УДК 29.113.073

ПЕРЕДАЧА ДАННЫХ С SIMULINK НА ОБЪЕКТЫ УПРАВЛЕНИЯ ПО CAN BUS

Михайлов Владимир Георгиевич, канд. техн. наук,
Республика Беларусь, г. Минск, sapr7@mail.ru

Аннотация. Рассмотрено использование для передачи данных с Simulink по CAN BUS, который сейчас широко применяется в робототехнике, авиа и автомобилестроении в системах управления для обмена данными между микроконтроллерами, благодаря своей простоте и надежности.

Выявлено, что ПО CAN_API.dll, откомпилированное в Microsoft Visual Studio (MVS) не работает с TDM-GCC-64 Matlab/Simulink из-за разного подхода в именах функций dll по стандарту C++11. Чтобы устранить эту проблему требуется перекомпилирование dll в среде TDM-GCC-64 под Windows, которое может выполнить только разработчик dll.

Оптимальным выбором для реализации передачи данных с Simulink на стенды с электроактуаторами по CAN BUS является использование адаптеров Titan TITAN ELECTRONICS INC, которые позволяют реализовать частоту обмена более 100 Гц для 6-ти осной платформы.

Предложен способ сжатия информации и повышения скорости обмена в 2 раза за счет побайтного занесения двух значений float в поле данных с использованием одинаковых значений идентификаторов объектов управления для двух цилиндров и последующего их разделения в программе микроконтроллеров цилиндров.

Разработана программа передачи/обмена данных с Simulink на устройства управления стендов, для которой оптимальным значением квантования является chc=350. Все это вместе позволяет реализовать частоту обмена 230 Гц и режим реального времени моделирования.

Ключевые слова: Имитационное моделирование, автомобиль, симулятор, электроактуатор, Matlab/Simulink, CAN BUS.

DATA TRANSMISSION WITH SIMULINK ON CONTROL OBJECTS ON CAN BUS

Mikhailov Vladimir. G., associate professor,
Minsk, Republic Belarus, sapr7@mail.ru

Abstract. Use for data transmission of CAN BUS which is widely applied in robotics, an avia and automotive industry in management systems to data exchange between microcontrollers, thanks to the simplicity and reliability now is considered.

It is revealed that CAN_API.dll software, compiled in the Microsoft Visual Studio (MVS) does not work with TDM-GCC-64 Matlab/Simulink because of different approach in names of the dll functions according to the standard C ++ 11. To fix this problem recompile of dll in the TDM-GCC-64 environment under Windows which only the dll developer can execute is required.

The optimum choice for implementation of data transmission with Simulink on stands with electroactuators on CAN BUS is use of adapters of TITAN ELECTRONICS INC which allow to realize the frequency of exchange of 100 Hz and above for 6 axis platforms.

The way of information compression and fall forward of exchange twice due to byte-by-byte entering of two float values in the data field with use of identical values of object identificators of control for two cylinders and the subsequent their division in the program of microcontrollers of cylinders is offered.

The program of transfer/data exchange with Simulink on control devices of stands for which best value of quantization is chc=280 is developed. All this together allows to realize the frequency of exchange of f=100 of Hz and also acceptable time of modeling is higher.

Key words: Simulation modeling, vehicle, simulator, electroactuator, Matlab/Simulink, CAN BUS.

Введение. Сейчас за рубежом для сокращения времени разработки и доводки конструкций самолетов, автомобилей, различных объектов управления все чаще применяют методы имитационного моделирования, в процессе которых задействуются реальные объекты управления, исполнительные устройства и человек-оператор [1, 2]. Примером этого являются авиасимуляторы, полунатурное моделирование автомобиля на стенде с задействованием водителя с имитированием визуализации дорожной обстановки и воздействия реального макро и микропрофиля дороги (рис. 1). Это позволяет для автомобиля проработать на моделях большое количество дорожных ситуаций и вариантов конструкции и испытать их в стендовых условиях, отработать эргономику, значительно сократить время создания новых моделей машин, не подвергая риску водителей и испытателей. А на авиасимуляторах отработать критические ситуации и обучить пилотов.

