Università degli Studi dell'Aquila

Facoltà di Ingegneria

GSMBOX

Sistemi Embedded – A.A. 2008/09

Docente

Prof. Luigi Pomante

Studente

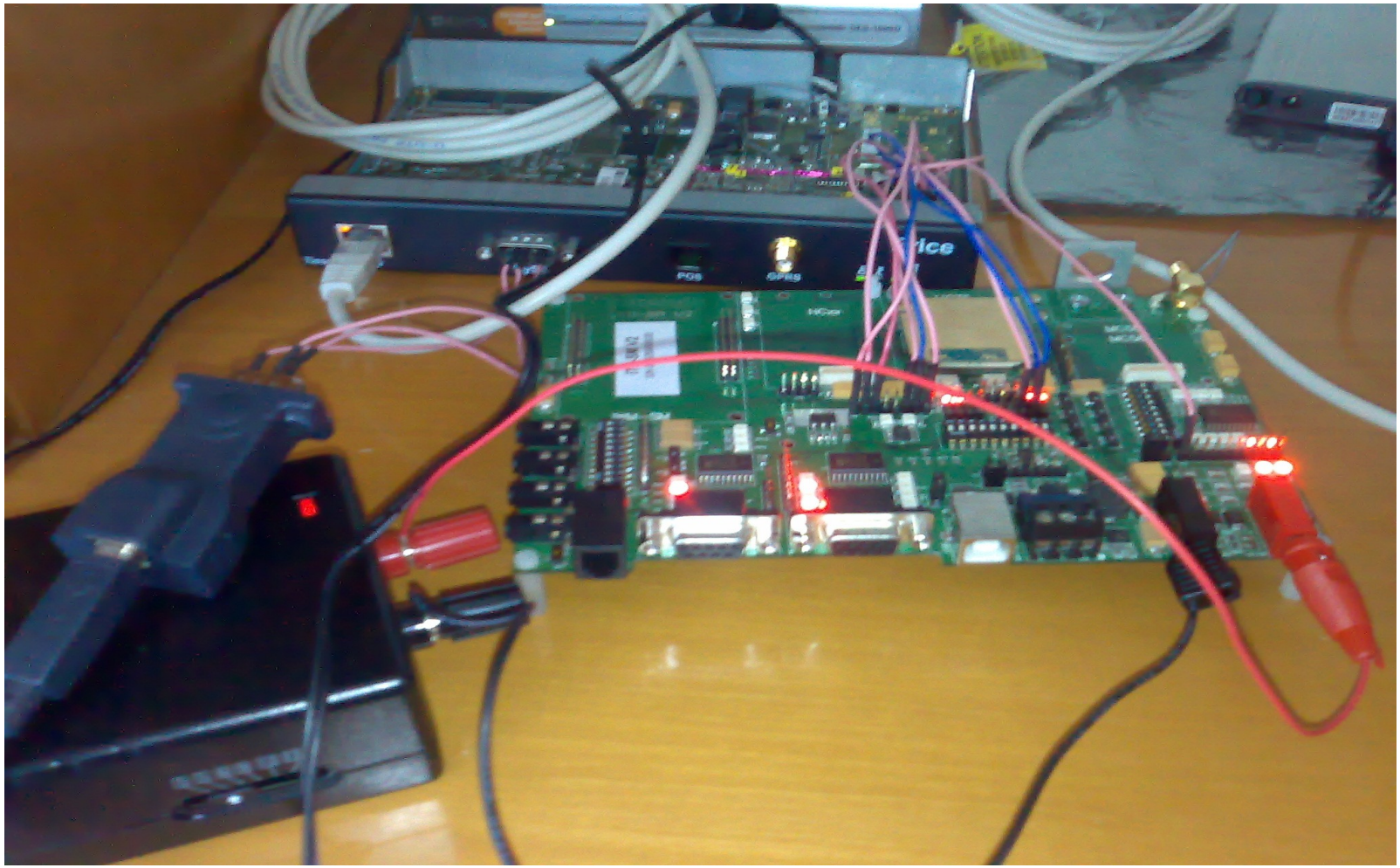
Davide Di Gesualdo

GSMBOX: l'idea

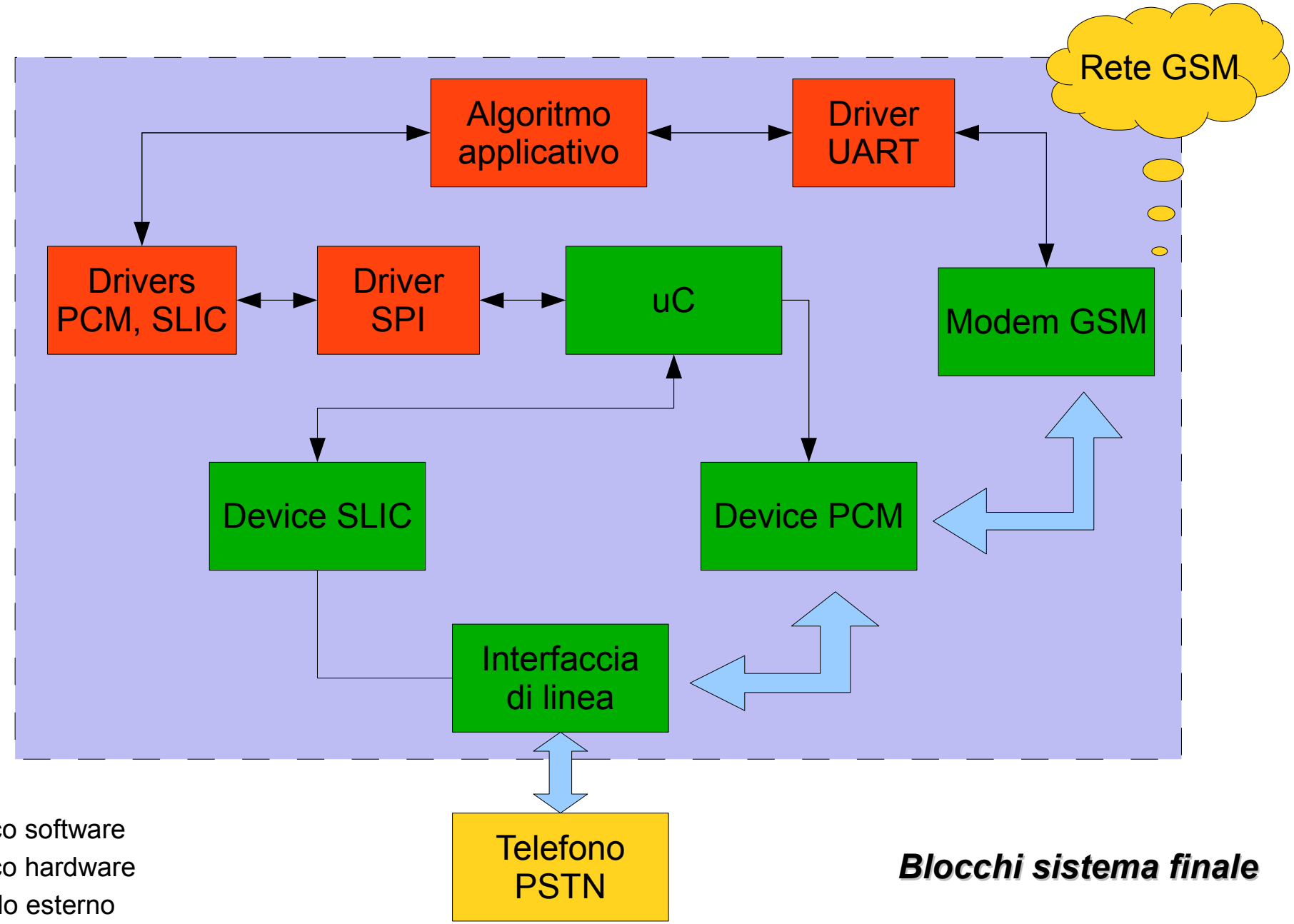
- Realizzazione di un sistema di interfacciamento tra il mondo PSTN e il mondo GSM (gateway PSTN – GSM)
  - Segnalazione PSTN
  - Generazione toni
  - Generazione e decodifica di DTMF
  - Gestione chiamate incoming e outgoing

- Utilizzo indipendente o integrato in PBX

Prodotto finito (?)



