



MÉRÉS:

Az I²C busz alkalmazástechnikája

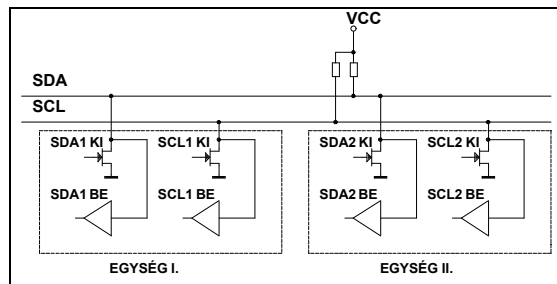
1. A mérés célja:

Az I²C (=Inter IC azaz IC-k közötti) kommunikáció bemutatása, megismertetése és alkalmazása intelligens I²C áramkörök segítségével. A méréshez mellékelt PDF fájlok tartalmazzák a használt áramkörök adatait.

2. A szükséges ismeretek:

2.1 I²C alapok

Az I²C, Inter IC azaz IC-k közötti busz. Az I²C busz nagybonyolultságú integrált áramkörök közötti soros információcsere biztosító, azt fizikailag három vezetékkel megvalósító sínrendszer. Az átviteli fél duplex módon történik, sebessége kb. 100-400 kbit/s-ig növelhető. Az ilyen buszt tartalmazó nagybonyolultságú integrált áramkörök egymással könnyen, kevés vezetékkel tudnak sorosan kommunikálni. A kommunikáció kétirányú adatvonalon (SDA=Serial Data) keresztül történik, és egy külön órajel (SCL=Serial Clock) szinkronizálja az adatvezetékén az adatokat. A busz elvi felépítése az 1. ábrán látható.

1. ábra: I²C kommunikáció elve

A tranzisztorok kikapcsolt állapotában a felhúzó ellenállás miatt, a vonalak magas állapotban vannak. Ez az alaphelyzet. Ha bármelyik tranzisztort bekapcsoljuk, az a vezetékét a földre kapcsolja, így nulla állapotú. Ezt a megoldást az elektronikában huzalozott vagy kapcsolatnak hívják.

A vezérlési elvből következik, hogy mindig csak egy egység vezérelheti az adott vezetékét, a többi egység a tranzisztort nem kapcsolhatja be. Az eddig tárgyalt adás mellett minden egység képes a vonalon lévő adatokat is venni egy erősítőn keresztül.

Még egy érdekes megállapítás: egy egység el tudja dönteni hogy a vezetékét más nem vezérli-e. Ha ugyanis az adatokat a vezetékre kapcsolja, a saját vevőerősítőjén ugyanazt az adatot kell vennie, mint amit kiküldött. Ha ez nem teljesül, valamelyik másik egység is „piszkálja” a vonalat, azaz a buszfoglaltság azonosítható.

A buszon az információáramlás iránya alapján megkülönböztetünk **Adó** ill **Vevő egységeket**. Az átvitel vezérlését a **Master** eszköz végzi, irányítva a **Slave** eszköz működését. Így összesen két funkció és két szerep különböztethető meg:

A funkciók: **TRX** = Transmitter (adó): Az egység amelyik adatot küld a buszra.
RCV = Receiver (vevő): Az egység amelyik adatot fogad a buszról.

A szerepek: **MST** = Master (mester): Az egység amelyik kezdeményezi az átvitelt, az átvitelhez az órajelet generálja, és be is fejezi az átvitelt.
SLV = Slave (szolga): A mester által megcímzett egység.



Egy mikrokontroller I²C egysége mindegyik szerepre és funkcióra képes. Természetesen ha ez az egység vezérli a perifériákat (és a gyakorlatban ez a leggyakoribb eset), akkor szerepe: mester és a perifériák a szolgák.

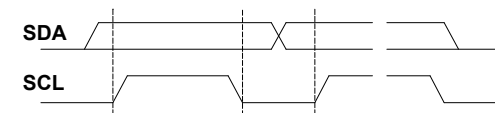
A busz **multi-master** kialakítású. Ez azt jelenti, hogy buszra kapcsolódó eszközök közül nem csak egy, hanem több is átvetheti az átvitel vezérlését. Ezzel kapcsolatos fontos tulajdonság az **arbitration**, vagy döntés. Ez egy eljárás, ami biztosítja, ha egyenél több mester akarja a buszt vezérelni, akkor ezt csak egyetlen egy tudja megtenni, így adatvesztés nem léphet fel.