Рис. 1. – Имитационное моделирование автомобиля в стендовых условиях и конструкция электроактуатора

В качестве исполнительных элементов в них сейчас широко используются более дешевые электроактуаторы в виде механического цилиндра с ременной и червячной шариковой передачей и электродвигателя с блоком электронного управления [3, 4].

Наиболее эффективным способом для имитационного моделирования является создание программы на языке C в S-Function Builder Matlab/Simulink в среде Windows 7–10 64 bit. Для разработки последней требуется использование компилятора TDM-GCC-64.

Сложным вопросом является обеспечение одновременного моделирования автомобиля в реальном масштабе времени на Matlab/Simulink и передача с него данных.

Обмен через бинарные файлы по сети не обеспечивает корректную передачу данных из-за несинхронизации процессов записи и чтения при частотах обмена > 100 Гц.

Для передачи данных целесообразно использовать протокол CAN BUS (1024 кб/с), который сейчас широко применяется в роботах, авиа и автомобилестроении в системах управления для обмена данными между микроконтроллерами, благодаря своей простоте, надежности и скорости. В нем используется витая пара в качестве кабеля (рис. 2).

Передача данных осуществляется с помощью кадра (frame), содержащего как служебную, так и передаваемую информацию. Поле данных составляет 8 символьных байт.

Для передачи данных по CAN BUS необходимо наличие аппаратных средств – адаптеров, которые связывают компьютерную технику с устройствами управления либо наличие их в устройствах управления. Сейчас большинство современных электроактуаторов и гидропульсаторов оборудованы интерфейсами CAN BUS и производители предлагают инструкции и средства создания программ на языке С.

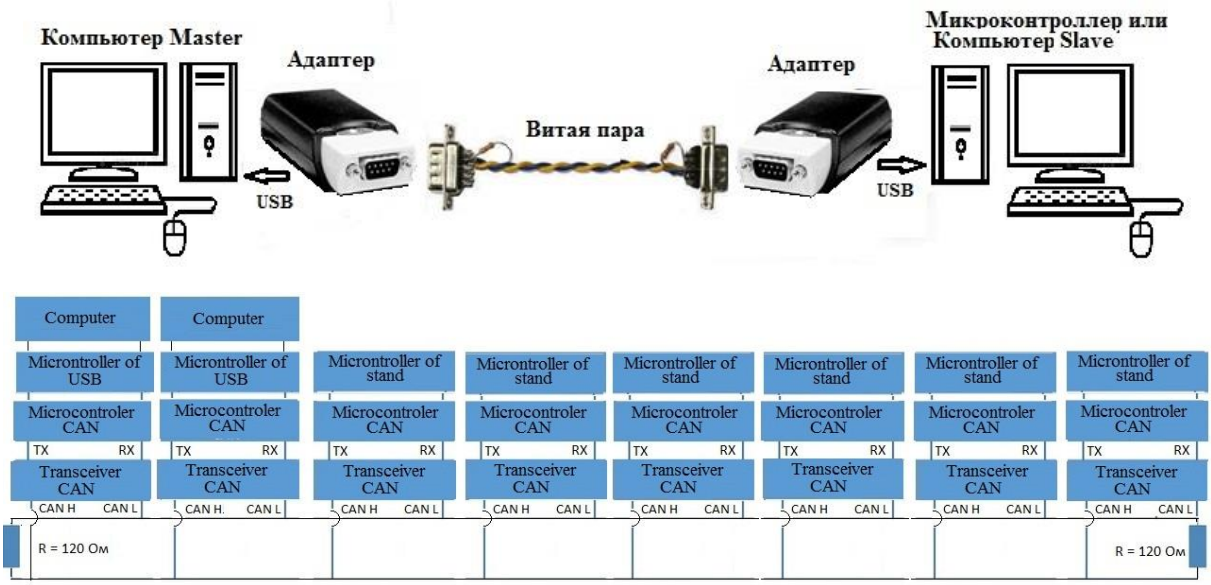

Рис. 2. – Предлагаемая общая схема передачи данных с Simulink на стенд

Помимо этого требуется специальное программное обеспечение (ПО) для адаптеров, которое отличается у разных производителей в виду различной реализация самого адаптера. Производители, как правило, предлагают ПО в виде CAN_API.dll, созданного в MicroSoft Visial Studio (MVS) либо Linux без предоставления исходных текстов.