Schema a blocchi



- Blocco software
- Blocco hardware
- Mondo esterno

Blocchi sistema finale

Strategia di progetto

- Scelta componenti hardware
- Scelta componenti software
- Interfacciamento componenti hw su development board
- Progettazione macchina a stati applicativa

- Implementazione macchina a stati // prototipazione (schematici, sbroglio, realizzazione)
- Test

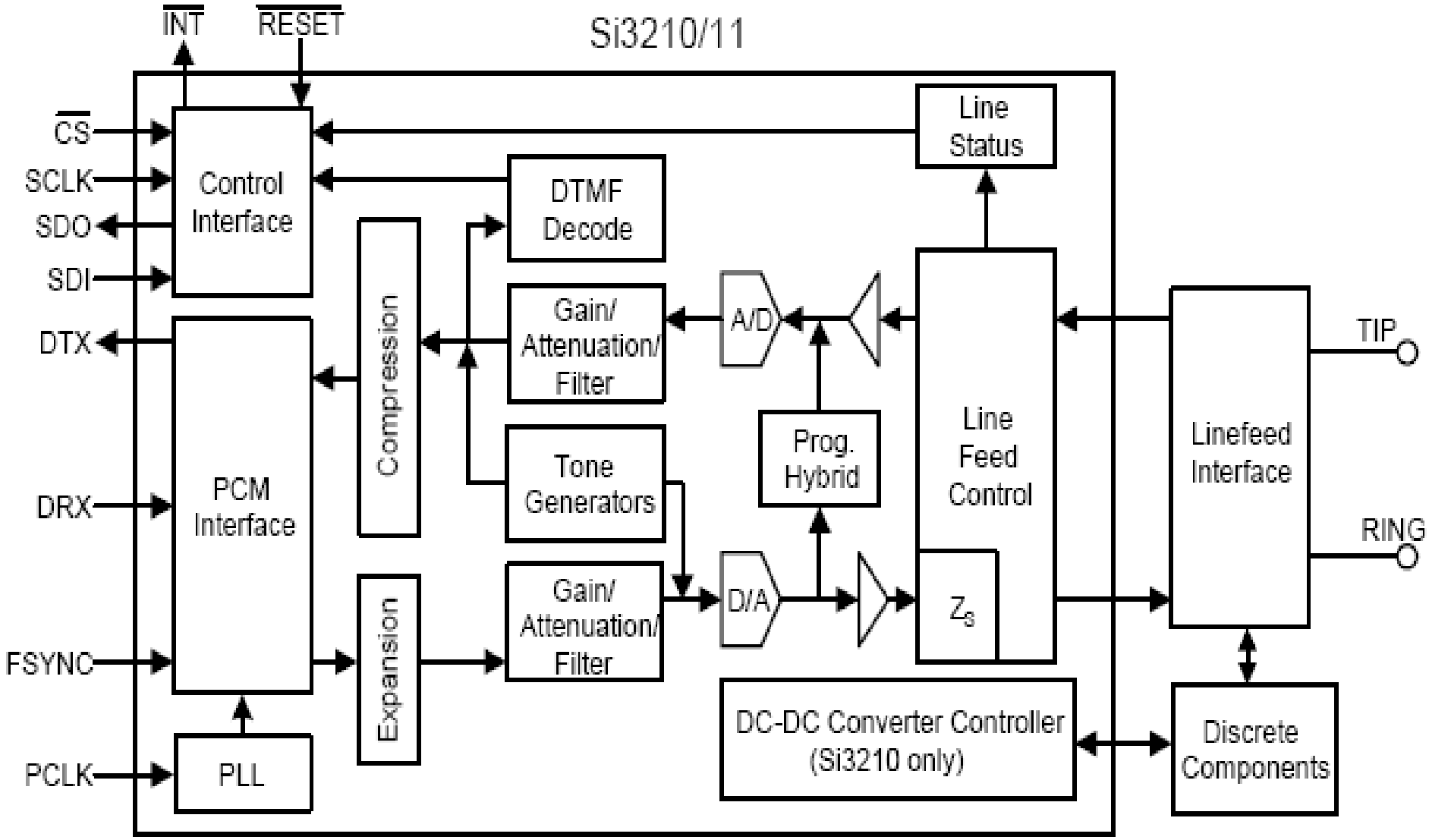
Scelta componenti hardware

- Interfaccia di linea: Si3201 (Silicon labs)
- *Device PCM + SLIC: Si3210 (Silicon labs)*
- *MicroControllore + modem GSM: TC65i (Cinterion)*
- *Development board: iTB-SM V2 (Intesyn)*
- *Logica programmabile (PLD/FPGA)*

Si3210 – cos'è

- Dispositivo per interfaccia digitale PCM (Pulse Code Modulation)
- ProSLIC (Programmable Subscriber Line Integrated Circuit), programmabile tramite SPI
- Codec (lineare 8bit, lineare 16bit, G-711u-law o A-law)
- Generatore tensioni batteria (DC-DC)

Si3210 – schema a blocchi



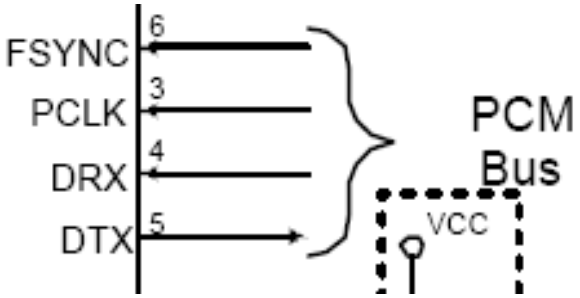
Si3210 – funzionalità

- Interfaccia di linea
- DC-DC converter per tensione di alimentazione linea (programmabile fino a 94.5 VDC)
- Generatore di toni: due oscillatori programmabili con i relativi timer
- Generazione ring: oscillatore a due poli programmabile con relativo timer
- Cattura DTMF (segnalazione in banda)
- Compendion CODECs u-law e A-law
- Interrupt capabilities

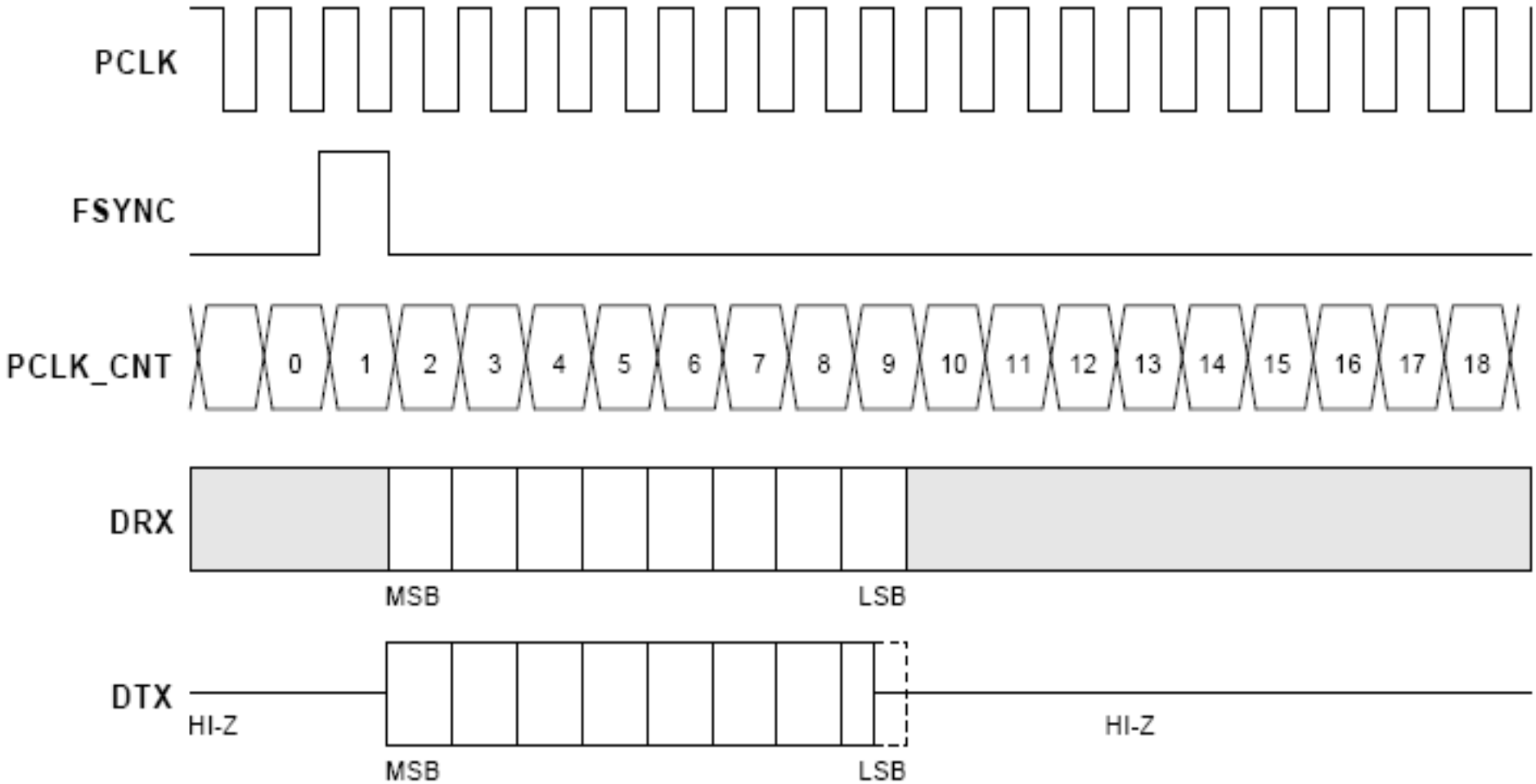
Si3210 – bus PCM (1)

- Quattro segnali utilizzati:
  - PCLK: PCM clock
  - FSYNC: frame sync, segnale di sincronizzazione per l'inizio del campionamento dei dati
  - DTX: data transmit, segnale di trasmissione dati
  - DRX: data receive, segnale di ricezione dati

- Parametri configurabili: startbit, frequenza, tristate in RX, FSYNC corto o lungo, 8 o 16 campioni acquisiti

