```
/* DO NOT EDIT THIS FILE.
 * This file is automatically generated by ./generate.sh from cbusdefs.csv
 */
#ifndef __CBUSDEFS
#define __CBUSDEFS

#ifdef  __cplusplus
extern "C" {
#endif

//
//              Copyright (C) Pete Brownlow 2011-2022   software@upsys.co.uk
//              Originally derived from opcodes.h (c) Andrew Crosland.
//              CSV version by Ian Hogg inspired by David W Radcliffe
//
//              Ver 8w
//
//                This work is licensed under the:
//                    Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International
License.
//                To view a copy of this license, visit:
//                    http://creativecommons.org/licenses/by-nc-sa/4.0/
//                or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042,
USA.
//
//                License summary:
//                  You are free to:
//                    Share, copy and redistribute the material in any medium or format
//                    Adapt, remix, transform, and build upon the material
//
//                  The licensor cannot revoke these freedoms as long as you follow the license
terms.
//
//                  Attribution : You must give appropriate credit, provide a link to the
license,
//                                and indicate if changes were made. You may do so in any
reasonable manner,
//                                but not in any way that suggests the licensor endorses you or
your use.
//
//                  NonCommercial : You may not use the material for commercial purposes. **(see
note below)
//
//                  ShareAlike : If you remix, transform, or build upon the material, you must
distribute
//                               your contributions under the same license as the original.
//
//                  No additional restrictions : You may not apply legal terms or technological
measures that
//                                               legally restrict others from doing anything the
license permits.
//
//                ** For commercial use, please contact the original copyright holder(s) to
agree licensing terms
//
//                  This software is distributed in the hope that it will be useful, but WITHOUT
ANY
//                  WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR
A PARTICULAR PURPOSE
//
//              Version history:
//              Pete Brownlow,6/4/11,Original from spec version 7e
//              Roger Healey,6/4/11,Add OPC_CMDERR response codes
//              Pete Brownlow,7/6/11,Updated to spec ver 7f and add new module ids
//              Pete Brownlow,4/7/11,Updated to spec ver 7g
//              Pete Brownlow,14/8/11,Updated to spec ver 7h
//              Pete Brownlow,18/2/12,Updated to spec ver 8a, Rocrail and animated modeller
module types added
//              Pete Brownlow,10/4/12,Updated to spec ver 8b, extended parameter bytes
//              Pete Brownlow,17/4/12,Updated parameter block definitions, added processor type
definitions.
//              Roger Healey,13/7/12,Add CANTOTI
//              Pete Brownlow,15/9/12,Updated to spec ver 8c, added 0x5D ENUM and 0x75 CANID
//              Pete Brownlow,4/1/13,Ver 8d New module types, added 0x5E  NNRST
//              Roger Healey,15/2/13,Now at version d. Added new Module Ids (as per ModuleIds
Issue 9) and
//                       Align Processor Ids with constants.inc
//                       Added .(fullstop) before each processor Id > 9
//                       Added OPC_NNRST
//              Roger Healey,27/4/13,Added CANSIG8 and CANSIG64
//              Roger Healey,6/8/13,Added CANCOND8C
//              Roger Healey,22/1/14,Added CANPAN, CANACE3C, CANPanel and CANMIO
//              Pete Brownlow,22/1/14,Ver 8g New module types, OPC_NNRST & OPC_NNRSM, catch up
with .inc file
//                       Added parameter definitions for manufacturer's CPU id
