezonatora kwarcowego na module phyCORE-MPC555 i nie powinien być zmieniany przez użytkownika. Na jego podstawie określono `TCR1CK` - częstotliwość podstawy czasu dla TPU3 oraz wyliczono (korzystając z definiowanej przez użytkownika częstotliwości PWM - `PWM_FRQ`) parametr `PWM_PER` potrzebny do inicjalizacji i obsługi kanałów PWM.
Pozostałe definicje odpowiadają sprzętowej konfiguracji sterownika (wykorzystaniu kanałów TPU_A i TPU_B do obsługi różnych sygnałów): `PWMCHANx`, `QDECCHANx`, `QDVELCHANx` - kanały do obsługi napędów, `PPWACHx`, `PERCHANx` - odczyt wypełnienia i okresu przebiegu wyjściowego akcelerometru dwuosiowego.
Przetworniki analogowo-cyfrowe i cyfrowo-analogowe są przyłączone do magistrali QSPI. Poniższe makra definiują maski sygnałów PCS do wybierania poszczególnych urządzeń na tej magistrali:

```
#define DESELECT    0xF     // no SLAVEs selected

#define DACSEL      0xD     // DAC is selected by LOW on PCS1

/*
    DAC MAX529 outputs:
        0       current value to be set in ch0  (unipolar)
        1       current value to be set in ch1  (unipolar)
 */

#define ADCSEL      0xE     // ADC is selected by LOW on PCS0

/*
    ADC TLC2543 inputs:
        0       current measured in ch0 (unipolar)
        1       current measured in ch1 (unipolar)
        2       temperature sense TMP1  (unipolar)
        3       JOYX                    (unipolar)
        4       JOYY                    (unipolar)
        5       ENC03J OUT              (unipolar)
 */
```

`DACSEL` służy do wybrania przetwornika cyfrowo-analogowego, `ADCSEL` - przetwornika analogowo-cyfrowego. W komentarzach podano przypisanie kanałów przetworników do sygnałów sterownika. Odczytywanie wielkości mierzonych z poziomu programu napisanego w języku C ułatwiają następujące makrodefinicje:

```
#define TPU_PPWA_ACCUM 3
#define TPU_PPWA_FUN   15

#define Pos0    (tpua->PARM.R[QDECCHAN0][TPU_QDEC_POSITION_COUNT])
#if USE_TPU_B
#define Pos1    (tpub->PARM.R[QDECCHAN1][TPU_QDEC_POSITION_COUNT])
#else
#define Pos1    (tpua->PARM.R[QDECCHAN1][TPU_QDEC_POSITION_COUNT])
#endif // if USE_TPU_B

#define Vel0    (tpua->PARM.R[QDVELCHAN0][TPU_QDVEL_DIFTIME])
#if USE_TPU_B
#define Vel1    (tpub->PARM.R[QDVELCHAN1][TPU_QDVEL_DIFTIME])
#else
#define Vel1    (tpua->PARM.R[QDVELCHAN1][TPU_QDVEL_DIFTIME])
#endif // if USE_TPU_B

#define AdxlX   (tpua->PARM.R[PPWACHX][TPU_PPWA_ACCUM])
#if USE_TPU_B
```

```
#define AdxlY   (tpub->PARM.R[PPWACHY][TPU_PPWA_ACCUM])
#else
#define AdxlY   (tpua->PARM.R[PPWACHY][TPU_PPWA_ACCUM])
#endif // if USE_TPU_B
#define AdxlP   (tpua->PARM.R[PERCHAN][TPU_PPWA_ACCUM])

#define Iact0   (QSMCM.RECRAM[0].R)
#define Iact1   (QSMCM.RECRAM[1].R)

#define Temp    (QSMCM.RECRAM[2].R)

#define Gyro    (QSMCM.RECRAM[5].R)

#define JoyX    (QSMCM.RECRAM[4].R)
#define JoyY    (QSMCM.RECRAM[3].R)
```

