iCC 1994

1st international CAN Conference

in Mainz (Germany)

Sponsored by

Allen Bradley
National Semiconductor
Philips Semiconductors

Organized by

CAN in Automation (CiA)
international users and manufacturers group
Am Weichselgarten 26
D-91058 Erlangen
Phone +49-9131-69086-0
Fax +49-9131-69086-79
Email:headquarters@can-cia.de
URL: http://www.can-cia.de

PERFORMANCE ANALYSIS OF CAN BASED SYSTEMS

L. Rauchaupt, Dr.-Ing.

Otto-von-Guericke-University of Magdeburg

ABSTRACT

The acceptance to use the Controller Area Network (CAN) in industrial automation systems has been grown noticeable in the last time. One reason surely is the short reaction time which goes back to the bus access method (CSMA/CA). The disadvantage of this bus access method is, however, that a defined reaction time only may be guarantied for the object with the highest priority. The question to be answered was, wether it is possible to realize message delay times and object cycle times which are acceptable for typical industrial automation systems. This also concerns the message time equidistance. Does the prioritized bus access conflict with the demands on message time equidistance?

The investigations based on conditions of the sensor/actuator area in industrial automation systems. That means a transmission distance of a few hundred meters and a baudrate of 500 kBit/s.

CHARACTERISTIC PARAMETERS OF DYNAMICAL BEHAVIOR

The advantages of serial bus systems in industrial automation systems are well known. On the other hand the serialization of the information causes a notable delay time in comparison to parallel wiring. This delay time depends on the length of the data protocol (frame length), the baudrate and on an overhead time (equation 1).

$$\text{Delay Time} = \frac{\text{Frame Length}}{\text{Baudrate}} + \text{Overhead Time} \qquad \textbf{(1)}$$

The data frame contains information to synchronize, to identify, to control and to save the data flow in addition to the user data. Therefore not only the user data format (net data length) but also the length of these information determines the frame length. The frame length is the most important factor of influence on the delay time.

The baudrate is another important value of the delay time. It depends significant on the transmission distance. The following presentation is based on distances, which are usual in industrial applications of the sensor/actuator area (about 100 m). That means a maximal baudrate of 500 kbit/s concerning the ISO standard 11898 /2/ and the CiA DS 102-1 /2/.

The overhead time contains the software delay time, the controller delay time and the bus access time. In this paper all parts of the delay time and their influences on the minimal cycle time of automized processes and on the variation of the delay time (equidistance) in CAN-based systems are discussed.

DELAY TIME

Theoretical Aspects of the Delay Time

Delay time is called the time from a data change in one bus node up to its registration in another bus node. One part of the delay time is the frame delay time t_T. The frame delay time is determined by the frame length L_T and the baudrate $d_Ü$. Table 1 shows the parts of a CAN Data Frame.

According to the specification the identifier is 11 or 29 bit long. That means that up to 2.032 respectively 536.870.912 data objects can be addressed. Additional to the bits in table 1 one have to consider the stuffbits. A stuffbit is added by the transmitter after 5 consecutive equal bit levels and is rejected by the receiver. This bit has the inverted bit level. It is used to identify a bus idle, to synchronize the CAN controller and to detect transmission errors. The bit stuffing method begins with the Start of Frame and ends with the CRC sequence. That's why the length of CAN frames L_T depends on the data value in contrast to other bus systems.

Meaning	No. of Bits	Comments
Start of Frame (SOF)	1	
Arbitration field	11+1 resp. 29+1	Arbitration field Identifier + RTR (standard resp. extended CAN)
Control field	6	two reserve bits + Data Length Code
Data field	0..64	0 bis 8 Byte
CRC field	15 + 1	CRC sequence + CRC delimiter
Acknowledge field	2	ACK slot + ACK delimiter
End of Frame (EOF)	7	

Table 1: CAN Data Frame

Because of the stuffbit method the delay time of an CAN data frame t_T has a minimal and a maximal value. Equation 2 applies to event driven bus access (CSMA/CA).

$$\frac{19+8\cdot L_D+25}{d_Ü} \le t_T \le \frac{19+8\cdot L_D+25+0{,}2\cdot(19+8\cdot L_D+15)}{d_Ü} \qquad \textbf{(2)}$$

The maximal delay time appears, if after each 5th bit in the bit stream a stuffbit is added. That means a 20% longer delay time in comparison to the minimal value.

