www.asap-firmware.com

Prefazione

Descrizione del Bus CAN
La soluzione ASAP
Esempi in linguaggio C

Titolo: FWS0005A

CAN BUS Monitor

Descrizione del Bus CAN

In questa sezione vogliamo fornire alcune indicazioni generali sulle caratteristiche del Bus CAN, utili per comprendere lo scopo di questa applicazione; per ulteriori informazioni si può visitare il sito CAN in Automation (CiA) www.can-cia.de che raggruppa i produttori e gli utilizzatori a livello internazionale.

Il bus CAN (Controlled Area Network) è un protocollo di comunicazione seriale sincrono che permette di realizzare sistemi di controllo real-time distribuiti garantendo un elevato livello di sicurezza.

E' nato su specifica Bosch per l'ultilizzo nel settore automobilistico, dove le centraline di controllo motore, i sensori, il sistema ABS dell'automobile sono collegati insieme con velocità fino a 1Mbit/s ed elementi meno critici come lampade e finestrini elettrici sono collegati assieme a velocità molto più basse (decine di kbit/s) ma con notevole risparmio sul cablaggio.

La robustezza del bus con distrurbi elettrici e la capacità di rilevazione e rispristino di condizioni di errore, lo rende adatto come bus di campo anche nelle applicazioni industriali.

La massima velocità raggiungibile è di 1Mbit/s con una lunghezza del collegamento di circa 10 metri ma diminuendo la velocità a 50kbit/s si può raggiungere una lunghezza di circa 1000 metri; la trasmissione è differenziale ed è quindi insensibile ai distrurbi EMI.

La codifica del dato binario è di tipo NRZ (Not Return to Zero) dove il livello del segnale rimane costante per tutta la durata del bit; per consentire la sincronizzazione si utilizza il metodo di bit stuffing introducendo un bit di valore complementare dopo cinque bit uguali.

Il livello logico sul bus corrisponde ad un wired AND, dove il livello alto detto *recessivo* può essere sovrascritto da quello basso detto *dominante*.

I nodi sul bus vengono indirizzati utilizzando l'informazione dell'indirizzo contenuta nel messaggio, cosiddetta *identificatore*; quest'ultimo determina anche la priorità del messaggio stesso che sarà più alta tanto più basso sarà il suo valore.

Per il "bus arbritration" viene usata la modalità Carrier Sense Multiple Access/Collision Detection (CSMA/CD) con Non Destructive Arbitration (NDA).

Se ad esempio il nodo A vuole trasmettere un messaggio nella rete, esso prima controlla che il bus sia nello stato di riposo (Carrier Sense); se è così il nodo inizia la trasmissione e diventa il bus master mentre gli altri si mettono in ricezione.

Quando due o più nodi iniziano a trasmettere nello stesso momento la collisione viene evitata utilizzando CD/NDA assieme al meccanismo di wired AND; un nodo che spedisce un bit recessivo ma rileva un bit dominante si predispone automaticamente in ricezione e tenterà la trasmissione appena il bus ritornerà nello stato di riposo.

Le specifiche CAN versione 1.0, 1.2 e 2.0A definiscono la lunghezza dell'identificatore a 11bits (2048 possibili valori) conosciuto come Standard CAN ; la versione 2.0B ha superato questa possibile limitazione permettendo l'utilizzo di messaggi con identificatori sia a 11 che a 29 bit (536 milioni di possibili valori!) conosciuto come Extended CAN.

I controllori CAN possono essere sia integrati nel microcontrollore (periferica on-chip) sia esterni (stand-alone); la tendenza del mercato è verso la soluzione on-chip anche se esistono dei controller esterni stand-alone come Intel (es. 82527), Siemens(es. 82C900, 81C91) e Philips(SJA1000) utilizzati in special modo dalle grandi case automobilistiche.

Un'ulteriore suddivisione può essere fatta per controller Basic e Full CAN; il primo tipo dispone solitamente di un message buffer di trasmissione e uno di ricezione mentre il secondo tipo dispone di 16 message buffer con funzioni di risposta automatica ad una richiesta di trasmissione.

A livello di protocolli ad alto livello troviamo CAL, CAN Open come subset del CAL e DeviceNet.