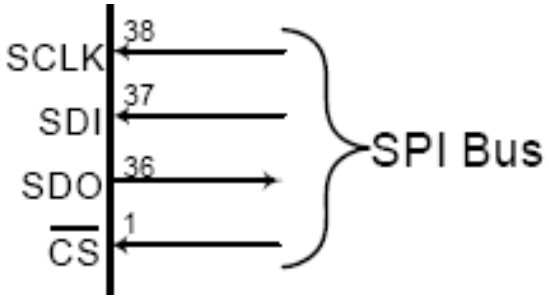


Si3210 – bus PCM (2)

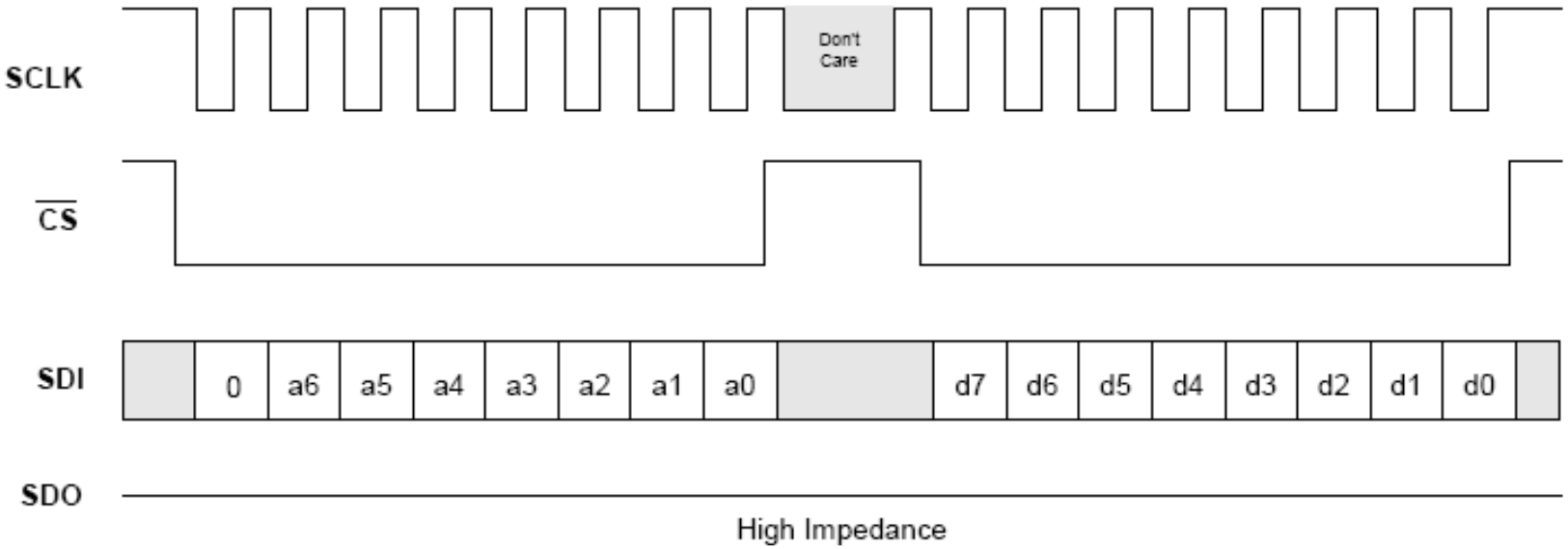


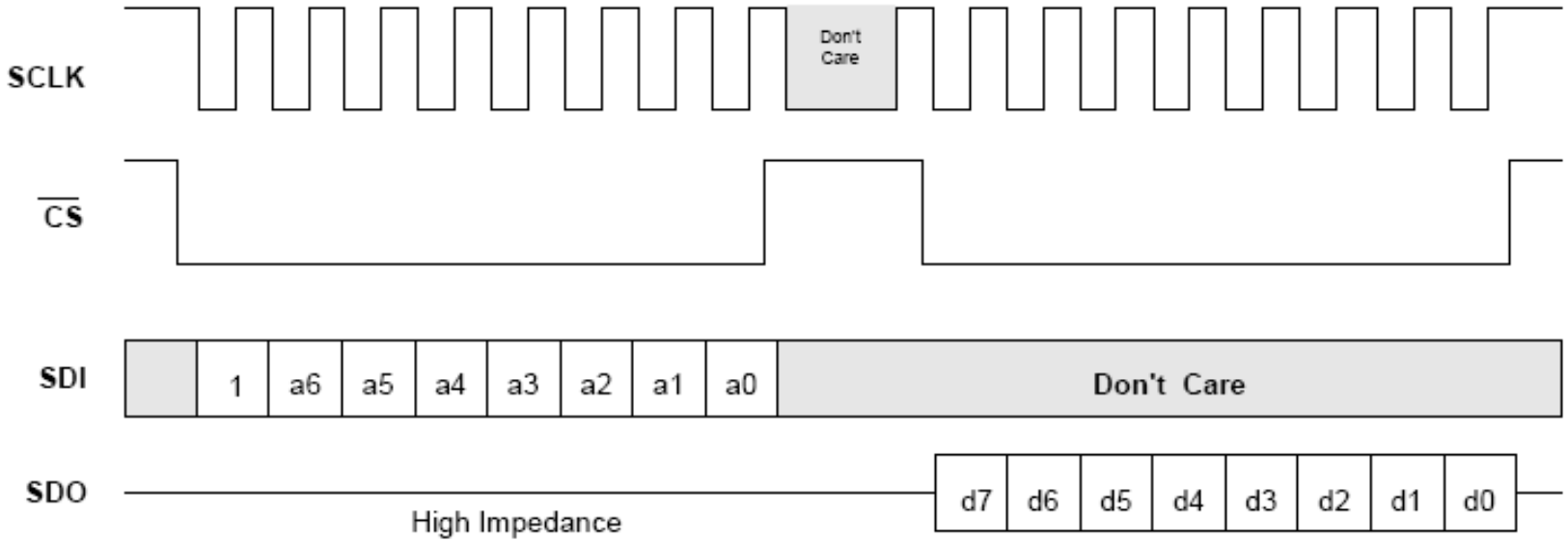
Si3210 – bus SPI (1)

- Quattro segnali utilizzati:
  - SCLK: SPI clock
  - CS: chip select, segnale di sincronizzazione per l'inizio del campionamento dei dati
  - SDO: SPI data out, segnale di trasmissione dati
  - SDI: SPI data in, segnale di ricezione dati
- Parametri configurabili: daisy chain mode



Si3210 – bus SPI (2)

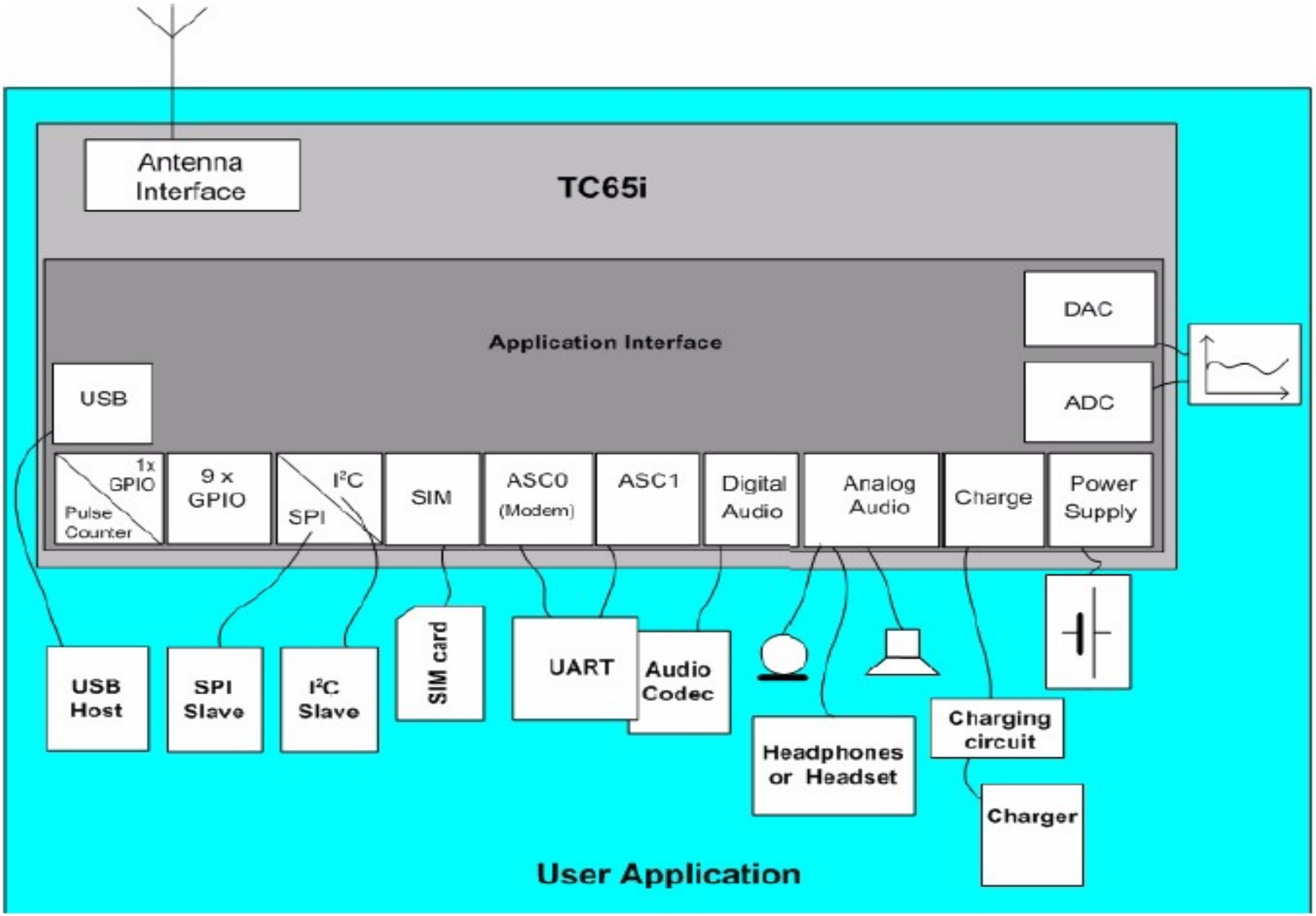




Cinterion TC65i – cos'è

- Modem GSM/GPRS quad band
  - Modulazioni/compressioni per GSM, CSD e FAX G3
  - Stack TCP/IP integrato
- Microcontrollore
  - Varietà di periferiche e interfacce
  - Programmabile in Java (JVM on board)
  - 1.7M flash, 400k RAM