Besides the CAN data frame we have also CAN remote frames. In that case a master in a CAN system must send a remote to another bus node to get data. The parts of a remote frame are shown in table 2. After a remote a node sends a data frame as shown in table 1.

Meaning	No. of Bits	Comments
Start of Frame (SOF)	1	
Arbitration field	11+1 resp. 29+1	Arbitration field Identifier + RTR (standard resp. extended CAN)
Control field	6	two reserve bits + Data Length Code
CRC field	15 + 1	CRC sequence + CRC delimiter
Acknowledge field	2	ACK slot + ACK delimiter
End of Frame (EOF)	7	
Intermission	3	

Table 2: CAN Remote Frame

The remote frame contains no data. Thus the delay time to get a data can be calculated with equation 3.

$$\frac{19+25+3+19+8\cdot L_D+25}{d_Ü} \le t_T \le \frac{19+25+0{,}2\cdot(19+15)+3+19+8\cdot L_D+25+0{,}2\cdot(19+8\cdot L_D+15)}{d_Ü} \qquad \textbf{(3)}$$

In most cases the number of CAN objects is enough for industrial applications, if one use the standard identifier (11 bit). Otherwise the frame length is extended by 18 bit, if extended identifier, corresponding to CAN specification 2.0B, are used.

Table 3 gives an overview over the delay times in microseconds for all possible data length. A baudrate of 500 kbit/s is supposed.

	Identifier	Net Data Length [Byte]								
		0	1	2	3	4	5	6	7	8
CSMA	11-bit-ID	88	104	120	136	152	168	184	200	216
min. TL	29-bit-ID	124	140	156	172	188	204	220	236	252
Difference	in %	41%	35%	30%	26%	24%	21%	20%	18%	17%
CSMA	11-bit-ID	102	121	140	159	178	198	217	236	255
max. TL	29-bit-ID	145	164	183	202	222	241	260	279	298
Difference	in %	43%	36%	31%	27%	24%	22%	20%	18%	17%
Remote	11-bit-ID	182	198	214	230	246	262	278	294	310
min. TL	29-bit-ID	254	270	286	302	318	334	350	366	382
Difference	in %	40%	36%	34%	31%	29%	27%	26%	24%	23%
Remote	11-bit-ID	209	228	248	267	286	305	324	344	363
max. TL	29-bit-ID	296	315	334	353	372	392	411	430	449
Difference	in %	41%	38%	35%	32%	30%	28%	27%	25%	24%

Table 3: CAN delay times in µs (500 kbit/s)

The maximal delay times shown in table 3 plus controller and software delay time can be guarantied as the answering time of the object with the highest priority in a CAN system. In contrast to other bus systems this delay time means that an object can be transmitted from any node to any other node in the system. Because CAN is a multi master system, the object has not to pass a central master as necessary in polling systems. Even if one use a remote frame the object can reach every node directly.

The values in table 3 show that the influence of an extended identifier on the delay time is immense. For industrial applications, with net data length up to 4 byte, it means, that the delay time is about 30% greater than with standard identifier. This fact influences also the minimal possible process cyle time and the expected variance of equidistance in CAN-based systems.

Delay Time Measurement

When measuring the delay time we took the software and controller delay time in consideration. We developed a measurement system consisting of a PC with a CAN-Board and software using MS-Windows, CAN bus nodes, an oscilloscope and other hardware /3/.

In our measurement system we used bus nodes with the following microcontroller and CAN-controller:

- microcontroller V25, CAN-controller 82526;
- microcontroller 80C535, CAN-controller 82526;
- microcontroller 80C552, CAN-controller 82527;
- microcontroller 80C592 with integrated CAN-controller;
- universal field bus controller IX0;
- microcontroller 80486, CAN-controller 82C200.