Livelli di Comunicazione CAN

- Livello Fisico : caratteristiche della connessione e di segnali elettrici
- Livello Collegamento : caratteristiche dei messaggi (oggetti) di comunicazione
- Livello Applicazione : regole di comunicazione specifiche dell'applicazione e definizione del contenuto degli oggetti

Tipi di Frame CAN

- CAN Data Frame : muove i dati dal trasmettitore al ricevitore/i
- CAN Remote Frame : richiesta di trasmissione di un Data Frame
- CAN Error Frame : indica un errore spedendo 6 bits dello stesso valore
- CAN Overload Frame : fornisce un ritardo fra la trasmssione di frame

Formato Frame Standard CAN

- Start : singolo bit dominante
- Identificatore : 11 bit
- RTR bit : Remote Transmission Request bit, nei Data Frame è di tipo dominante
- Campo controllo : 4 bits per lunghezza dati
- Campo Dati : da 0 a 8 bytes
- Campo CRC : 15 bits
- Campo Acknowledge : 2 bits (slot e delimiter), nel trasmettitore sono di tipo recessivo
- Fine del frame : 7 bits di tipo recessivo
- Spazio di interframe …

Gestione degli Errori

- Distinzione tra errori temporanei e permanenti con esclusione automatica dei nodi difettosi
- Bit error : il bit trasmesso è diverso da quello rilevato
- Stuff error : violazione della regola di bit stuffing con più di 5 bit consecutivi uguali
- CRC error : errore tra il CRC a 15bit calcolato e quello ricevuto
- Acknowledgement error : mancata rilevazione livello dominante durante Ack slot
- Format error : violazione durante la trasmissione del formato fisso di alcuni campi

Gestione delle anomalie

Errore attivo : il nodo è ancora presente nella comunicazione e spedisce un active error consistente in 6 bits dominanti

Errore passivo : il nodo è ancora presente nella comunicazione e spedisce un passive error consistente in 6 bits recessivi

Esclusione : il nodo non può né trasmettere né ricevere (Bus Off)

Regole di Bit Stuffing

Numero di bit in un frame = (identificatore + controllo) + (n.dati x 8)
= 34bits + (n.dati x 8)

Calcolo del massimo numero di stuff bits nel caso peggiore :

S = (n –1) : 4 , dove n è il numero di bits nel frame

S = (34 + (n.dati x 8) –1) : 4 = 8 + (2 x n.dati)

Probabilità di mancata rilevazione di messaggi errati

$P < (4.7 \times 10^{-11} \times \text{error rate})$

Esempio :
1 errore ogni 0.7s a 500kbps, 8h / giorno, 365 giorni / anno,
media statistica = 1 errore non rilevato in 1000 anni

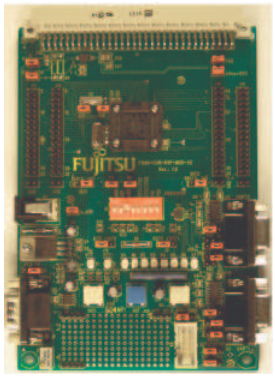
Conclusione

- Protocollo semplice
- Orientato agli oggetti
- Multi master
- Real Time
- Fino a 1Mbit/s
- Comunicazione affidabile
- Standard ISO11898

La soluzione ASAP

Quando si utilizza una rete CAN Bus uno strumento di cui prima o dopo bisogna dotarsi è sicuramente quello che consente di monitorare e analizzare gli oggetti trasmessi sul bus; generalmente più alte sono il bit rate e il numero di oggetti trasmessi, maggiori sono le prestazioni che lo strumento deve garantire, come la velocità di acquisizione e la possibilità di filtrare alcuni identificatori.

Quando si inizia con una rete di medio/piccole dimensioni può comunque essere sufficiente un “CAN Bus monitor” che garantisca di non perdere gli oggetti trasmessi e li visualizzi a schermo su PC.



Ed è proprio questo che è stato realizzato in questa Soluzione, utilizzando come hardware di supporto la scheda di valutazione Fujitsu FLASH-CAN-64P-M09-V02, nella figura a fianco.

Per ulteriori informazioni visitare www.fme.fujitsu.com

La scheda ospita il microcontrollore MB90F497 che dispone di due UART e di un controller CAN 2.0B con 8 message buffer

Dopo aver programmato il microcontrollore con il file “mycanmon.abs” utilizzando il relativo tools in dotazione, è sufficiente collegare il cavo della linea seriale al connettore X3 e il cavo CAN bus al connettore X4.

Come monitor seriale su PC si è utilizzato HyperTerminal di Windows, che si può attivare lanciando il file di configurazione “com1bps19200.ht”.