TC65i – schema a blocchi



TC65i – funzionalità (1)

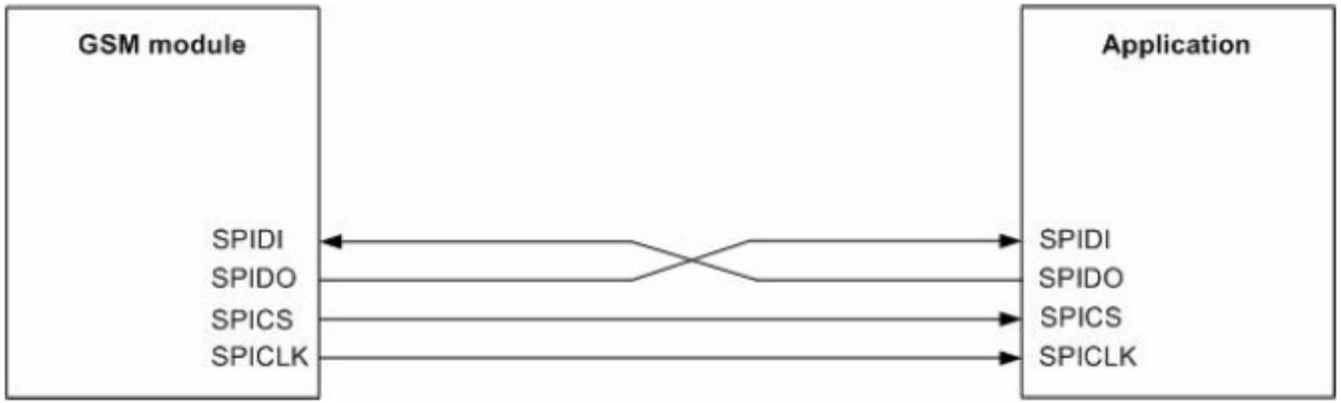
- Interfaccia SIM compatibile con la ISO 7816 (IC Card standard)
- Interfaccia seriale UART ASC0
  - 8 cavi, segnalazione ITU-T V.24
  - Controllo chiamate vocali, CSD, fax e GPRS
  - Controllo del modulo tramite comandi AT
- Interfaccia seriale UART ASC1
  - 4 cavi, segnalazione ITU-T V.24
- 10 pin GPIO (default output), interrupt capable

TC65i – funzionalità (2)

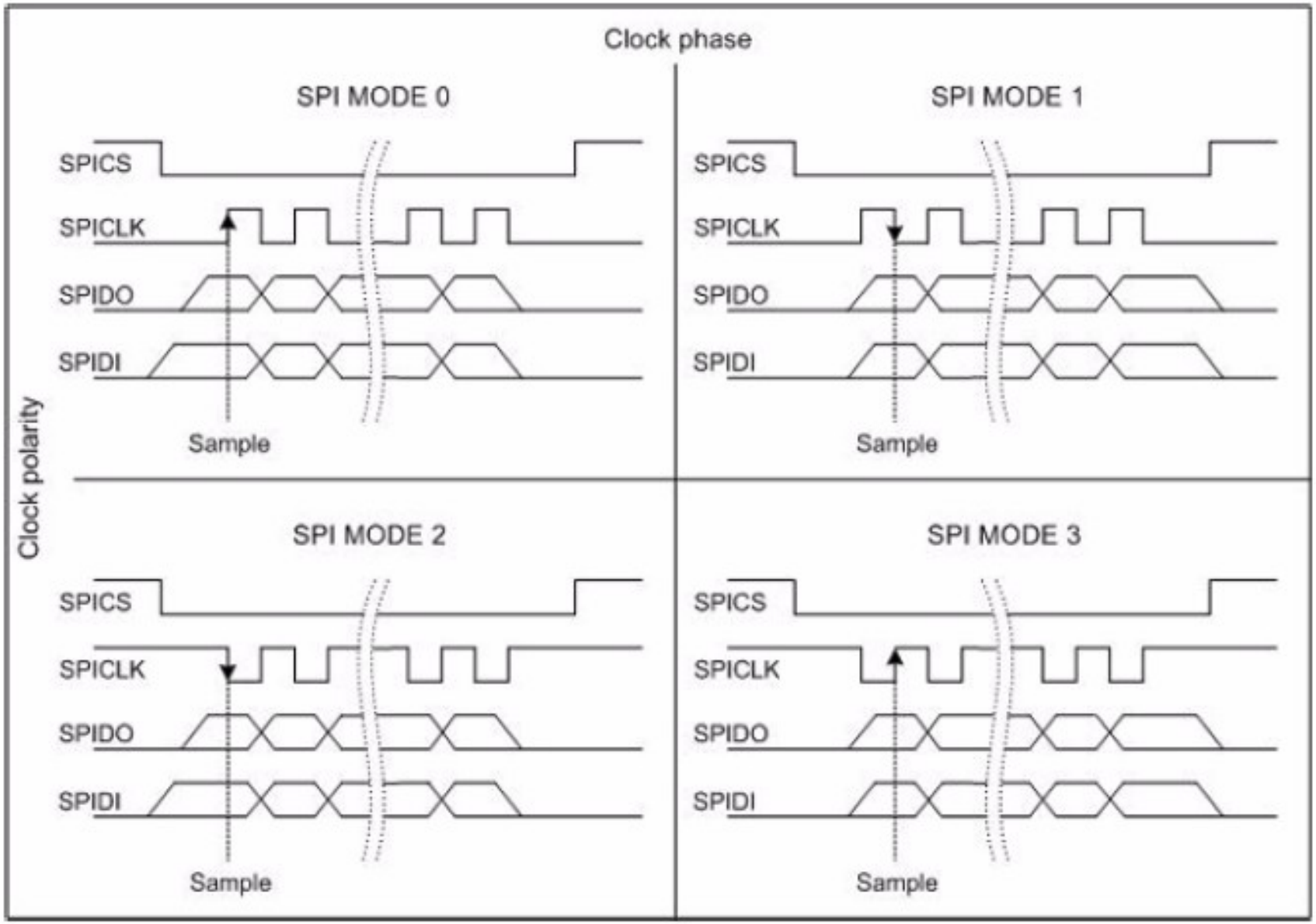
- Interfaccia USB 2.0 Full speed (12Mbit/s)
  - Utilizzata per controllo e download del firmware
- Interfaccia I2C master
- Interfaccia audio
  - Due interfacce audio analogiche
  - Un'interfaccia audio digitale (DAI)
- ADC a 2 canali, range 0-2400mV, risoluzione 1
- Interfaccia e connettore antenna

TC65i – bus SPI (1)

- Interfaccia di comunicazione e controllo tra il modulo GSM e un modulo applicativo
- Velocità di trasferimento fino a 6.5Mbit/s
- Modulo GSM configurato come master
- Quattro modalità di comunicazione (configurabili)



TC65i – bus SPI (2)



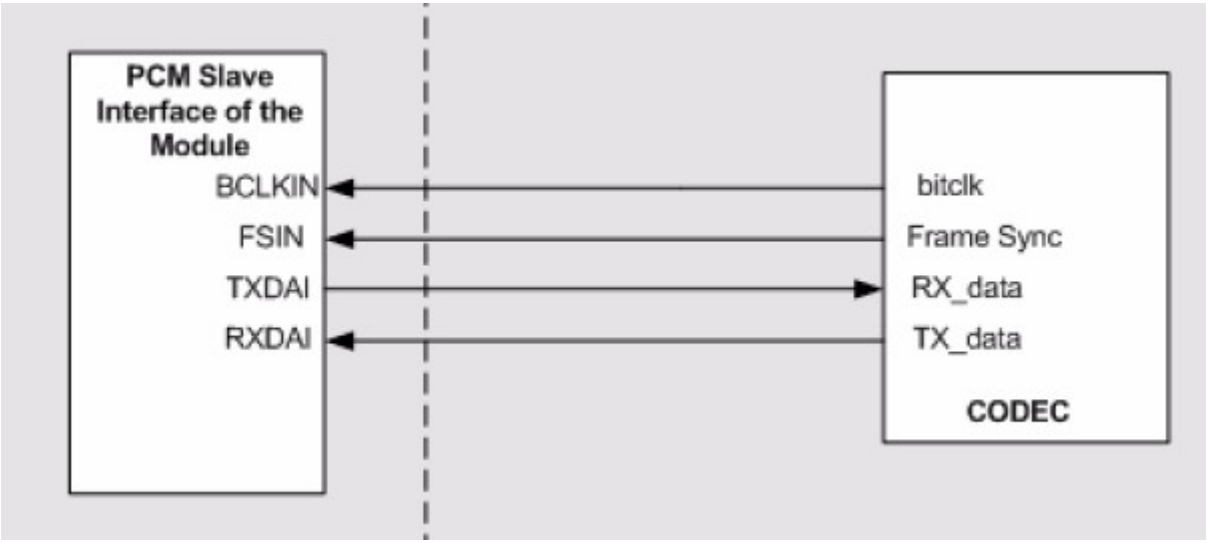
TC65i – interfaccia DAI (1)