To determine the controller and software delay time within the PC (microcontroller 80486, CAN-controller 82C200) we measured the delay time of the transmit interrupt routine (remote frame) and

the time from the end of a receiving data frame to the registration in the mailbox. The values shown in table 4 were measured by transmitting a 1-byte-object with baudrates of 20 kbit/s and 500 kbit/s.

	Baudrate	Delay Time	Frame Equivalent
PC Transmission	20 kBit/s	550 µs	about 1 Byte
PC Reception	20 kBit/s	510 µs	about 1 Byte
PC Transmission	500 kBit/s	200 µs	about 2 Frames
PC Reception	500 kBit/s	200 µs	about 2 Frames

The frame equivalent in table 4 is the information which could be transmitted during the PC controller and software delay time. That means for a baudrate of 500 kbit/s the delay time in the PC is equivalent to about two data frames. If one use a CAN PC-board without an own microcontroller and with a basic CAN controller, it is possible that information get lost if the bus load is high. That's why you should use CAN PC-boards with an own microcontroller and a dual ported RAM in industrial applications.

In the decentralized bus nodes we measured the following delay time parts:

- **Delay time t1** until the end of the transmit object value actualization (incrementation),
- **Delay time t2** until the registration of all object values to transmit in the CAN controller,
- **Delay time t3** until the start of frame of the first object,
- **Delay time t4** until the reception of the first object in the PC,
- **Delay time t5** until the reception of the last object in the PC.

In Table 5 you see all these delay time parts of different microcontroller and CAN controller with several configuration. The baudrate was 500 kbit/s.

	1*1 Byte					14*1 Byte					5*8 Byte				
	V25	80C535	80C592	80C552	IX0	V25	80C535	80C592	80C552	IX0	V25	80C535	80C592	80C552	IX0
t1	0,128	0,148	0,110	0,160	0,005	0,619	1,446	1,080	1,572	0,026	0,280	0,543	0,405	0,595	0,066
t2	0,360	0,510	0,329	0,456	0,225	2,670	5,788	11,480	5,788	4,580	1,790	4,480	4,667	4,636	3,890
t3	0,323	0,460	0,314	0,432	0,227	1,006	1,820	1,288	1,845	0,251	0,655	1,350	1,056	1,370	0,627
t4	**0,630**	**0,740**	**0,620**	**0,690**	**0,480**	**1,300**	**2,100**	**1,600**	**2,100**	**0,510**	**1,100**	**1,720**	**1,500**	**1,790**	**0,990**
t5						3,380	6,000	11,690	5,960	4,770	2,170	4,810	5,000	4,790	4,060

Table 5: Controller and software delay time in µs with several configuration (500 kBit/s)[1]

The measurement results reflect following factors of influence:

1. Clock frequency
   The clock frequency is 12 MHz for the 80C535, 11,05 MHz for the 80C552 and 20 MHz for the IX0. The internal clock frequency of the 82527 is 10 MHz. The other controller operate at 16 MHz.
2. Processor type
   The V25 is the only 16-bit controller.
3. Communication between microcontroller and CAN-controller
   The IX0 is an universal field bus processor with free capacity for little applications. The 80C592 is using an internal DMA. All other microcontroller exchange the CAN data via the external data bus to the CAN-controller.
4. Object handling
   The 80C592 and the IX0 have an internal basic-CAN-controller. That's why the application processing is interrupted by the CAN-controller after each object transmission.

If one consider the results with lower baudrates, the following conclusions can be made. Controller with an integrated basic-CAN (80592, IX0) make the data very fast available for transmission (internal memory area, DMA). On the other hand they have to manage each object during the transmission. Finally they are slower than the V25/82526.

The number of objects influences the controller delay time more than the data length. This especially applies to the 80535 with a full-CAN. The kind of the controller has nearly no influence on the total delay time if the baudrate is low. Thus the minimal possible process cycle time is determined by the maximal delay time of the data frame and the number of objects (e.g.: 20 kbit/s and 32 nodes -> 32 * 3200 µs = 73,6 ms).