Тестированием установлено, что ПО CAN_API.dll, откомпилированное в MVS, не работает в TDM-GCC-64 Matlab/Simulink из-за разного подхода в именах функций dll по стандарту C++11/17. Чтобы

устранить эту проблему требуется перекомпилирование dll в среде TDM-GCC-64 под Windows, которое может выполнить только разработчик dll.

Сейчас в версии Matlab/Simulink R2019b появились компоненты, позволяющие создать коммуникацию по CAN BUS через Virtual Channel, Vehicle Network Toolbox. Они требуют отдельной покупки и использования недешевых адаптеров определенных фирм, таких как, Vector, Kvaser, PEAK-System и других и их программного обеспечения.

К сожалению, нет публикаций по практической реализации данного решения для стендов с большим количеством цилиндров и оценок насколько это эффективно.

Целью данной работы является проработка решений и оценка возможности использования CAN BUS для передачи данных на электроактуаторы стенда и другие устройства программным путем через S-Function Builder и осуществление моделирования в Matlab/Simulink в реальном масштабе времени.

1. Выбор адаптера и средств разработки

Определяющим фактором для выбора адаптера является возможность реализации в нем частоты 100 Гц на каждый цилиндр из 6-ти, используемых в платформе 6-DOF. В этом случае можно обеспечить погрешность 1 % на верхней частоте. Исходя из этого был выбран адаптер фирмы Titan TITAN ELECTRONICS INC., SN #T16820100, Тайвань, являющийся наиболее оптимальным по соотношению цена-качество [8]. Кроме того, эта фирма пошла мне навстречу, бесплатно предоставила два адаптера и перекомпилировала ПО под TDM-GCC-64 и помогла на начальном этапе с освоением CAN BUS, за что я благодарен ей.

Для работы CAN BUS вначале необходимо установить драйвер USB-CAN **CDM21228_Setup.exe**. Определить какой подключен COM PORT (например, **COM3**). Затем запустить в командной строке **CAN_BAUDRATE_SET.exe COM3.** Должно быть получено сообщение

Searching…

Find COM PORT: COM3

Setting every baud rate to 3Mbit…

Set baud rate Success ,

свидетельствующее об успешной установке. После чего требуется отключить USB кабель от компьютера и через 5 с вновь его подсоединить либо выключить и включить компьютер.

Примечание. *В ряде случаев чтобы установить CAN_BAUDRATE_SET.exe на Windows 10 64 bit необходимо задать Свойства этого файла в режиме Совместимость, с отмеченными галочками, как показано на рисунок 3. После чего выполнить вышеприведенную команду.*

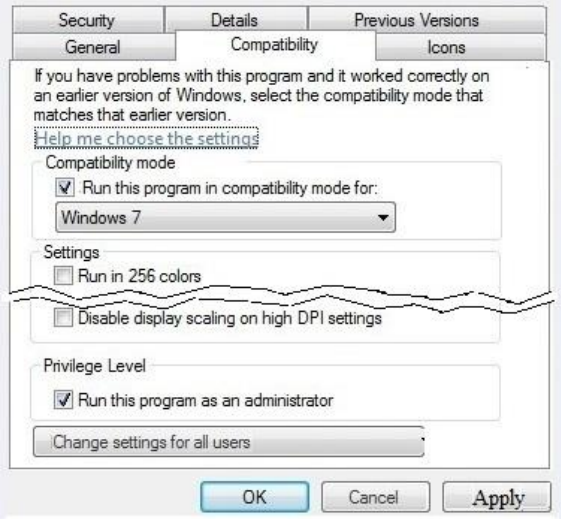

Рис. 3. – Установка режима совместимости для CAN_BAUDRATE_SET.exe

Для моделирования использовался пакет Matlab/Simulink R2015b (ode5 Dormand-Prince, t=0,001 с) ввиду того, что он работает в 2 раза быстрее чем R2018b/R2019b. Причиной последнего возможно является перенасыщенность его новыми функциями.

Для разработки программы использовался компилятор TDM-GCC-64 (tdm64-gcc-5.1.0-2.exe). Его установка и настройка описана в работе [7]. Для Windows 10 требуется еще установка пути на C:\ TDM-GCC-64\bin через кнопку Set Path на панели Matlab.