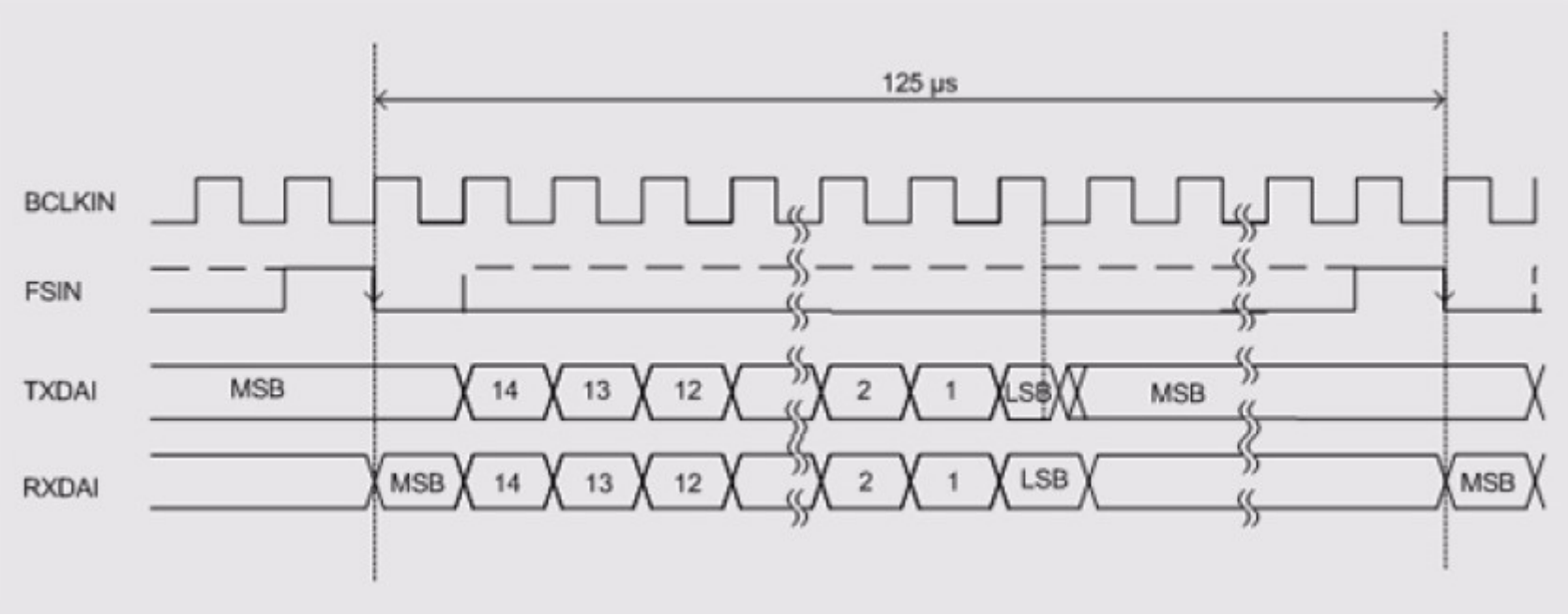
- DAI – Digital Audio Interface
- Interfaccia per dispositivi PCM (es: codec)
- Alternativa all'audio analogico per chiamate vocali
- Parametri configurabili
  - Modo master o slave
  - Sincronizzazione con frame lungo o corto
  - Frequenza di clock a 256 o 512 KHz

TC65i – interfaccia DAI (2)

- Caratteristiche del PCM
  - Campionamento 16bit lineare
  - Frequenza di campionamento a 8 kHz
  - Allineamento MSB
  - Durata del frame 125us
  - Frame sync comune per ricezione e trasmissione
- Slave mode
  - PCM clock e Frame sync esterni

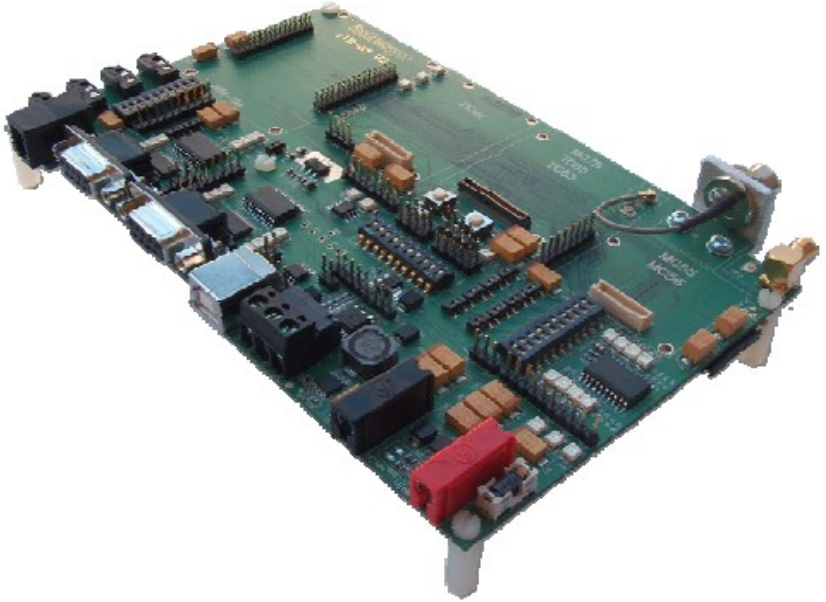
TC65i – interfaccia DAI (3)



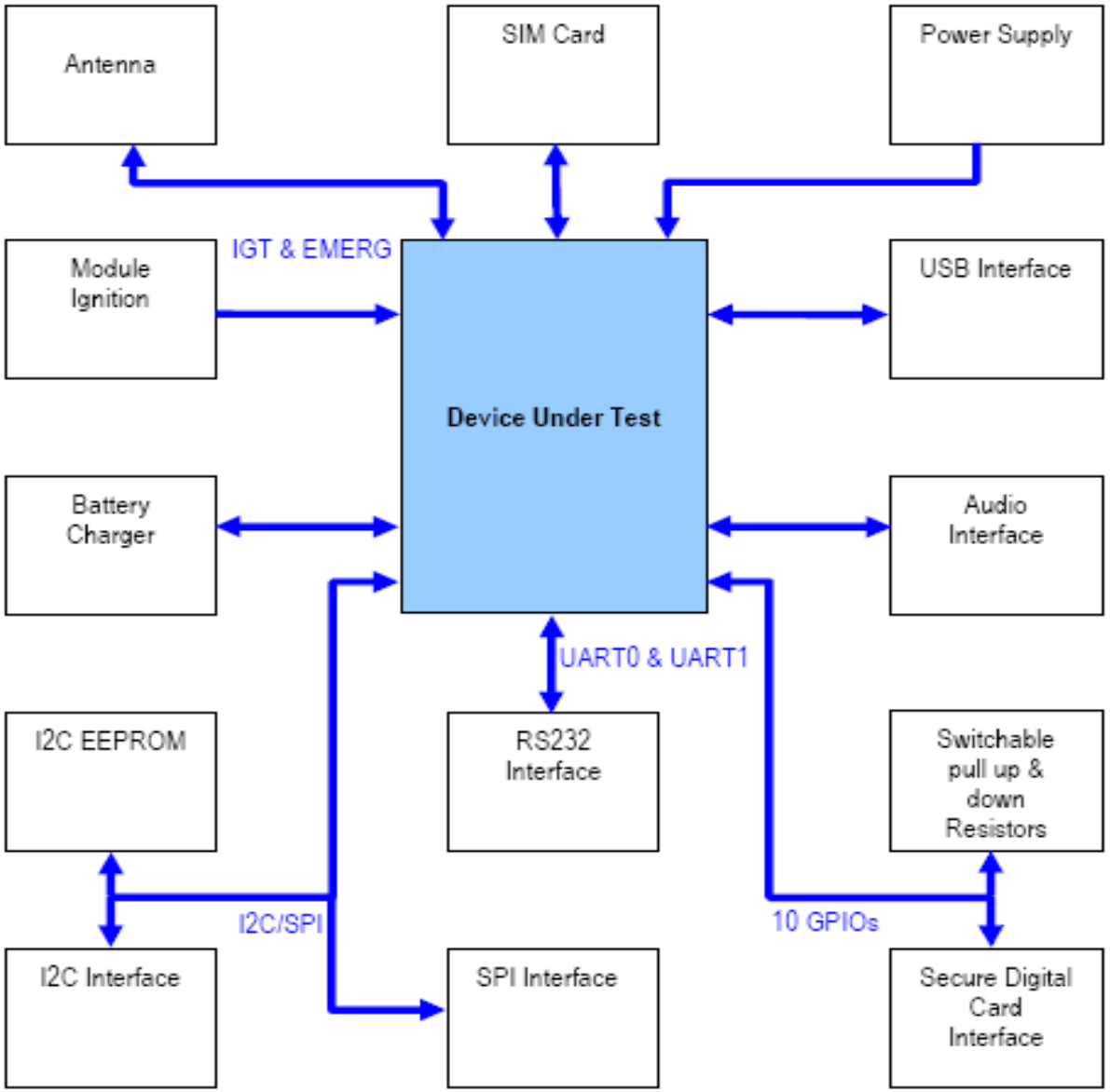


iTB–SM V2 – cos'è

- Scheda di sviluppo Intesyn
  - Possibilità di connessione di svariati modem GSM/GPRS/EDGE/UMTS
  - Connettori, punti di test, LED e altre periferiche
  - Circuiteria di alimentazione modem