In the case of higher baudrates one bus node can not exceed the bus load because the software delay time is higher than the data frame delay time. That means if there are only a few bus nodes in the network the minimal possible process cycle time is determined by the processing time in the bus nodes to actualized the process values (pre-processing) and make them available for transmission (e. g.: 500 kbit/s and 3 nodes -> about 700 µs). If there are more than about 4 bus nodes the minimal process cycle time is determined by the delay time of the PC and the number of bus nodes (e.g.: 500 kbit/s and 32 nodes -> about 32 * 220 µs = 7,04 ms).

If there are many or long objects, then the application processing of the controller with integrated basic-CAN (80592) is often interrupted by the data transmission. This shows the late entry of the last object (delay time t2).

In addition to the investigations described until now we measured the delay time of an object with low priority, when objects with higher priority are sent. One result of these investigations was, that the bus access time is nearly independent from the priority, if the bus load is less than 50%. The bus idle time is big enough for objects with every priority. With other words objects with every priority are transmitted latest after the running data frame. Using remote frames the delay time of the last polled object is determined, but much greater than using an event driven bus access (CSMA/CA).

PROCESS CYCLE TIME

Not only the delay time describes the dynamic behaviour of a bus system. Industrial processes often are cyclic. That's why there is a demand on cyclic data transmission. We wanted to know whether this is necessary or not. To fulfil the addressed demand it is only necessary to guaranty, that every object can be transmitted in a certain cyclic time. The minimal possible cycle time in the system is determined by the data frame delay time t_T of all objects in the system (equation 4). Here one have to not forget the 3 bit intermission.

$$t_Z = \sum_{i=1}^{ObjAnz} \frac{(L_{T_i} + 3)}{d_Ü} \qquad \textbf{(4)}$$

The minimal process cycle time $t_{Z_{min}}$ in the system has to be greater than the maximal cycle time described by equation 4. If one ignore the controller and software delay time the minimal process cycle time $t_{Z_{min}}$ by event driven transmission (CSMA/CA) fits the equation 5.

$$t_{Z_{min}} \geq \sum_{i=1}^{ObjAnz} \frac{19 + 8 \cdot L_{D_i} + 25 + 0{,}2 \cdot (34 + 8 \cdot L_{D_i}) + 3}{d_Ü} \qquad \textbf{(5)}$$

Using remote frames the minimal possible cycle time increases in comparison with the event driven bus access (CSMA/CA) corresponding to equation 6.

$$t_{Z_{min}} \geq \sum_{i=1}^{ObjAnz} \frac{19 + 25 + 0{,}2 \cdot 34 + 3 + 19 + 8 \cdot L_{D_i} + 25 + 0{,}2 \cdot (34 + 8 \cdot L_{D_i}) + 3}{d_Ü} \qquad \textbf{(6)}$$

The minimal process cycle time for 32 objects all with the same data length and a baudrate of

	Identifier	Net Data Length [Byte]								
		0	1	2	3	4	5	6	7	8
CSMA	**11-bit-ID**	**3,01**	**3,52**	**4,03**	**4,54**	**5,06**	**5,57**	**6,08**	**6,59**	**7,10**
min. TZmin	**29-bit-ID**	4,16	4,67	5,18	5,70	6,21	6,72	7,23	7,74	8,26
CSMA	**11-bit-ID**	**3,29**	**3,90**	**4,52**	**5,13**	**5,75**	**6,36**	**6,98**	**7,59**	**8,20**
max. TZmin	**29-bit-ID**	4,83	5,44	6,05	6,67	7,28	7,90	8,51	9,13	9,74
Remote	**11-bit-ID**	**6,02**	**6,53**	**7,04**	**7,55**	**8,06**	**8,58**	**9,09**	**9,60**	**10,11**
min. TZmin	**29-bit-ID**	8,32	8,83	9,34	9,86	10,37	10,88	11,39	11,90	12,42
Remote	**11-bit-ID**	**6,89**	**7,50**	**8,12**	**8,73**	**9,34**	**9,96**	**10,57**	**11,19**	**11,80**
max. TZmin	**29-bit-ID**	9,65	10,27	10,88	11,49	12,11	12,72	13,34	13,95	14,57

Table 6: Minimal process cycle time of 32 objects (500 kBit/s)

Our investigations have shown that no data get lost if no bus node sends faster than the minimal process cycle time. We considered even bus idle times, since the real data frames are shorter than calculated (less stuffbits). The measurements also show, that not in all cases the data frames are sent in order of the priority, if the bus nodes are started synchronous. The data frame sequence depends on the position of processing in every node, in that moment when the starting edge of the PC interrupts this processing.