//                       Added extern C so can be included by c++ code
//                       Fixed CANGC1e type definition
//              Phil Wheeler,1/2/14,Corrected some typos. Added PRM_BETA, dotted some more
```

```
decimal numbers
//               Phil Wheeler,9/3/14,Corrected CANSIG MTYP definitions
//               Pete Brownlow,19/4/14,Ver 8h Added module type defs for CANTOTIMIO, CANACE8MIO,
CANBIP, CANSOL
//               Pete Brownlow,06/7/15,Ver 8j Add new module types as per modules definitions
v17 : CANCDU, CANACC4CDU, CANWiBase, WiCAB, CANWiFi, CANFTT
//                         Add new opcode ALOC, added CPU manufacturer code CPUM_ATMEL
//               Mike Bolton,2/3/16,Ver 8k    Add module type CANRFID8
//               Pete Brownlow,2/7/16,Ver 8m Add new module types CANHNDST, CANTCHNDST, CANmchRFID
and CANPiWi
//                         Add processor codes for ARM chips used in Raspberry Pi's
//               Pete Brownlow,16/2/17,Ver 8n Add module ids to bring it up to module ids document
ver 25
//               Pete Brownlow,29/7/17,Ver 8p Add new module ids and new cab signalling opcode
//               Pete Brownlow,09/1/18,Ver 8q Add new parameter flag for module can consume its
own events
//               Ian Hogg,11/9/18,Ver 8r Added CANACE16CMIO, CANPiNODE, CANDISP, CANCOMPUTE
//               Richard Crawshaw,29/02/2020,Fixed order of columns in CbusCabSigAspect2.
//               Pete Brownlow,01/09/20,Ver 8s for additional module ids defined in the ModuleIds
file ver 33.
//                                      Updated descriptive comments for some module types
//                                      Updated CABDAT opcode to match RFC0004
//               Pete Brownlow,06/09/20,Ver 8t Added module type for CANRCOM. Fixed: Opcode for
CABDAT, names for CANRC522 and CANMAG
//               Pete Brownlow,13/10/20,Ver 8u Added module types 67 to 74 including some Arduino
projects
//                                              Added SPROG manufacturer code 44 and new SPROG CBUS
module types
//                                              Additional error code for overload - now removed as
not required after all
//                                              New bus type USB for modules with only USB and no
CAN
//               Pete Brownlow,19/02/21,Ver 8u Added manufacturer code 13 for new development -
who don't have a manufacturer id yet
//                                              Added proccessor identification codes for 18F25k83,
18F26k83 and 18F14K22.
//               Andrew Crosland,21/09/2021,Ver 8t Added PICs P18F14K22 P18F26K83 P18F27Q84
P18F47Q84 and P18F27Q83
//               Andrew Crosland,19/01/2022,Ver 8t, Added OPC_VCVS, Verify CV service mode - used
for CV read hints, update SPROG modules types (PR#13)
//               Duncan Greenwood,07/10/2021,Ver 8t Added OPC_DTXC opcode (0xE9) for CBUS long
messages - RFC 0005
//               Richard Crawshaw,11/10/2021,Ver 8t Fixed trailing comma in CbusCabSigAspect0
//               Pete Brownlow,28/07/2022,Ver 8v Resolve and merge changes in 8u branch with
changes subsequently applied to master, now ver 8v in new branch,
//                                                                              Add requested module
type ids 75 to 78
//                                              Resolve changes from PR #13,  move proposed and/or
agreed opcodes not yet in the published spec to below the others
//               Pete Brownlow,5/08/2022, Ver 8w  Add module type 79 for CANBUFFER
//               Pete Brownlow,5/01/2023, Ver 8w  Add module type 80 for CANPMSense
//
// CBUS Manufacturer definitions
// Where the manufacturer already has an NMRA code, this is used
//
#define MANU_DEV 13       // For new manufacturer development - who don't have a manufacturer id
yet
#define MANU_MERG165      // https://www.merg.co.uk