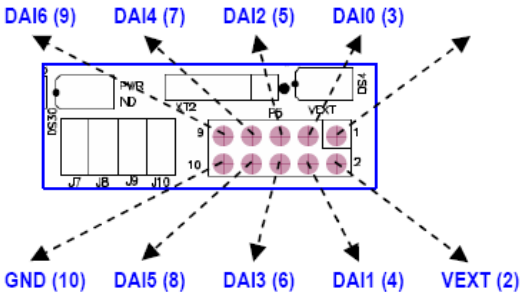


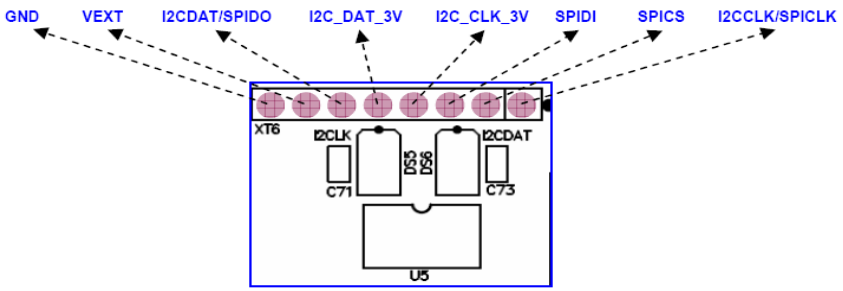
iTB–SM V2 – schema a blocchi

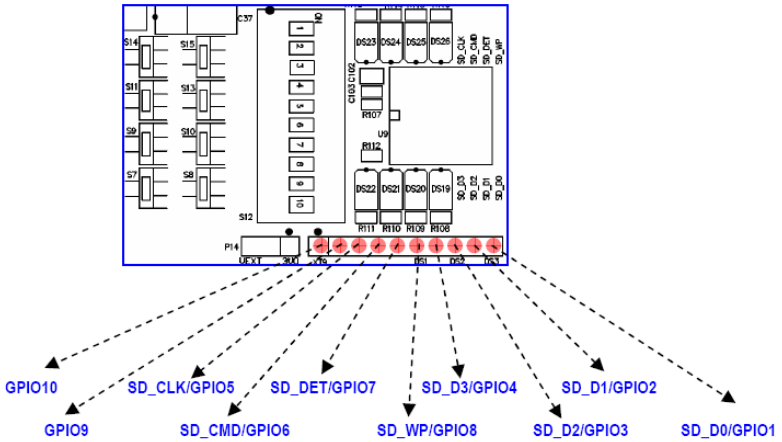


iTB–SM V2 – funzionalità

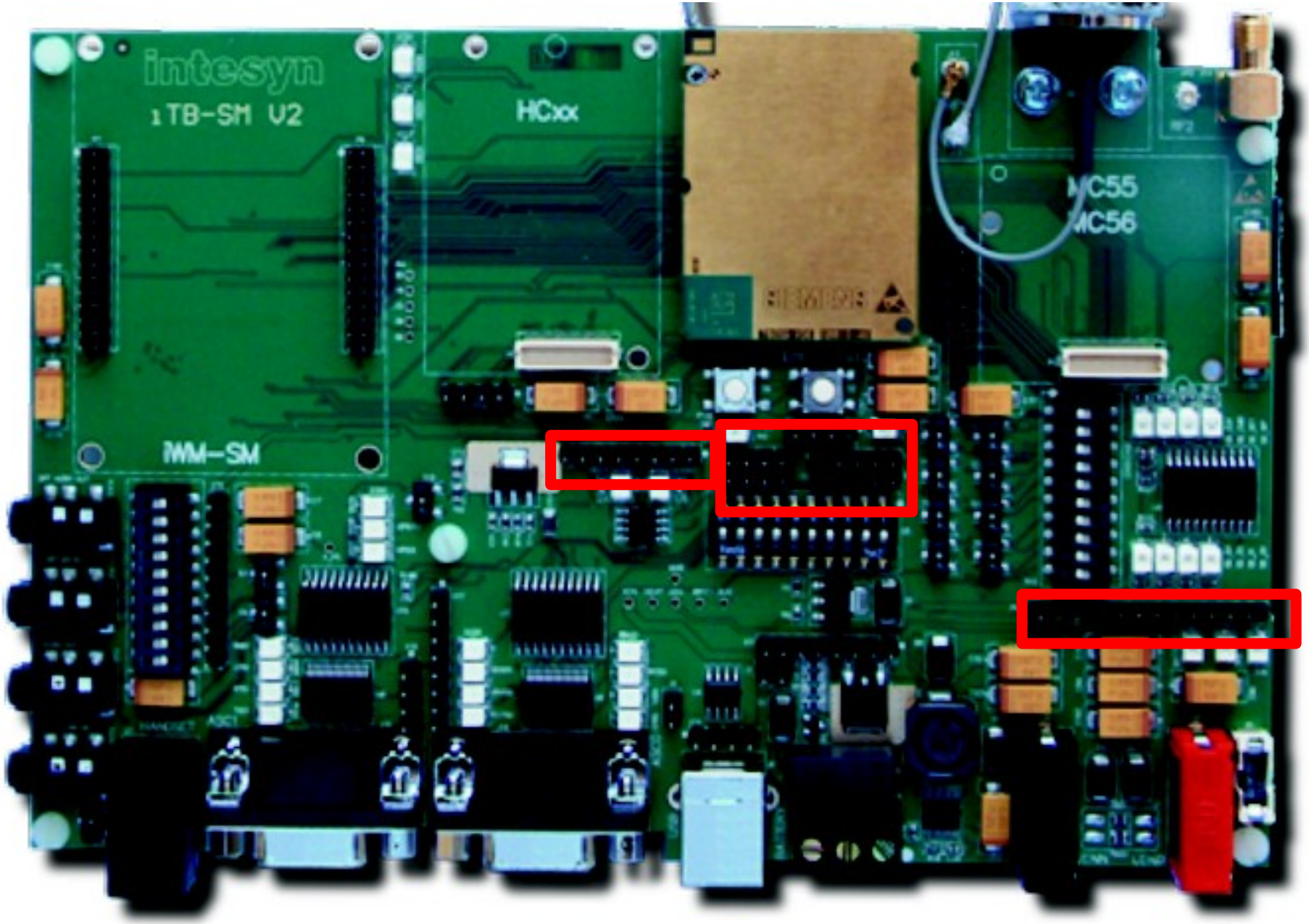
- Interfacce
  - Alimentazione
  - Modulo GSM
  - Antenna
  - SIM card
  - Comunicazione seriale RS-232 e USB
  - Audio (analogico e digitale)
  - I2C e SPI
  - Secure Digital
  - GPIO, ADC, DAC







iTB–SM V2 – test points



Logiche programmabili

- Si3210: dispositivo slave (PCM e SPI hanno bisogno di clock e FS)
- TC65i: configurabile master o slave, ma..

- Necessità di segnali di sincronizzazione esterni (CLK e FS) generati da una FPGA o CPLD
- Abbiamo un grado di libertà in più
- Ottimo candidato: PSoC (basso costo, versatilità)

Scelta componenti software

- Linguaggio di programmazione: Java
- Ambiente di sviluppo: NetBeans
- *Driver SPI*
- *Driver Si3210*
- *Driver GPIO*
- *Macchina a stati (applicazione event-driven)*
- *Diagramma delle classi*

Driver SPI

- Primitive a basso livello di astrazione
- Funzioni implementate
  - String read(int n): legge n byte dal bus SPI e ne ritorna la stringa corrispondente
  - int writeData(String data): scrive la stringa data sul bus SPI e ritorna il numero di byte scritti
- Utilizzato dal driver Si3210

Driver Si3210 (1)

- Driver di comunicazione
  - int readRegister(int register): legge il valore del registro register e ne restituisce il valore
  - void writeRegister(int register, int value): scrive il valore value nel registro register
  - int readRegisterIndirect(int register): legge il valore del registro indiretto register e ne restituisce il valore
  - void writeRegisterIndirect(int register, int value): scrive il valore value nel registro indiretto register

Driver Si3210 (2)

- Driver di configurazione e controllo SLIC
  - void initChip(): inizializza un chip Si3210
  - void enableInterrupts() (disableInterrupts()): abilita (disabilita) il sensing degli interrupt
  - void powerOn() (powerOff()): accende (spegne) un chip
  - void calibrate(): effettua la procedura di calibrazione
  - void startPCM() (stopPCM()): abilita (disabilita) la ricetrasmissione PCM