Niektóre z nich są określone warunkowo, z automatycznym uwzględnieniem konfiguracji zworek na płytce sterownika, dzięki uwzględnieniu parametru `USE_TPU_B` opisanego wcześniej.
Odczyt obejmuje pomiary: parametrów ruchu kół (`Posx` - położenie, `Velx` - prędkość), wielkości prądów (`Iactx`), temperatury (`Temp`), przyspieszeń liniowych (`AdxlX(Y)` - szerokość impulsu, `AdxlP` - okres przebiegu), prędkości kątowej wahadła (`Gyro`) oraz położeń joystick-a (`Joyx`).
Podobnie ułatwione jest zadawanie sygnałów sterujących:

```
#define SetI0(x) QSMCM.TRANRAM[7].R = (x) & 0xff
#define SetI1(x) QSMCM.TRANRAM[9].R = (x) & 0xff

#if USE_MPWM
#define SetU0(x) MIOS1.MPWMSM0PULR.R = (UINT16)(PWM_PER-(x))
#define SetU1(x) MIOS1.MPWMSM1PULR.R = (UINT16)(PWM_PER-(x))
#else // if USE_MPWM
#define SetU0(x) tpu_pwm_update(tpua,PWMCHAN0,PWM_PER,(UINT16)(PWM_PER-(x)),0)
#if USE_TPU_B
#define SetU1(x) tpu_pwm_update(tpub,PWMCHAN1,PWM_PER,(UINT16)(PWM_PER-(x)),0)
#else
#define SetU1(x) tpu_pwm_update(tpua,PWMCHAN1,PWM_PER,(UINT16)(PWM_PER-(x)),0)
#endif // if USE_TPU_B
#endif // if USE_MPWM
```

Odpowiednie makrodefinicje obejmują podstawienia i wywołania funkcji pomocniczych ustawiających odpowiednie wielkości w TPU, MPWM i przetworniku cyfrowo-analogowym. Ustawianie prądów zadanych (`SetIx`) odbywa się przez wpisanie wartości zadanej do kolejki nadawczej QSPI. Ustawianie napięć (`SetUx`) odbywa się (w zależności od parametru `USE_MPWM`) przez wpisanie wartości do rejestru MPWMSMxPULR modułu MPWM lub przez wywołanie funkcji `tpu_pwm_update` ustawiającej czas stanu wysokiego PWM w odpowiednim kanale TPU. Wybór TPU jest możliwy dzięki ustawieniu parametru `USE_TPU_B`.
Inicjalizacja sprzętu sterownika polega na wywołaniu kilku procedur odpowiedzialnych za ustawienie parametrów pracy poszczególnych bloków. Parametry i prototypy tych procedur podano poniżej.

```
#define PIT_TB (XTAL/256)       // PIT time base frequency

/* calculated PIT time constant */
#define PIT_CONST    ((PIT_TB+(TPSEC>>1))/TPSEC)

EXTERN INT32 pulse; // free running tick counter

void PitInit(void);
```

PIT_TB wyznacza częstotliwość podstawy czasu dla bloku PIT (ang. *Periodic Interrupt Timer*). Parametr dla dzielnika częstotliwości PIT (PIT_CONST) jest wyliczany na podstawie wybranej przez użytkownika częstotliwości przerwań cyklicznych (parametru TPSEC).
Zmienna globalna pulse jest licznikiem przerwań cyklicznych wyznaczających podstawę czasu dla sterownika.
Procedura PitInit() uruchamia przerwania cykliczne z częstotliwością zadaną przez TPSEC.

```
extern const unsigned char tpumska[];   // TPU mask with QDVEL (MW'04)
EXTERN struct TPU3_tag *tpua;           // pointer for TPU routines
EXTERN struct TPU3_tag *tpub;           // pointer for TPU routines

void TpuInit(void);
```