Моделирование проводилось с использованием реального возмущения дороги [9] в виде массива объемом 96000 записей типа float в электронной памяти и выполнением логических операций по переключению КПП, расчету тяговой динамики автомобиля в программе на С в S-Function Builder с использованием совершенных моделей, обеспечивающих погрешность 10–15 % с учетом спектра [6, 7, 9, 10]. Кроме этого в отдельных блоках Simulink моделировались вибрации, управляемость автомобиля. Визуализация осуществлялась с помощью миникомпьютера Raspberry 3B+.

Выбор такого подхода обуславливался необходимостью обеспечения максимальной скорости моделирования в реальном масштабе времени.

2. Реализация обмена информации

Реализация моделирования с передачей данных по CAN BUS осуществлялась с помощью структурной схемы (рисунок 4) и блок-схемы (рис. 5), в которой задействовано 18 интеграторов. Для моделирования использовался компьютер (AMD Rysen 5 2600, 6-ти ядерный 3,4/4 ГГц, RAM 16 Гб DDR-4 2666 МГц, накопитель M2. SSD 2500 Мб/с, MB ASUS PRIME B450 PLUS, t=0,0001 с) с передачей данных в асинхронном режиме, а также детали от игровой приставки. Прием сообщений осуществлялся на 2-х ядерном компьютере AMD 2,5 ГГц, RAM = 8 Гб, DDR-2 400 МГц, MB ASUS M2-VM, который имитировал исполнительное устройство.

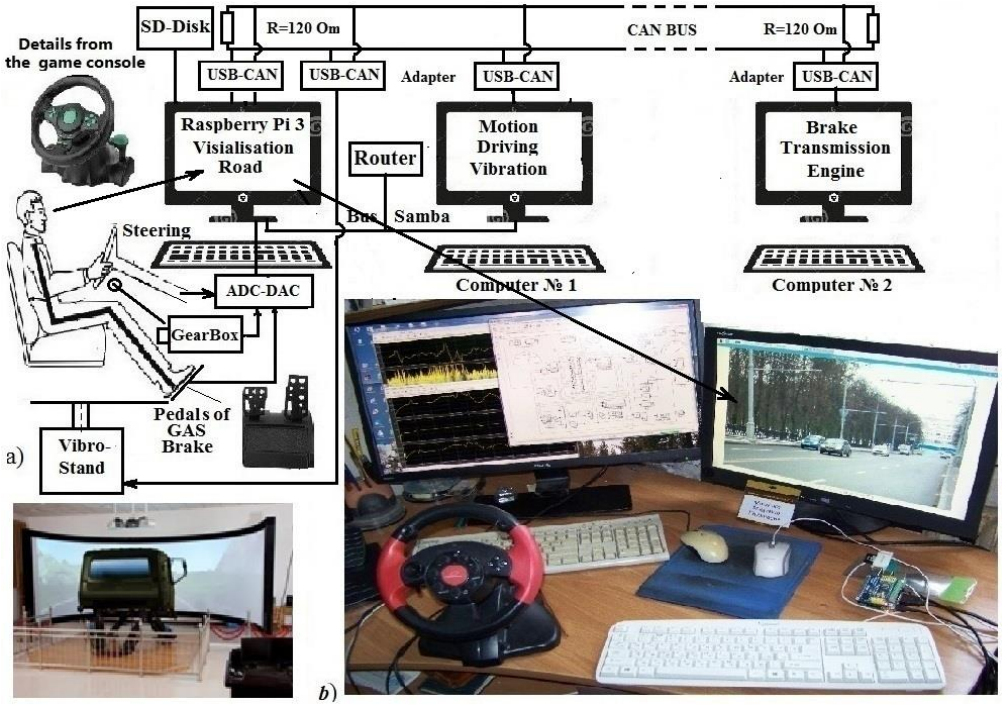

Рис. 4. – Схема имитационного моделирования и макетный настольный образец

Рис. 5. – Процесс и блок-схема моделирования движения, колебаний и управляемости автомобиля

Для моделирования автомобиля и передачи данных с Simulink на стенды разработана программа на языке С для S-Function Builder, общий объем которой составил 1372 строк исходного текста. Из них 850 строк служебные. Первоначальная отладка CAN BUS проводилась на