Az információ átvitel során biteket, illetve ezek csoportját azaz bájtokat viszunk át.

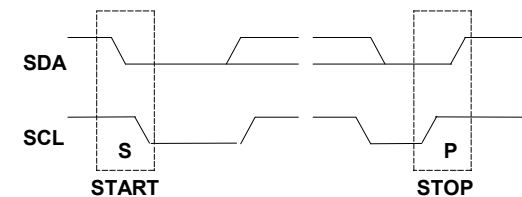
Bit átvitel

A busz két vonala alapállapotban magas szintű. Egy bit átvitele a következő módon zajlik: az eredetileg magas szinten lévő **SDA** vonalra kerül a 0 vagy 1 értéknek megfelelő 0 ill, 5V feszültség szint. Az **SCL** vonal magas szintje alatt érvényes az adat. Az adat csak az **SCL** vonal alacsony szintje alatt változhat.

ADAT stabil



BIT átvitel az I2C buszon



START és STOP feltételek

2. ábra: Bit átvitel, START és STOP feltétel az I²C buszon

A busz aktív és inaktív állapotát a **START** és **STOP** feltételekkel tudjuk definiálni.

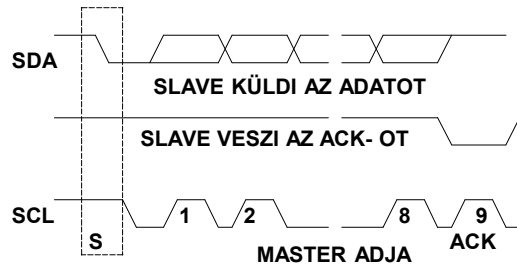
- START** feltétel akkor lép fel és a busz aktív lesz amikor **SCL** magas állapotában az **SDA** vonalon egy H-L átmenet van.
- STOP** feltétel akkor lép fel, amikor **SCL** magas állapotában az **SDA** vonalon egy L-H átmenet van.

A **START** és **STOP** állapotokat (és az órajelet) csak a mester generálhatja. A busz aktív a **START** és **STOP** állapot között. Ezután válik a busz szabaddá.

Bájt átvitel

Az **SDA** vonalon sorosan átvitt adat mindegyike 1 bájt = 8 bit hosszúságú. Az átvitt bájtok száma nincs korlátozva. Az adónak a vevő minden bájt vételét egy **L szintű nyugtázó (ACK = acknowledge) bit** küldésével igazolja. Az ehhez szükséges órajelet a mester generálja, az adó az **SDA** vonalat elengedi. A vevőnek ekkor az **ACK** generáláshoz le kell húznia az **SDA** vonalat. Az átvitel a legmagasabb helyiértékű (**MSB**) bittel kezdődik.

Az adatbiteket az adó, az **ACK** bitet a vevő küldi. Ha egy vevő nem képes egy adatot venni akkor az **ACK** bit küldése helyett az **SCL** vonalat 0 szinten tartja. Ez egy várakozó állapot. Vegyük észre, hogy az adat- és az órajel vonalat az adó és a vevő felváltva használja.

3. ábra: Bájtt átvitel az I²C buszon

A "minden bájtt nyugtázása" szabály alól két kivétel van:

- Az egyik akkor lép fel, ha a mester a vevő (**MST/RCV**). Ilyenkor az adónak valahogy jelezni kell az adatbájtt sorozat végét, a küldőnek nem adva **ACK**-ot. Az **ACK** jelhez kapcsolódó órajelet a mester természetesen generálja, de az **SDA** vonalat nem húzza le L szintre. Ezt hívják negatív nyugtázásnak (**NACK**).
- A másik kivétel: a szolga akkor nem küld **ACK** jelet, ha nem képes újabb adatbájtokat elfogadni. Ez akkor lép fel, ha olyan átvitelt kezdeményezünk, amit nem képes fogadni.

Adatforgalom a buszon

A buszon lévő minden eszköznek saját címe van. Mielőtt adatátvitel történne a buszon, a mester **START** állapotba hozza a buszt, majd kiadja a buszra a szolga címét, amellyel adatot akar cserélni. Az a szolga, amelyik felismeri a saját címét, **ACK** jelet küld vissza. A címzést a mester végzi közvetlenül **START** állapot után. Ez az első küldött bájtt.