Driver Si3210 (3)

- void startRinging() (stopRinging()): abilita (disabilita) gli oscillatori di ring
- void goOnhook() (goOffhook()): modifica lo stato della linea di trasmissione
- void playStandardTone(int tone): abilita gli oscillatori del chip e genera un tono di linea standard
- void playDTMF(int dtmf): abilita gli oscillatori del chip e genera un tono DTMF

Driver GPIO

- Driver GPIO general purpose
- Usato per comandare due linee del Si3210
  - Linea di reset: pin di output, reset del chip Si3210
  - Linea di interrupt: pin di input (interrupt trigger), comunicazione dal Si3210 al modulo
- Scrittura routine di interrupt (ISR) per la corretta gestione degli eventi scatenati dal ProSLIC

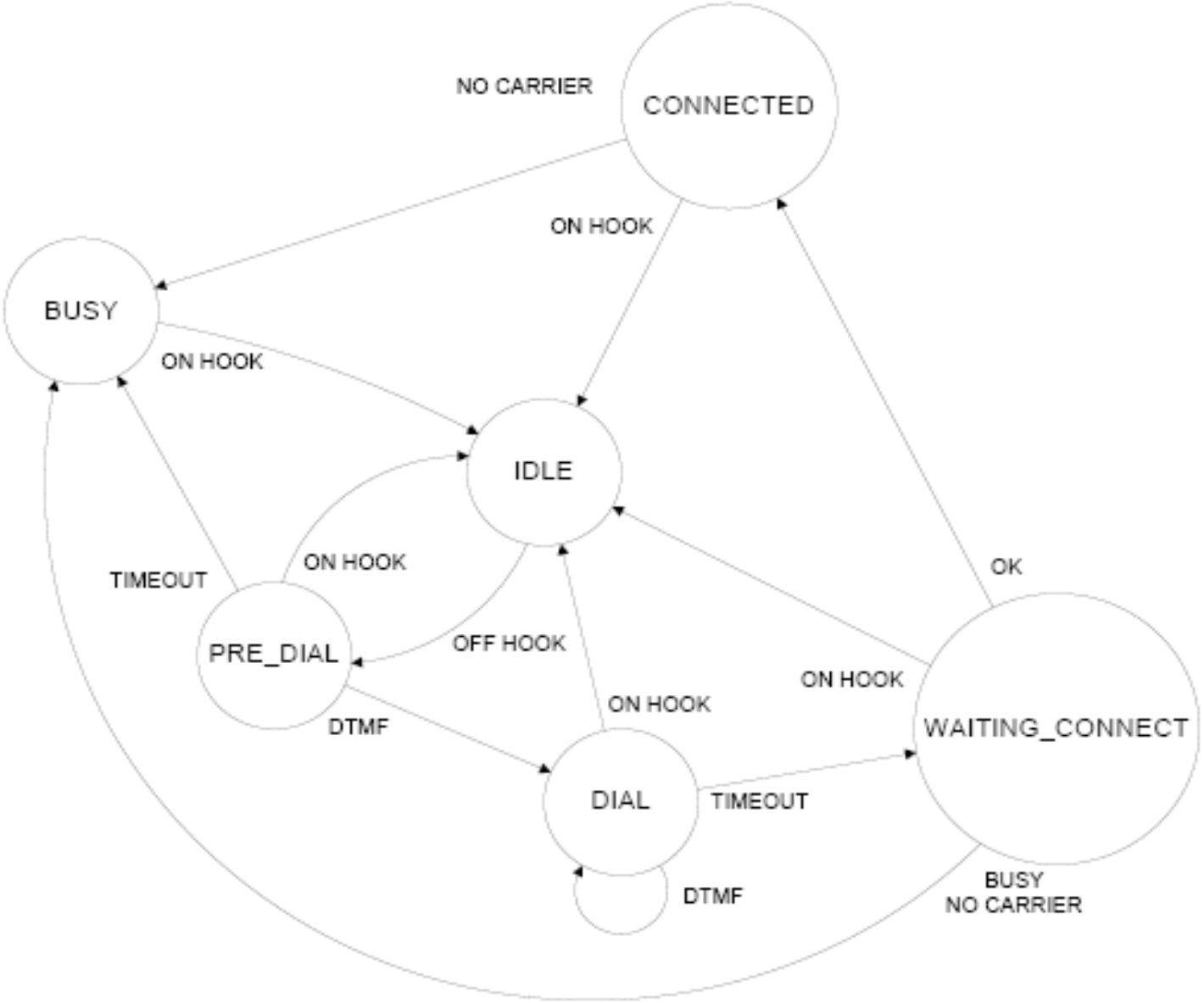
Comunicazione

- UART
  Universal Asynchronous Receiver Transmitter
- GPIO
  General Purpose I/O
- SPI
  Serial Peripheral Interface
- PCM
  Pulse Code Modulation

Progettazione macchina a stati (1)

- Applicazione event-driven
- Due funzionalità principali
  - Incoming call
  - Outgoing call
- Main loop semplice (serializzazione degli eventi)

Progettazione macchina a stati (2)



Progettazione macchina a stati (3)

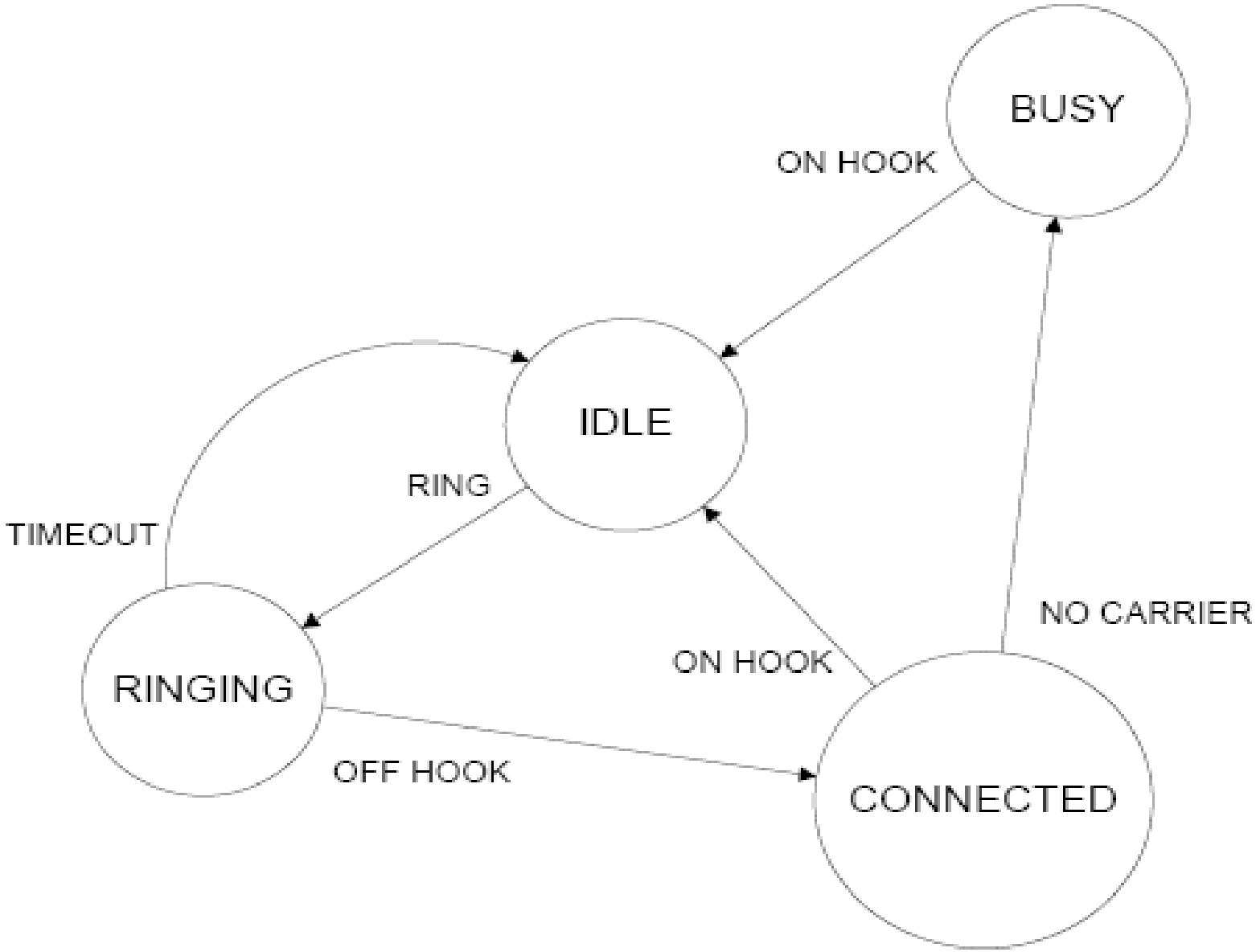
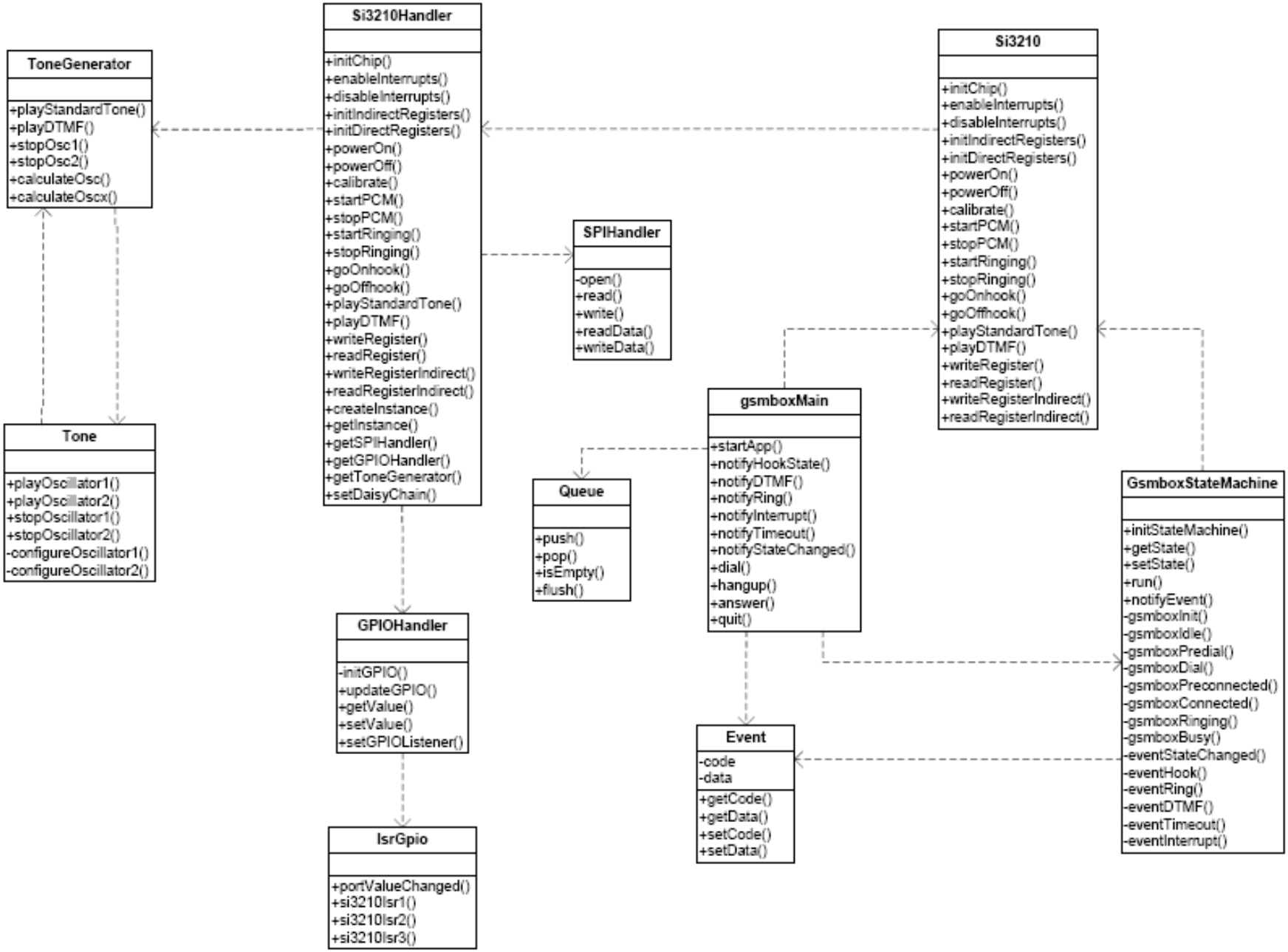