упрощенной программе **wrdc.c**, приведенной ниже, которая поможет Вам понять механизм работы CAN BUS и создать свою программу. Ее функция void **srd**(struct ds *ps, struct dr *pr) использована в общей программе для S-Function Builder.

```
#include <stdio.h>
#include <stdlib.h>
#include "CAN_API.h"
#include <windows.h>

//-----------struct send data CAN BUS
struct ds {
float a1;
float b1;
float a2;
float b2;
float a3;
float b3;
};
struct ds sf; // struct send data into CAN BUS

struct dr { // struct read data from CAN BUS
float a1;
float b1;
float a2;
float b2;
float a3;
float b3;
};
struct dr rf; // struct read data from CAN BUS

double stm=469.997;
int ch;
typedef TCAN_HANDLE (*FNPTR1)(CHAR *,CHAR *,CHAR *,CHAR *,void *,DWORD);
typedef TCAN_HANDLE (*FNPTR2)(TCAN_HANDLE);
typedef TCAN_HANDLE (*FNPTR3)(TCAN_HANDLE,CAN_MSG*);
typedef TCAN_HANDLE (*FNPTR4)(TCAN_HANDLE,CHAR *);

  HMODULE hMod;
  TCAN_HANDLE Handle;
  TCAN_STATUS Status;
//-------------Function Exchange data on CAN BUS
 void srd(void)
 {
byte *yf, *yf1, *yf2;
int q, nbz;
long j;
FILE *fp;
    CHAR *ComPort = "COM4";
    CHAR *szBitrate = "1000";
//"9216000";
    CHAR *acceptance_code = "1FFFFFFF";
```

```
    CHAR *acceptance_mask = "00000000";
    VOID *flags = CAN_TIMESTAMP_OFF;
    DWORD Mode = Normal; //LoopBack;
    FNPTR1 CAN_Open;
    FNPTR2 CAN_Close;
    FNPTR2 CAN_Flush;
    FNPTR3 CAN_Write;
    FNPTR3 CAN_Read;
    FNPTR4 CAN_Version;
    FNPTR2 CAN_Status;
    char version[10];
    CAN_MSG SendMSG;
    CAN_MSG RecvMSG;
//    Handle = -1;
    Status = 0;
    nbz=0;
    j=0;
//---------
    CAN_Open  =  (FNPTR1)GetProcAddress(hMod, "CAN_Open");
    CAN_Close =  (FNPTR2)GetProcAddress(hMod, "CAN_Close");
    CAN_Flush =  (FNPTR2)GetProcAddress(hMod, "CAN_Flush");
    CAN_Write =  (FNPTR3)GetProcAddress(hMod, "CAN_Write");
    CAN_Read  =  (FNPTR3)GetProcAddress(hMod, "CAN_Read");
    CAN_Version =(FNPTR4)GetProcAddress(hMod, "CAN_Version");
    CAN_Status = (FNPTR2)GetProcAddress(hMod, "CAN_Status");
    RecvMSG.Flags = CAN_FLAGS_STANDARD;  //EXTENDED;
    RecvMSG.Size = 8;
//--------------------
    if (ch==0)
Handle = CAN_Open ( ComPort, szBitrate, acceptance_code, acceptance_mask, flags, Mode );
//-- Example read data from RecvMSG.Data
    RecvMSG.Id = 0x001;
     memset ( version, 0, sizeof ( char ) * 10 );
    Status = CAN_Flush ( Handle );
    Status = CAN_Version ( Handle, version );
     for (j=0;j<60000;j++)  {
      Status = CAN_Read ( Handle, &RecvMSG );
    if ( Status == CAN_ERR_OK ) {
    switch (RecvMSG.Id) {
    case 1:   yf=(byte *)&rf.a1;      for (q=0;q<8; q++) *yf++=RecvMSG.Data[q];  break;
    case 2:   yf=(byte *)&rf.a1+8;  for (q=0;q<8; q++) *yf++=RecvMSG.Data[q]; break;
    case 3:   yf=(byte *)&rf.a1+16; for (q=0;q<8; q++) *yf++=RecvMSG.Data[q];  break;
    default:
    break;
    }
// Контроль приема данных
   fp=fopen("tc.txt","at");
    fprintf(fp,"Handle=%d ID=%X Status= %d %.4f %.4f %.4f %.4f %.4f %.4f\n", Handle,
RecvMSG.Id, Status,rf.a1,rf.b1,rf.a2,rf.b2,rf.a3,rf.b3);
   fclose(fp);
  }
  } //-- End Example read data from RecvMSG.Data
/*
//----------- Example send data into SendMSG.Data
    for (nbz=0;nbz<3;nbz++)   {
```

```
    yf=(byte *)&sf.a1+8*nb;
    for (q=0;q<8; q++)
    SendMSG.Data[q] = *yf++;
//----------------------
    switch (nbz) {
    case 0: SendMSG.Id = 0x001; break;
    case 1: SendMSG.Id = 0x002; break;
    case 2: SendMSG.Id = 0x003; break;
     default:
     break;
    }
    memset ( version, 0, sizeof ( char ) * 10 );
    Status = CAN_Flush ( Handle );
    Status = CAN_Version ( Handle, version );
    Status = CAN_Write ( Handle, &SendMSG );
    if (nbz==2) { // Контроль передачи данных
    fp=fopen("tcb.txt","at");
    fprintf(fp,"Handle=%d ID=%X Status= %d %.4f %.4f %.4f %.4f %.4f %.4f\n", Handle,
RecvMSG.Id, Status,rf.a1,rf.b1,rf.a2,rf.b2,rf.a3,rf.b3);
    fclose(fp);
}
//-- End Example send data into SendMSG.Data
*/
  if (ch==3999)
    Status = CAN_Close ( Handle );
}
//-------------End function sd()
int main(void)
{
float f1, f2;
int r;
long j;
   hMod = LoadLibrary ("CAN_API.DLL");
   if (hMod==NULL)  {
       printf ("LoadLibrary failed\n");
    }
   for (ch=0;ch<4000;ch++)
    srd();
    FreeLibrary(hMod);
    printf ( "Test finish_FreeLibrary\n" );
return 0;
} //End program main
```