A cím hét bites. A nyolcadik, a legkisebb bit dönti el a szolgálal történő adatcsere irányát. 0.bit jelöli az írást, ilyenkor a mester küld adatokat (W), 1 értékű bit pedig az olvasást (R).

A buszra kapcsolódó eszközök címei két kategóriába sorolhatók: Az egyik kategóriában a cím programozható, ezek általában a mikrokontrollerek. A másik kategóriát a különféle funkciókat megvalósító periféria áramkörök alkotják.

Ezeknél az eszközök címe két részből tevődik össze: egy típus címből (4 bit), amit a gyártók osztanak ki az eszközök számára és egy hardver címből (3 bit) amely az címző lábainak 0-ba ill. 1-be kötésével állítható. A típus cím az azonos (típusú) tokoknál mindig megegyezik. Ezzel a címmel jelentkezik be a slave eszköz ill. ezzel a címmel szólítja meg a master eszköz a slave-et adatcsere előtt.

Például az EEPROM eszközök típuscíme 1010 azaz AH. Ha egy ilyen EEPROM három címző lábát (A0-A2) 0-ra kötjük, akkor az eszköz írási címe 1010 000 0 = A0H, és az olvasási címe 1010 000 1 = A1H lesz.

Tekintsük át az egy mester — több szolga struktúrájú buszon zajló adatátvitelt! **M** jelöli a mester által-, **L** a szolga által küldött adatbiteket

MASTER WRITE



MASTER WRITE

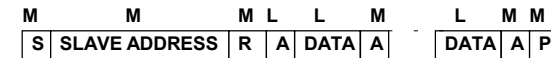
**S-START P-STOP R-READ W-WRITE M-MASTER L-SLAVE
A-ACK NA-NEG. ACK DATA-ADATOK**

A mester **START** állapotba hozza a buszt (S) és kiküldi a szolga címét. A cím legkisebb helyiértékű bite W=0. Ezt a szolga az **ACK** jel visszaküldésével igazolja (A). Ezek után a mester küldi az adatokat a szolgálalnak



(DATA), és az minden bájtt vételét (A) küldésével igazolja. Az utolsó adat küldése után a mester **STOP** állapotba hozza a buszt (P).

MASTER READ



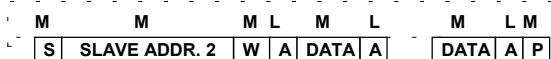
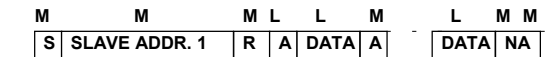
MASTER READ

A mester **START** állapotba hozza a buszt (S) és kiküldi a szolga címét. A cím legkisebb helyiértékű bite R=1. Ezt a szolga az **ACK** jel visszaküldésével igazolja (A). Ezek után a mester fogadja az adatokat a szolgálal (DATA), és minden bájtt vételét (A) küldésével igazolja. Az utolsó adat küldését a mester negatív nyugtázással jelzi (NA) Ezek után a mester **STOP** állapotba hozza a buszt (P).

A fenti esetben a mester minden átvitelnél a buszt újból nyitja és zárja.

Amennyiben a buszon több szolgálal akar a mester adatot cserélni, a minden átvitelt lezáró **STOP**, majd az indító újabb **START** állapot sokat lassít az átvitel. Ilyenkor használható az ismételt **START** állapot generálása. Ez az jelenti, hogy az átviteleket nem **STOP** (P) hanem a következőt indító **START** (S) állapottal fejezzük be, azaz a mester a buszt folyamatosan használja.

Ismételt START állapot használata



ISMÉTELT START ÁLLAPOT

Egy olyan esetet látható, amikor először a mester adatokat kapott egy szolgálal, majd utána adatokat küld nem szükségképpen azonos című szolgálalnak.