Diagramma delle classi



Esempio 1: Main loop

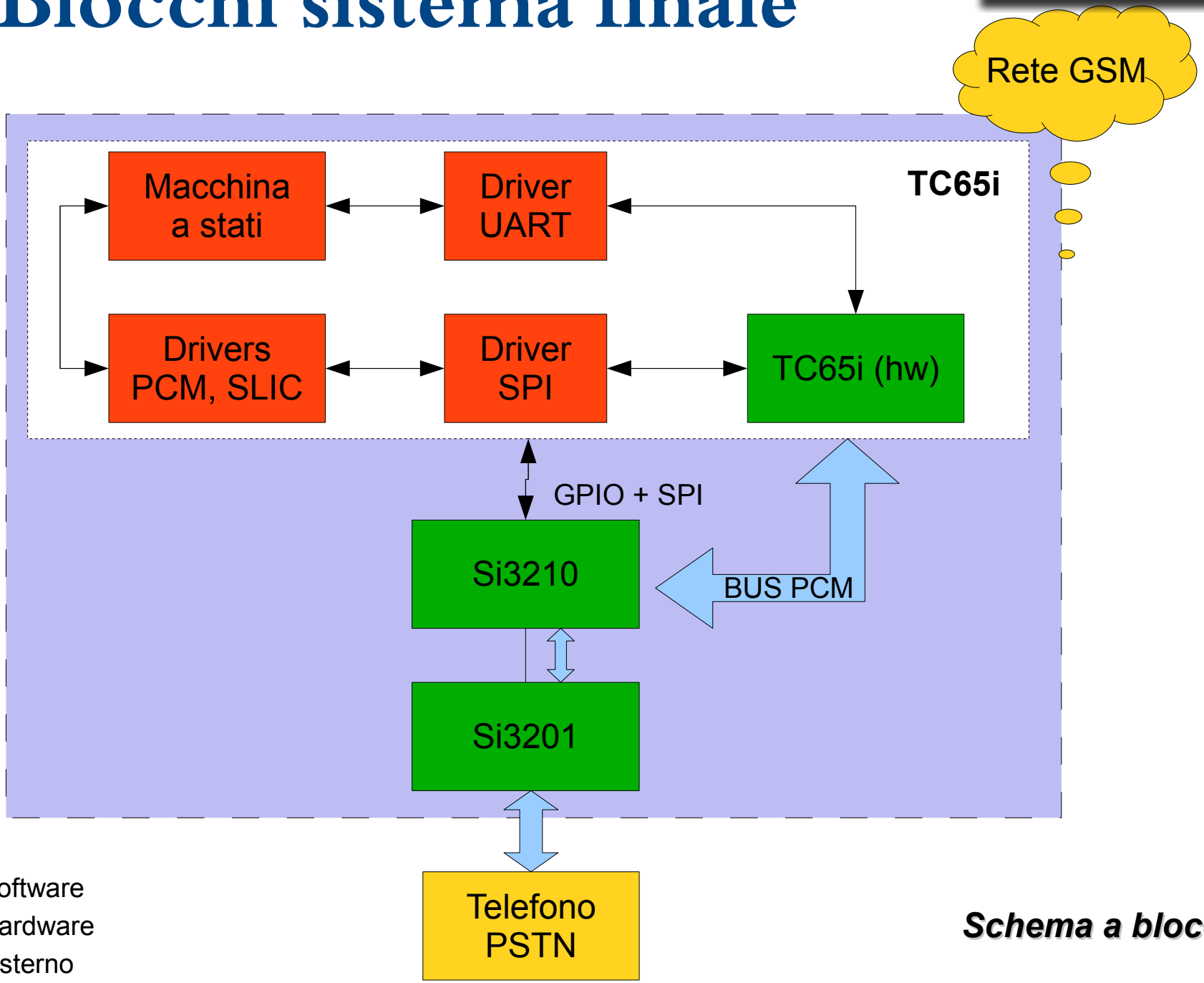
```
public void startApp() {
    Event e;
    char c;

    while(loop) {
        if(interrupt) {
            while(Si3210Handler.getInterruptStatus() == 0) {
                isr.si3210Isr1();
                isr.si3210Isr2();
                isr.si3210Isr3();
            }
            interrupt = false;
        }

        if(!eventQueue.isEmpty()) {
            e = (Event) eventQueue.pop();
            gsmboxStateMachine.notifyEvent(e);
        }
        if(console.isDataAvailable() > 0) {
            c = console.getChar();
            notifyConsole(c);
        }
    }
    destroyApp(true);
}
```

Esempio 2:

Notifica cambio di stato

```
private void eventStateChanged(int state) {
    switch(state) {
        case GSMBOX_PIN_STATE:
            gsmboxPin();
            break;

        case GSMBOX_INIT_STATE:
            gsmboxInit();
            break;

        case GSMBOX_IDLE_STATE:
            gsmboxIdle();
            break;

        case GSMBOX_PREDIAL_STATE:
            gsmboxPredial();
            break;

        case GSMBOX_DIAL_STATE:
            gsmboxDial();
            break;

        case GSMBOX_PRECONNECTED_STATE:
            gsmboxPreconnected();
            break;

        case GSMBOX_CONNECTED_STATE:
            gsmboxConnected();
            break;

        case GSMBOX_BUSY_STATE:
            gsmboxBusy();
            break;

        case GSMBOX_RINGING_STATE:
            gsmboxRinging();
            break;

        default:
            break;
    }
}
```

Blocchi sistema finale



- Blocco software
- Blocco hardware
- Mondo esterno

Schema a blocchi

Conclusioni e sviluppi futuri

- Realizzato il sistema gateway PSTN-GSM con alto livello di integrazione (SoC + ProSLIC e qualche componente al contorno)
- Semplicità nello sviluppo delle applicazioni e dei driver grazie alle API disponibili
- Ulteriori features per connettività completa
  - FAX G3 (connessione CSD)
  - Collegamento rete tramite GPRS