Concerning the process cycle time there is no real different between the multi master system CAN and polling systems. In CAN systems one have to ensure that no object sends the data before the minimal process cycle time is over /3/, /4/.

DELAY TIME VARIANCE

Theoretical Aspects of the Delay Time Variance

The variance of the delay time may have two reasons. Firstly it is possible that the bus access time of an object is different during each bus access because of its priority. And secondly the data frame length depends on the data value due to the stuffbits. Following the influence of these reasons is discussed.

If all objects fulfil the demand on the transmission cycle time given in equal 4 the maximal variance of the delay time may be the time delay of one cycle. Depending whether the controller and software delay time is shorter or the data frame delay time, the one or the other delay time determines the value of the delay time variance.

If the data transmission is event driven (CSMA/CA) then it is synchronised to the process. According to the former made remarks changed values can be transmitted immediately, if the bus load is less than 100%. That's why the variance of the delay time is important lesser, than the process cycle time.

In polling systems the changed value can only be sent in fixed time cycles. If the data transmission cycle is not synchronous to the process cycle, it is possible that a changed value can be sent not until a full cycle is completed. That means in the worst case the variance of the delay time can be two cycle times. If one use a full-CAN-controller, it is possible that the CAN-controller is disabled for incoming from the bus data, because the microcontroller has the access right. In this case the value is sent after the following remote.

Stuffbits	0 Byte	1 Byte	2 Byte	3 Byte	4 Byte	5 Byte	6 Byte	7 Byte	8 Byte
0	0,00%	0,00%	0,00%	0,00%	6,53%	5,24%	3,92%	2,30%	2,60%
1	33,12%	33,05%	25,55%	16,31%	21,35%	18,42%	14,98%	9,80%	10,65%
2	43,06%	40,30%	38,53%	32,87%	29,65%	27,99%	25,44%	19,65%	20,42%
3	18,60%	20,20%	24,78%	29,20%	23,93%	25,04%	25,61%	24,36%	24,33%
4	4,97%	5,48%	8,92%	15,20%	12,55%	14,81%	17,45%	21,06%	20,33%
5	0,20%	0,89%	1,93%	5,07%	4,56%	6,12%	8,48%	13,22%	12,56%
6	0,05%	0,08%	0,27%	1,16%	1,17%	1,87%	3,06%	6,35%	6,00%
7		0,01%	0,03%	0,18%	0,22%	0,42%	0,85%	2,35%	2,24%
8				0,02%	0,03%	0,07%	0,18%	0,71%	0,67%
9						0,01%	0,03%	0,16%	0,17%
10							0,01%	0,03%	0,03%

Tafel 7: Number of stuffbits depending on the net data length

In the discussion in chapter Delay Time we supposed the theoretically maximal number of stuffbits for each data frame. To get the right stuffbit influence to the variance of delay time we wanted to know the probability of the maximal number of stuffbits and the average value of the number of stuffbits. We simulated the possible data frames and count the stuffbits. Up to two data bytes it is practicable to simulate all possible data frames. Then the number of possible data frames is so large, that we simulated only about 1.000.000 randomized data frames for each net data length.

Net Data Length	Average No Stuffbits
0 Byte	1,96
1 Byte	2,01
2 Byte	2,24
3 Byte	2,64
4 Byte	2,34
5 Byte	2,54
6 Byte	2,81
7 Byte	3,35
8 Byte	3,27

Tafel 8: Average Number of stuffbits

The values in table **Fehler! Textmarke nicht definiert.** show, that the values shown in table 5 and 6 in practice not occurs. In table 8 you see the average number of stuffbits in CAN data frames, depending on the number of data bytes. The influence of the stuffbit method on the variance of the delay time is small.