W celu pomiaru prędkości kątowej kół wykorzystano funkcję QDVEL, której nie ma w standardowej masce TPU3. Obraz binarny maski mikrokodu z dodaną funkcją QDVEL (tpumska[]) jest dołączany przez linker. Wygodny dostęp do rejestrów TPU_A i TPU_B jest możliwy dzięki funkcjom i strukturom zdefiniowanym w pliku *tpu.h*. Zmienne globalne tpua i tpub są wskaźnikami na struktury opisujące rejestry TPU3 inicjalizowanymi w procedurze TpuInit(). W tejże procedurze są inicjalizowane wszystkie kanały TPU_A i TPU_B używane w ustalonej konfiguracji (według parametru USE_TPU_B).

4 Uwagi końcowe

Opisana modyfikacja została wstępnie przetestowana przez uruchomienie prostego programu umożliwiającego sterowanie ręczne robotem przy pomocy joystick-a. Wszystkie układy pomiarowe i sygnały sterujące zachowywały się prawidłowo. Stanowisko jest gotowe do implementowania i badania bardziej zaawansowanych algorytmów sterowania wózkiem MK. Wydaje się jednak, że prace te należy poprzedzić zbadaniem parametrów modelu dynamiki robota ([11]).

Literatura

[1] Muszyński R., Tchoń K., *Singularities and mobility of nonholonomic systems: the ball rolling on a plane*, 6th IFAC Symposium on Robot Control, SYROCO'00, Vienna 2000, Preprints vol.1, ss. 259-264.

[2] *phyCORE-MPC555 Hardware Manual*, Edition August 2001, PHYTEC.

[3] Kabała M., Tchoń K., Wnuk M., *Robot mobilny napędzany w układzie wewnętrznym*, VII KKR, Lądek Zdrój, 2001, Prace Naukowe ICT PWr., Konferencje 46, t.1, ss. 149-158.

[4] Kabała M., Wnuk M., *Konstrukcja i oprogramowanie dwukołowego robota mobilnego*, Raport SPR 20/2002, Inst. Cyb. Techn. PWr, 2002.

[5] Tchoń K., Kabała M., Wnuk M., *Algorytm śledzenia trajektorii robota mobilnego MK*, XIV Krajowa Konferencja Automatyki, Zielona Góra, 2002.

[6] Kabała M., Tchoń K., Wnuk M., *Dwukołowy, nieholonomiczny robot mobilny*, Materiały Konferencji Automation 2002, Warszawa, 2002, ss. 269-280.

[7] Kabała M., Wnuk M., *Kulisty robot mobilny RoBall (projekt wstępny)*, Raport SPR 37/2002, Inst. Cyb. Techn. PWr, 2002.

[8] Kabała M., Muszyński R., Wnuk M., *Singularity robust, dynamic linearization control algorithm for MK mobile robot*, 7th Symposium on Robot Control SYROCO'03, Wrocław, 2003, vol.1, ss. 557-562.

[9] Muszyński R., Wronka C., *MK mobile robot with simplified controller*, 7th Symposium on Robot Control SYROCO'03, Wrocław, 2003, vol.1, ss. 563-567.

[10] Wnuk M., *Moduł z mikrokontrolerem MC68332*, Raport SPR 7/2004, Inst. Cyb. Techn. PWr, 2004.

[11] Wnuk M., *Pomiar parametrów fizycznych dwukołowego robota mobilnego*, w: Postępy robotyki: Przemysłowe i medyczne systemy robotyczne, WKiŁ, Warszawa, 2004 (w druku).

Robert Szlawski
dr inż. Marek Wnuk
Instytut Cybernetyki Technicznej
Politechniki Wrocławskiej
ul. Janiszewskiego 11/17
50-372 Wrocław

Niniejszy raport otrzymują:

1. OINT ..... - 1 egz.

2. Zleceniodawca ..... - 2 egz.

3. Autorzy ..... - 2 egz.

Razem : 5 egz.

Raport wpłynął do redakcji I-6
we wrześniu 2004 roku.