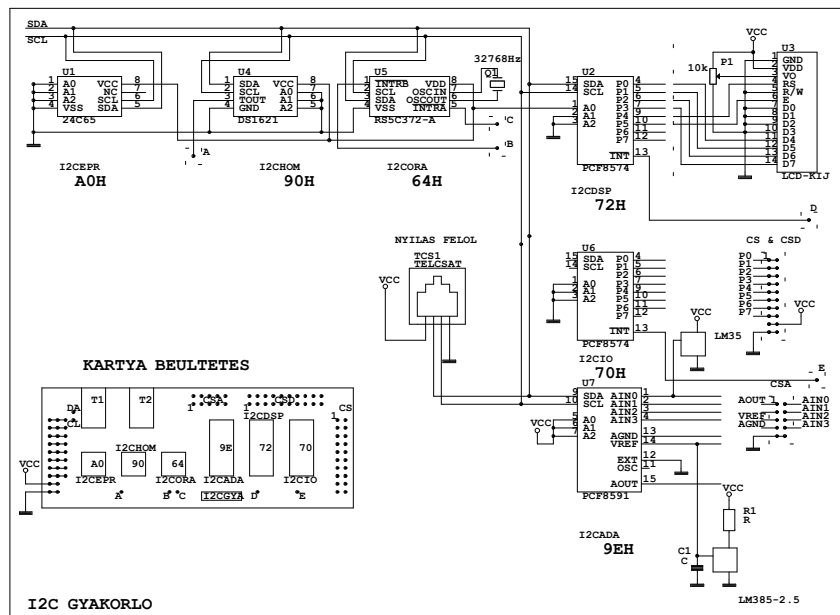
A fenti adatátviteli protokollokat két módon lehet megvalósítani: vagy beépített hardver segítségével (ilyeneket tartalmaznak az erre felkészített mikrokontrollerek és az I²C buszra kifejlesztett periféria áramkörök), vagy megvalósítására szoftver "bit-billegtetéses" programot írhatunk. A fent ismertetett adatátviteli változatokat az **66. ábrán** foglaltuk össze. Az M illetve L betűk azt jelzik, hogy az adott bájtos a mester vagy a szolga küldi.

2.2 Az I²C gyakorló panel ismertetése

A méréshez egy olyan panelt használunk, amely több, a leggyakrabban használt I²C buszos eszközt tartalmaz.

Megnevezés	Cím (HEX)	Megjegyzés
I2CEPR – 256 bájtos EEPROM	A0H	Microchip 24LC01 típus
I2CHOM – DS1621 hőmérő	90H	Magát a tok hőmérsékletét méri –55+127 fokok között
I2CORA – RS5C372 – RICOH óra	64H	
I2CDSP – PCF8574 8 I/O	72H	
I2CIO - PCF8574 8 I/O	70H	
I2CADA – PCF8591	9EH	

A kapcsolási rajz alapján az egyes elemek jól azonosíthatók. A tokok programozását a részletes adatlapok tartalmazzák (ezek PDF formátumban rendelkezésre állnak), mi a következőkben csupán a programozásukhoz szükséges legfontosabb ismereteket közöljük.



I2CEPR

Az EEPROM-ok kezelése egyszerű : íráskor a vezérlőbájt kiadása után ki kell adni az írni kívánt memóriarekesz címét (ha a tok 256 bájtjánál nagyobb kapacitású, akkor ez két bájt HI és LO sorrendben) , majd az írandó adatot. Ne feledjük, hogy a tokoknál 5-10 msec szükséges egy bájt megírására. Lehetőség van egyserre 16 (nagyobb kapacitású tokoknál 64) bájt (page) egyszerre történő írására is.

Olvasás a tokból háromféle módon történhet: az előző írásnál vagy olvasásnál beállított címmutató által meghatározott helyről (current address read), adott címről (random read), több egymás utáni bájt kiolvasása (sequential read). Ilvenkor a mester az olvasás végét NACK feltétel generálásával jelzi.

A következő diagrammok a tok kezelést mutatják be.