Используемые в программе функции и настройка адаптера подробно описаны в [8]. В каталог с программой **wrdc.c** должны быть установлены файлы **CAN_API.dll** и **CAN_API.h**. Чтобы ее откомпилировать в TDM-GCC-64 введите команду

gcc wrdc.c

В результате получится исполняемый файл **a.exe**, который использовался для приема сообщений по CAN BUS. Особенностью примененного подхода является то, что вначале информация для 6-ти актуаторов заносится в структуру данных struct ds sf. Далее в функции void srd() реализовано сжатие данных: занесение в поле 8 байт SendMSG.Data[0] двух значений float данных побайтно вместо одного (фрагмент выделен желтым цветом). Фактически это передача данных блоками на два актуатора сразу, имеющих одинаковый идентификатор. А в программе контроллера актуатора производится программное их разделение. Это позволяет повысить производительность обмена данными в два раза (тем самым до 6 Мб/с).

3. Результаты моделирования

Передача данных осуществлялась с периодичностью по счетчику циклов ***chc***, который обнулялся при достижении максимальной величины. Счетчик циклов ***chc*** определяет количество тактов моделирования, после которого происходит передача данных на 6 электроактуаторов по CAN BUS. Это позволяет сократить затраты времени на передачу информации. Квантование в 100 Гц вполне достаточно для воспроизведения стендом верхних частот 10 Гц с погрешностью 1 %.

Результаты моделирования приведены на рис. 6. Базовое время моделирования без CAN BUS на 6-ти ядерном компьютере составляет 105 с при реальном процессе 470 с. Как видно из графика использование CAN BUS значительно увеличивает время моделирования (в 4,5 раза при *chc* =350). Характер изменения частоты посылки и приема сообщений имеет вид небольшой нелинейности.

При уменьшении *chc* резко увеличивается время моделирования ***t*** и оно определяет выбор *chc*. При параметре квантования ***chc***=350 достигается режим реального времени моделирования, частота передачи данных при этом составляет 230 Гц. Выяснено, что режим реального времени может

также достигнут путем подбора шага интегрирования, частоты процессора в UEFI BIOS материнской платы, подключением других подсистем, функций, использованием более новой версии Matlab/Simulink R2019b, работающей медленнее, отказом от сжатия данных; использованием более сложной пространственной модели автомобиля, применением синхронного режима передачи.