Delay Time Variance Measurement

To measure the variance of the delay time 6 nodes sent data frames with a baudrate of 100 kbit/s. The transmission cycle time configured in the node with the lowest priority was equal to the minimal process cycle time. We measured the delay time from a data change in the node with the lowest priority until the registration in the PC. The average and the variance of the delay time based on 500 data frames.

The comparison between the two bus access methods (Table **Fehler! Textmarke nicht definiert.**) shows that the variance of the delay time with an determined access (polling) is important greater than with an event driven bus access

		Net Data Length [Byte]			
		1	2	5	8
Average [ms]	CSMA/CA	3,47	4,13	5,95	7,87
	Polling	7,06	7,69	9,39	11,13
Variance [ms]	CSMA/CA	0,02	0,07	0,06	0,13
	Polling	1,62	1,79	1,98	2,39
Variance [%]	**CSMA/CA**	**1%**	**2%**	**1%**	**2%**
	Polling	**23%**	**23%**	**21%**	**21%**

Table 9: Average and variance at 100% bus load

(CSMA/CA). The variance of the delay time depends on the number of objects, the data length, and. if using high baudrates on the software delay time of the central master.

The measured values with lesser bus loads are shown in Table **Fehler! Textmarke nicht definiert.**.

		Net Data Length [Byte]			
		1	2	5	8
	Bus Load	**54%**	**58%**	**65%**	**71%**
Average [ms]	CSMA/CA	7,08	7,54	9,18	10,99
	Polling	13,05	12,99	14,19	15,77
Variance [ms]	CSMA/CA	0,04	0,13	0,09	0,05
	Polling	1,04	2,89	4,23	5,38
Variance [%]	**CSMA/CA**	**1%**	**2%**	**1%**	**0%**
	Polling	**8%**	**22%**	**30%**	**34%**

Table 10: Average and variance with different bus loads

These results allow following conclusions:

1. Using remote access the delay time variance is greater than using event driven access CSMA/CA.
2. With remote access the delay time variance depends noticeable on the bus load. The more an object is requested the more is the probability, that the value has just changed.
3. Is the transmission cycle time in the neighbourhood of the process cycle time the probability is high that the access to the CAN controller is disabled for incoming data. That means low prior objects can only sent during the next cycle. This is important to the delay time variance.
4. With CAN an adapted equidistance can be realized very simple. By definition of different transmission cycle times data are transmitted only if necessary.

CONCLUSIONS

In systems with high dynamic behaviour CAN is determined, as in other bus systems too, by the software delay time. The software delay time depends on the number of objects and the data length. That's why it is to expect that the dynamic behaviour of the CAN Application Layer depends very strong on the configuration (master, slave, classes, number of objects, profile).

It is possible to design a CAN system which meets the demands of industrial applications. Bus load exceedings only occur, if bus nodes are defect. Using a minimal process cycle time (inhibit time) it is possible to guaranty maximal bus access times for each object. With bus loads less than 50% collisions and bit wise arbitration are an exception. Thus changes of object values can transmit with CAN faster than with polling or token systems. This fact, the longer frames (remote and data), and the fixed access regime using polling cause, that the delay time variance within polling systems is greater than in CAN (CSMA/CA) systems. The influence of stuffbits can be ignored.

LITERATURE

/1/ ISO/DIS 11898: Road vehicles - Interchange of digital information - Controller area network (CAN) for high speed communication, 1992

/2/ CiA DS 102-1: CAN Physical Layer for Industrial Applications, 1992

/3/ Rauchhaupt, L.; Schindler, T.; Beikirch, H.: Bestimmung dynamischer Kenngrößen an CAN-Systemen.
Elektronik plus 6/1993 - Sonderheft CAN, S. 20-24

/4/ Güttler, F.; u. a.: Sensorbusse für den Maschinen- und Anlagenbau - Abschlußbericht zum AIF-Projekt. Otto-von-Guericke Universität Magdeburg, IPE, 1993.