FIGURE 8-1: BYTE WRITE

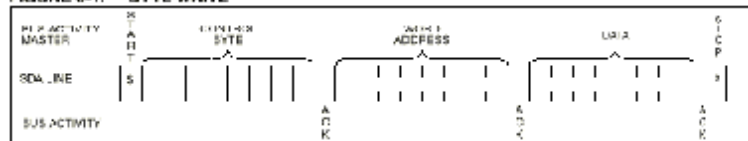


FIGURE 6-2: PAGE WRITE

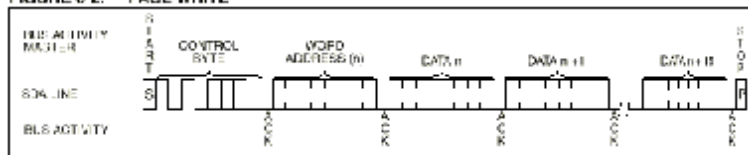


FIGURE 8-1: CURRENT ADDRESS READ

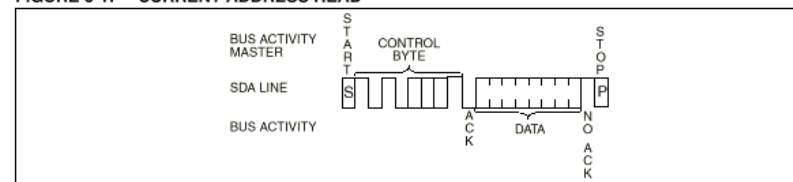


FIGURE 8-2: RANDOM READ

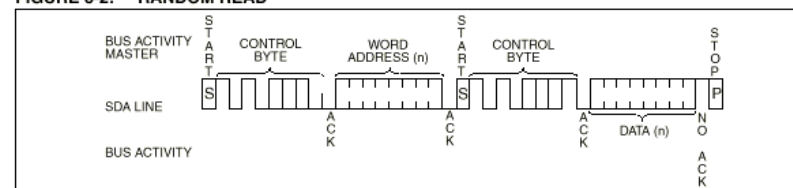
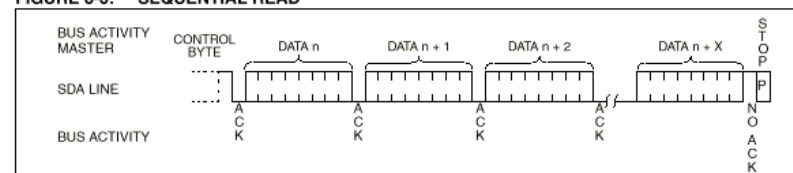


FIGURE 8-3: SEQUENTIAL READ



I2CHOM

Nagyon sok műszaki feladat megoldása során van szükség hőmérséklet mérésére. A mérési tartomány, és a pontosság az a két legfontosabb paraméter, amit ezeknél a mérésnek figyelembe kell venni. Amennyiben a mérési tartomány a „biológia létünk” által meghatározott ($-55^{\circ}\text{C} \dots +125^{\circ}\text{C}$ között), akkor kétségtelenül az egyik legjobb választás a DALLAS cég hőmérsékletmérő integrált áramkörének az alkalmazása. Ezeknél az érzékelő maga az integrált áramkör. A mérési eredményeket a tok bináris adatsorozat formájában küldi el a feldolgozó egységnek. A tokok további kivezetéseivel megvalósíthatók a hőmérsékletméréssel kapcsolatos szabályozási, jelzési feladatok: hőmérséklet állandó értéken tartása (termosztát), jelzésadás a beállított maximális illetve minimális hőmérséklet túllépésekor.

A DS1621 jelű tok mérési pontossága $\pm 0.5\text{ }^{\circ}\text{C}$ fok, amely szoftveresen $\pm 0.1\text{ }^{\circ}\text{C}$ –ra növelhető. A 9 bites mérési eredményt két egymás utáni bájtban olvashatjuk ki a tokból, a második bájt csupán az $0.5\text{ }^{\circ}\text{C}$ -ot tartalmazza.

TEMPERATURE/DATA RELATIONSHIPS Table 2

TEMPERATURE	DIGITAL OUTPUT (Binary)	DIGITAL OUTPUT (Hex)
+125°C	0111101 00000000	7B00h
+25°C	0001001 00000000	1900h
+1/2°C	0000000 10000000	0080h
+0°C	0000000 00000000	0000h
-1/2°C	1111111 10000000	FF80h
-25°C	1100111 00000000	E700h
-55°C	1001001 00000000	C800h

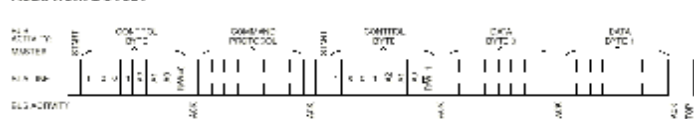
Amint látható, a negatív értékek ábrázolása kettes komplement formában történik.
Mivel a tok belső regiszterei is elérhetők, ezért a következő ábrák mutatják a tok írását és olvasását.