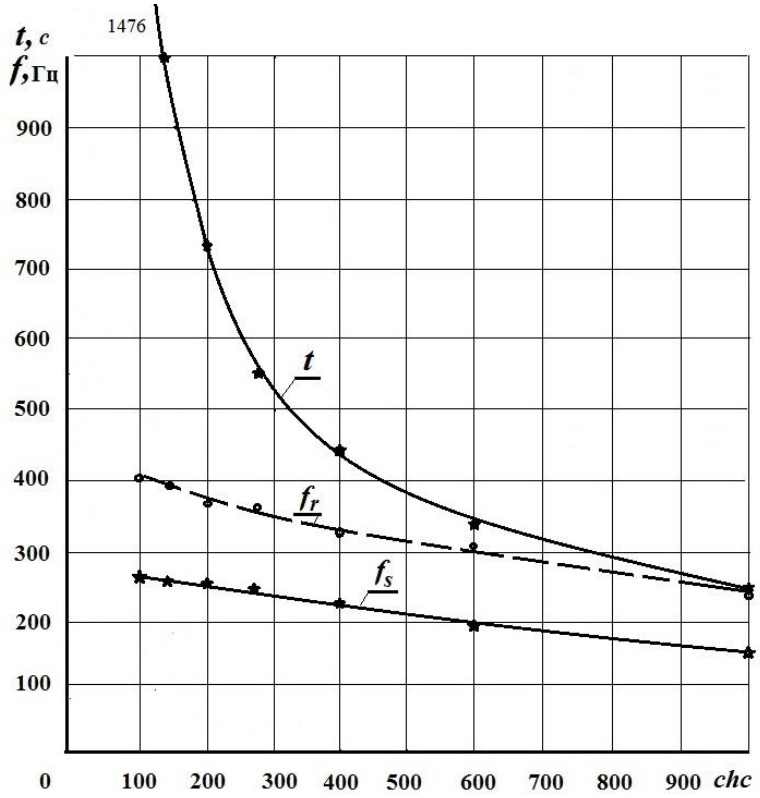

Рис. 6. – Влияние параметра квантования на частоту посылки и приема сообщений и время моделирования.
t – время моделирования, f_s – частота посылки сообщений, f_r – частота приема сообщений

При увеличении параметра квантования с 350 до 1000 частота передачи данных уменьшается с 230 до 150 Гц. Но этого вполне достаточно для управления стендами (>100 Гц). Имеющийся запас в дальнейшем позволяет реализовать моделирование при усложнении модели автомобиля.

Скорость моделирования 2-х ядерного компьютера с CAN BUS получилась 8,7 раза меньше чем у современного и недостаточна для реализации режима реального времени.

Частота чтения данных получается в 1,5 раза быстрее чем их запись за счет исключения процесса моделирования.

Результаты моделирования свидетельствуют о возможности разнесение вычислений подсистем между компьютерами при использовании CAN BUS для решения сложных задач и целесообразности полного перехода на CAN BUS.

К сожалению, автопроизводители СНГ (РФ, РБ) пока не проявляют интереса к этому из-за сложной экономической ситуации и своей недальновидности. Как не востребованы мои знания и опыт. Из-за чего не удалось реализовать вариант вибростенда кабины и широкого экрана с поворачивающимся проектором, как показано на рис. 4 внизу слева.

Выводы

1. Рассмотрено использование для передачи данных с Simulink на устройства управления и реализация режима реального времени по CAN BUS, который сейчас широко применяется в робототехнике, авиа и автомобилестроении в системах управления для обмена данными между микроконтроллерами, благодаря своей простоте, надежности и скорости.

2. Установлено, что ПО CAN_API.dll, откомпилированное в MVS не работает с TDM-GCC-64 из-за разного подхода в именах функций dll у компилятора MVS. Чтобы устранить эту проблему требуется перекомпилирование dll в среде TDM-GCC-64, которое может выполнить только разработчик dll.