All’accensione della scheda viene richiesto di immettere il bit rate, selezionabile da 1Mbit/s a 10kbit/s; a questo punto il CAN monitor è già operativo e tramite seriale visualizzerà tutti gli oggetti trasmessi nel seguente formato:

<identificatore>	<R><N >	< data bytes >	<time stamp>

I vari campi sono separati dal carattere punto e virgola (**;**) che si può utilizzare come separatore per il formato CSV in un foglio di calcolo.

- Identificatore = è un valore decimale che indica il numero dell'identificatore contenuto nel messaggio
- R = è presente se il messaggio è di tipo RTR
- N = è il valore del campo data lenght code (DLC) contenuto nel messaggio
- Data Bytes = sono i bytes dei dati in formato esadecimale
- Time stamp = è un contatore progressivo con risoluzione di un millisecondo utile per capire la distanza temporale di trasmissione tra un oggetto e l'altro

Esempi :
Oggetto con identificatore 890 con 6 bytes di dato, avvenuto al millisecondo 13450 :
890; 6; 01 35 03 FE AD 40; 13450

Oggetto tipo RTR con identificatore 1023, richiedente 6 bytes di dato, avvenuto al millisecondo 15700
1023; R6; ; 15700
Nel caso di RTR i bytes di dato non sono presenti
La sigla OVR se presente indica un errore di Overrun in ricezione nel CAN bus message.

Il progetto software è formato da un driver per la seriale UART1 e per il CAN controller.
La ricezione degli oggetti CAN viene eseguita in interrupt utilizzando un buffer circolare.
La trasmissione dei dati al PC avviene nella modalità già indicata nella soluzione ASAP codice FWS0004A, sfruttando la semplicità di formattazione della funzione di libreria "sprintf()" e la velocità di trasmissione in interrupt utilizzando un buffer.
Attualmente non è stata ancora implementata la funzione di flitraggio degli oggetti che sarà prevista in eventuali versioni future.

ASAP

fa parte dei consulenti riconosciuti da Fujitsu Microelectronics Europe

Per ogni ulteriore informazione, ASAP è raggiungibile al seguente indirizzo di posta elettronica :
info@asap-firmware.com

Esempi in linguaggio C

In questa soluzione è stato realizzato un progetto per microcontrollore Fujitsu MB90F497 con Softune C Worbench versione 3.4, contenuto nel file FWS0005A.ZIP.

Esso contiene :

CB90F497.C : modulo gestione periferica CAN Bus on-chip
CBOBJECT.C : modulo gestione oggetti CAN Bus
UART497.C : modulo di gestione UART1
USR.C : modulo utente di configurazione ricezione oggetti CAN Bus
VECTORS.C : modulo gestione vettori di interrupt
MAIN.C : modulo con main loop

La modularità con il quale è stato progettato il software consente il porting anche su microcontrollori di marche diverse.

Definizione valori del baudrate seriale configurabili :

```
enum{_BPS_19200,_BPS_9600,_BPS_4800,_BPS_2400,_NO_BPS};
```

Definizione valori del baudrate CAN Bus configurabili :

```
enum{_1000KBPS,_500KBPS,_250KBPS,_125KBPS,_100KBPS,_50KBPS,_20KBPS,_10KBPS,_MAX_KBPS};
```

Valori del Bit Timing register in funzione del bit rate :

```
CONST _VAR_STATIC _INT16U cb90f497_BusTimeReg[_MAX_KBPS]=
{
/* 1000Kbps: 8TQ, sample a 6TQ(75%), SJW=1 */
0x1401,
/* 800Kbps: 10TQ, sample a 8TQ(80%), SJW=1 */
0x1601,
/* 500Kbps: 16TQ, sample a 14TQ(87,5%), SJW=2 */
0x1c41,
/* 250Kbps: 16TQ, sample a 14TQ(87,5%), SJW=2 */
0x1c43,
/* 125Kbps: 16TQ, sample a 14TQ(87,5%), SJW=2 */
0x1c47,
/* 50Kbps: 16TQ, sample a 14TQ(87,5%), SJW=2 */
0x1c4f,
/* 20Kbps: 16TQ, sample a 14TQ(87,5%), SJW=2 */
0x1c71,
/* 10Kbps: 25TQ, sample a 17TQ(68%), SJW=4 */
0x7fff,
};
```

Ciclo di main loop :

```
……
……
while (1)
    {
    cbobject_Main();

    if (main_TimerSysRead())
        {
        cbobject_Timer();
        }
    }
```