Write to DS1621



Read from DS1621



A kommunikáció mindig a cím után kiadott paranccsal kezdődik, a parancsok listája:

KONFIGURÁCIÓ/STÁTUSZ REGISZTER

DONE	THF	TLF	NVB	1	0	POL	1SHOT
------	-----	-----	-----	---	---	-----	-------

DS1621 COMMAND SET Tábla 3

INSTRUCTION	DESCRIPTION	PROTOCOL	2-WIRE BUS DATA AFTER ISSUING PROTOCOL	NOTES
TEMPERATURE CONVERSION COMMANDS				
Read Temperature	Reads last converted temperature value from temperature register	A4h	<read 2 bytes data>	
Read Counter	Reads value of count remaining from counter	A8h	<read data>	
Read Slope	Reads value of the slope accumulator	A0h	<read data>	
Start Convert T	Initiates temperature conversion	EEh	idle	1
Stop Convert T	Halts temperature conversion	D2h	idle	1
REGISTER AT COMMANDS				
Access TH	Reads or writes high temperature limit value into TH register	A4h	<write data>	2
Access TL	Reads or writes low temperature limit value into TL register	A0h	<write data>	2
Access Config	Reads or writes configuration code to configuration register	ACh	<write data>	2

A tokot a megfelelő működés miatt konfigurálni kell: azaz a CONFIG regiszternek megfelelő bitjeit be kell állítani:
DONE: 0 – konverzió folyamatban, 1 – konv. kész.
THF,TLF: hőmérséklet magas, alacsony bit 1-ha a határt túllépjük
NVB: 1 memóriairás aktív bit,
POL: a kimeneti bitek aktív szintje állítható be,
1SHOT: ha 0 akkor folyamatosan mér, ha 1 akkor csak a Start Convert T parancs hatására.

I2CORA

Az óra lényegében egy regiszterekből álló számláló lánc. 16 belső regisztert tartalmaz, ezek tartalmát írhatjuk és olvashatjuk. Az értékek BCD kódban vannak ábrázolva. A műveletek vezérlése a kontroller által kiküldött vezérlő szóval történik. A mérésnél használt regiszterek:

Internal address					Contents	Data*															
A7	A6	A5	A4	A3		D7	D6	D5	D4	D3	D2	D1	D0								
0	0	0	0	0	Second counter	—	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11			
1	0	0	0	1	Minute counter	—	M0	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11			
2	0	0	1	0	Hour counter	—	—	12/24 P/A	H0	H1	H2	H3	H4	H5	H6	H7	H8	H9			
3	0	0	1	1	Day of the week counter	—	—	—	—	—	W0	W1	W2	W3	W4	W5	W6	W7			
4	0	1	0	0	Day counter	—	—	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10			
5	0	1	0	1	Month counter	—	—	—	MO0	MO1	MO2	MO3	MO4	MO5	MO6	MO7	MO8	MO9			
6	0	1	1	0	Year counter	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7	Y8	Y9	Y10	Y11	Y12			
7	0	1	1	1	Time trimming register	—	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11			
8	1	0	0	0	Alarm_A (minute register)	—	AM0	AM1	AM2	AM3	AM4	AM5	AM6	AM7	AM8	AM9	AM10	AM11			

Két riasztó, illetve periodikus megszakítás (nyitott drain-ű) kimenettel rendelkezik: ezen kimenetek konfigurálható módon periodikus megszakítás, vagy beállított időpontban (hét valamelyik, vagy több napján,



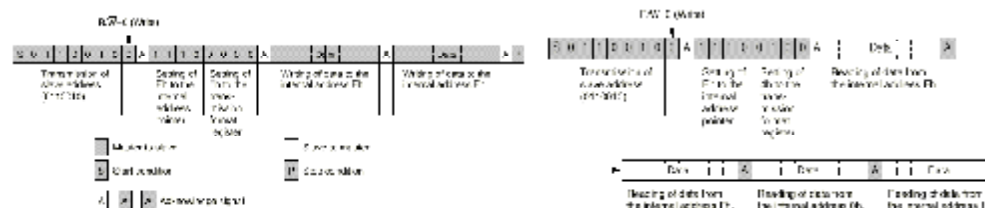
adott óra adott percében) bekövetkező riasztás céljaira használhatók. Az óra címe: 0110 010K. Ha K=0, akkor írási parancs, ha 1, akkor olvasási parancs következik.