3. Оптимальным выбором для реализации передачи данных с Simulink на стенды с электроактуаторами по CAN BUS является использование адаптеров Titan TITAN ELECTRONICS INC, которые

позволяют реализовать необходимую частоту обмена более 100 Гц для шестиосной платформы.

4. Предложен способ сжатия информации и повышения скорости обмена в 2 раза за счет побайтного занесения двух значений float в поле данных с использованием одинаковых значений идентификаторов объектов управления для двух цилиндров и последующего их разделения в программе микроконтроллеров цилиндров.

5. Разработана программа передачи/обмена данных с Simulink на устройства управления стендов. Оптимальным значением квантования на 6-ти ядерном компьютере является ***chc***=350, обеспечивающее частоту обмена *f*=230 Гц и режим реального времени моделирования.

6. Для обеспечения режим реального времени с CAN BUS необходимо использование более быстродействующего компьютера (CPU ~ 4 ГГц на Windows 10), увеличивающие скорость моделирования в 8,7 раза по сравнению со старым на Windows 7.

Список литературы

1. Mercedes-Benz Innovation Vehicle Developing. – URL: https://www.mercedes-benz.com/en/mercedes-benz/next/advanced-engineering

2. Emanuele Obialero, A Refined Vehicle Dynamics Model for Driving Simulators // Charhalmers University of Technology / Göteborg, Sweden 2013. Master's thesis, P.120.

3. Customized Flight Simulator Car Driving Simulation 6 Dof Motion Base Platform/. - URL: https://szfdra.en.made-in-china.com/product/lsymBGZJbIcn/China

4. Electric Simulation Table. – URL: https://www.moog.com/products/simulation-tables/electric-simulation-table.html/

5. Troubleshooting and Limitations Compiling C/C++ MEX Files with MinGW-w64. – URL: https://nl.mathworks.com/help/matlab/matlab_external/compiling-c-mex-files-with-mingw.html.

6. Михайлов, В.Г. Использование S-Function Builder Matlab/Simulink / В.Г. Михайлов // Системный анализ и прикладная информатика – 2018. - № 4. - С.57-64.

7. Михайлов, В.Г. О некоторых подходах моделирования автомобиля на симуляторах / В.Г. Михайлов // Системный анализ и прикладная информатика – 2019. - № 3. - С.29–35.

8. USB-CAN USER'S MANUAL 2017-07-06 Edition. - URL: https://insat.ru/upload/iblock/da3/titan_USB-CAN%20Manual.pdf

9. Михайлов, В.Г., Получение и использование единого массива продольного профиля и микропрофиля дороги для моделирования ТС / В.Г. Михайлов // Журнал автомобильных инженеров. – 2018. - № 2. - С. 4–7.

10. Михайлов, В.Г. О колебательной модели грузового автомобиля / В.Г. Михайлов, Д.В. Мишута //Автомобильная промышленность. –2016. - №7. - С.23–27.

References

1. URL: https://www.mercedes-benz.com/en/mercedes-benz/next/advanced-engineering

2. Emanuele Obialero, A Refined Vehicle Dynamics Model for Driving Simulators // Charhalmers University of Technology / Göteborg, Sweden 2013. Master's thesis, P.120.

3. URL: https://szfdra.en.made-in-china.com/product/lsymBGZJbIcn/China-

4. URL: https://www.moog.com/products/simulation-tables/electric-simulation-table.html/

5. URL: https://nl.mathworks.com/help/matlab/matlab_external/compiling-c-mex-files-with-mingw.html.

6. Mikhailov V.G. *Sistemnyj analiz i prikladnaja informatika*, 2018, no. 4, pp. 57-64.

7. Mikhailov V.G. *Sistemnyj analiz i prikladnaja informatika*, 2019, no.3. pp. 29–35.

8. URL: https://insat.ru/upload/iblock/da3/titan_USB-CAN%20Manual.pdf

9. Mikhailov V.G. *Zhurnal avtomobil'nyh inzhenerov*, 2018, no. 2, pp. 4–7.

10. Mikhailov V.G., Mishuta D.V. *Avtomobil'naja promyshlennost'*, 2016, no.7, pp. 23–27.

Рецензент: П.И. Поспелов, – д-р техн. наук, проф., МАДИ