A tokba történő íráskor a start feltétel, és az írási cím (0110 0100) megadása után a következő bájtnban meg kell adni az első írandó regiszter 4 bites címét, és a 4 bites átviteli formátumot, ami írásnál mindig csak 0000 lehet. Ezek után sorban egymás után elküldhetők az egymást követő című regiszterekbe írandó bájtok, amit a stop feltétel fejez be. A következő kis pszeudókódreszletek a tok írását ill. olvasását mutatják be:

ÍRÁS A TOKBA:		OLVASÁS A TOKBÓL:	
I2cstart	'I2C startfeltétel generálása	I2cstart	'start feltétel
I2cwrite &H64	'cím kiküldése	I2cwrite &H64	'RS5C372A írási címe
I2cwrite Dum	'0 = 0 kezdőregisztercímű, '0-ás átviteli formátum	I2cwrite &H04	'első kiolvasandó regiszter
I2cwrite S	'másodperc beírása az óra regiszterébe	I2cwrite S , Ack	'másodperc olvasása
I2cwrite M	'percek	I2cwrite M , Ack	'percek
I2cwrite H	'órák	I2cwrite H , Ack	'órák
I2cwrite DW	'a hét hányadik napja	I2cwrite W , Ack	'hét napja
I2cwrite D	'napok	I2cwrite D , Ack	'napok
I2cwrite Month	'hónapok	I2cwrite Month , Ack	'hónapok
I2cwrite Year	'évek	I2cwrite Year , Nack	'évek
I2cstop	'I2C stopfeltétel generálása	I2cstop	'I2C stopfeltétel generálása

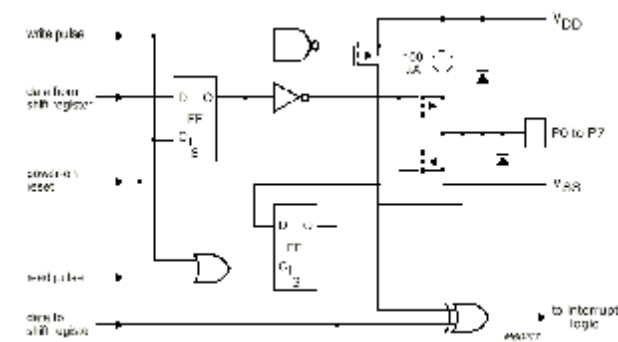
A használt I2cstart, I2cstop, I2cwrite, I2cwrite rutinok az I²C szokványos eljárásai, S,M,...bájtos változók.

Example of data writing (all data writing is similar with the I2C to DS1621)



I2CIO, I2CDISP

Mindkét egység a PCF 8574 8 bites I/O áramkörön alapul. Programozáskor írási paranccsal a 8 kimenetre tudjuk írni a kiküldött adatbájtot, illetve olvasási paranccsal a tok kivezetéseit olvassuk be. A kivezetések áramköri kialakítása a következő:

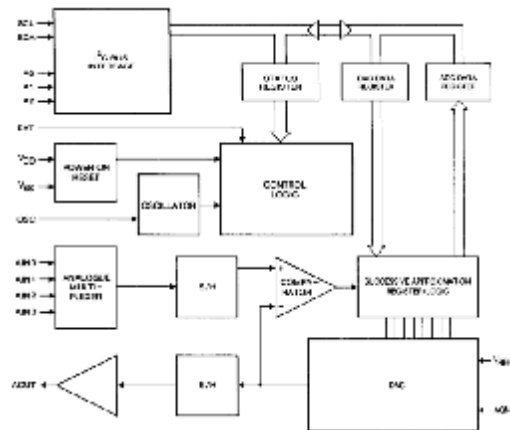


A gyakorló panelen azért van két ilyen tok, mert az egyik tok felhasználásával alfanumerikus LCD kijelző illesztése is lehetőség nyílik. Amint látható, a kimenetek nyitott drain kialakításúak.

I2CADA

A tok 4 multiplexelt analóg bemeneti csatornát és egy analóg kimeneti csatornát tartalmaz. A tok felépítését a blokkvázlat mutatja.

A tok programozásakor egy írási paranccsal ki kell küldenünk a konfigurációs bájtot. Az ez után küldött bájtoknak megfelelő analóg értékek a D/A kimenetre kerülnek. A tok olvasásakor a legutolsó konverzió eredményét lehet beolvasni. A részletek az ábráról olvashatók le.



KONFIGURÁCIÓS BÁJT:

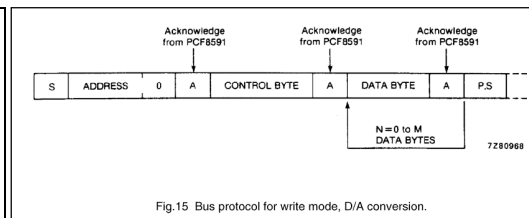
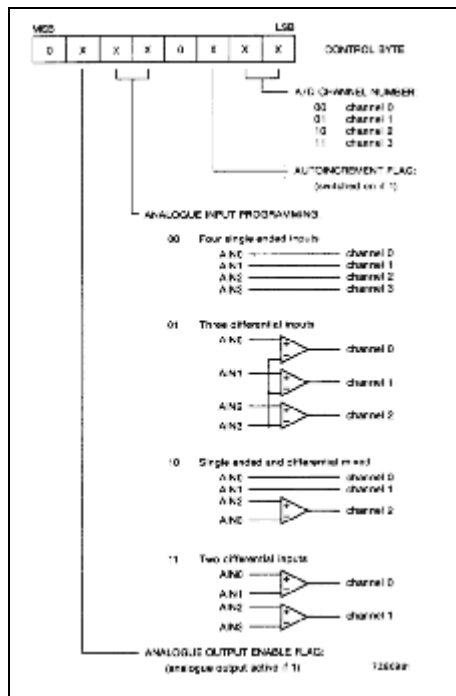


Fig.15 Bus protocol for write mode, D/A conversion

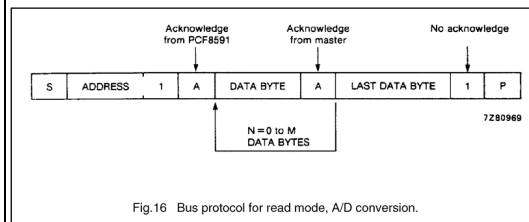


Fig.16 Bus protocol for read mode, A/D conversion.

Az I²C adatátviteli protokollokat két módon lehet megvalósítani: vagy beépített hardver segítségével (ilyeneket tartalmaznak az erre felkészített mikrokontrollerek pl. (Pl. a Phillips 80C552 tokja) és az I²C buszra kifejlesztett periféria áramkörök), vagy megvalósítására szoftver "bit-billegetéses" programot írhatunk.

3. Házi feladatok:

31. Írja meg az I²C protokoll **start_i2c**, **stop_i2c**, **send_i2c_byte**, **get_i2c_byte** alaprutinait. Először készítsen blokkvázlatokat!
32. Készítse el egy soros eeprom kezelő program I²C kommunikációt alaprutinok formájában tartalmazó blokkvázlatát!
33. Írjon programot amellyel a PC soros portjának felhasználásával (TERMINAL) az órát felprogramozza, illetve, a tartalmát kiolvassa! Készítse el a program blokkvázlatát!
34. Írjon programot, amely a hőmérő tartalmát a soros vonalon keresztül kiolvassa, és a képernyőn megjeleníti!

4. Mérési feladatok:

41. Próbálja ki a 32. pontban közölt programot! A panelen kösse össze a megfelelő pontokat, programozza fel a PIC-et!
42. Próbálja ki a 33. pontban közölt soros programot! A panelen kösse össze a megfelelő pontokat, programozza fel a PIC-et!
43. A 34. pontban megadott házi feladatot programozza be és próbálja ki!

5. Kérdések

51. Hogyan működik az I²C busz? Milyen szerepek és funkciók vannak? Mutassa be a START, STOP állapotokat, és egy bit átvitelét!
52. Hogyan történik a bájtok átvitele a buszon? Hogyan történik a bájtok nyugtázása? Mire szolgál Mi a negatív nyugtázás és mire szolgál?
53. Mikor és miért használjuk az ismételt START állapotot?
54. Milyen írási és olvasási módjai vannak egy I²C buszos eepromnak?
55. Ismertesse a hőmérsékletmérő IC-vel történő kommunikációt!
56. Hogyan működik a 8 bites I/O I²C buszos áramkör?
57. Ismertesse az A/D és D/A konverziót megvalósító I²C buszos áramkört!