#define MANU_SPROG        44       // https://www.sprog-dcc.co.uk/
#define MANU_ROCRAIL      70       // http://www.rocrail.net
#define MANU_SPECTRUM     80       // http://animatedmodeler.com  (Spectrum Engineering)
#define MANU_SYSPIXIE     249      // Konrad Orlowski
#define MANU_RME 248      // http://rmeuk.com  (Railway Modelling Experts Limited)
//
//
// MODULE TYPES
//
// Please note that the existence of a module type id does not necessarily mean that firmware has
been implemented
//
// MERG Module types
//
#define MTYP_SLIM0        // default for SLiM nodes
#define MTYP_CANACC4      1        // Solenoid point driver
#define MTYP_CANACC5      2        // Motorised point driver
#define MTYP_CANACC8      3        // 8 digital outputs
#define MTYP_CANACE3      4        // Control panel switch/button encoder
#define MTYP_CANACE8C     5        // 8 digital inputs
#define MTYP_CANLED       6        // 64 led driver
#define MTYP_CANLED64     7        // 64 led driver (multi leds per event)
#define MTYP_CANACC4_2    8        // 12v version of CANACC4
#define MTYP_CANCAB       9        // CANCAB hand throttle
#define MTYP_CANCMD       10       // CANCMD command station
#define MTYP_CANSERVO     11       // 8 servo driver (on canacc8 or similar hardware)
#define MTYP_CANBC        12       // BC1a command station
#define MTYP_CANRPI       13       // RPI and RFID interface
#define MTYP_CANTTCA      14       // Turntable controller (turntable end)
#define MTYP_CANTTCB      15       // Turntable controller (control panel end)
```

```
#define MTYP_CANHS        16      // Handset controller for old BC1a type handsets
#define MTYP_CANTOTI      17      // Track occupancy detector
#define MTYP_CAN8I8O      18      // 8 inputs 8 outputs
#define MTYP_CANSERVO8C   19      // Canservo with servo position feedback
#define MTYP_CANRFID      20      // RFID input
#define MTYP_CANTC4       21      //
#define MTYP_CANACE16C    22      // 16 inputs
#define MTYP_CANIO8       23      // 8 way I/O
#define MTYP_CANSNDX      24      // ??
#define MTYP_CANEther     25      // Ethernet interface
#define MTYP_CANSIG64     26      // Multiple aspect signalling for CANLED module
#define MTYP_CANSIG8      27      // Multiple aspect signalling for CANACC8 module
#define MTYP_CANCOND8C    28      // Conditional event generation
#define MTYP_CANPAN       29      // Control panel 32/32
#define MTYP_CANACE3C     30      // Newer version of CANACE3 firmware
#define MTYP_CANPanel     31      // Control panel 64/64
#define MTYP_CANMIO       32      // Multiple I/O â€" Universal CANMIO firmware
#define MTYP_CANACE8MIO   33      // Multiple IO module 16 inputs emulating CANACE8C on CANMIO
hardware
#define MTYP_CANSOL       34      // Solenoid driver module
#define MTYP_CANBIP       35      // Universal CANBIP firmware - Bipolar IO module with additional
8 I/O pins (CANMIO family)
#define MTYP_CANCDU       36      // Solenoid driver module with additional 6 I/O pins (CANMIO
family)
#define MTYP_CANACC4CDU   37      // CANACC4 firmware ported to CANCDU
#define MTYP_CANWiBase    38      // CAN to MiWi base station
#define MTYP_WiCAB        39      // Wireless cab using MiWi protocol
#define MTYP_CANWiFi      40      // CAN to WiFi connection with Withrottle to CBUS protocol
conversion
#define MTYP_CANFTT       41      // Turntable controller configured using FLiM
#define MTYP_CANHNDST     42      // Handset (alternative to CANCAB)
#define MTYP_CANTCHNDST   43      // Touchscreen handset
#define MTYP_CANRFID8     44      // multi-channel RFID reader
#define MTYP_CANmchRFID   45      // either a 2ch or 8ch RFID reader
#define MTYP_CANPiWi      46      // a Raspberry Pi based module for WiFi
#define MTYP_CAN4DC       47      // DC train controller
#define MTYP_CANELEV      48      // Nelevator controller
#define MTYP_CANSCAN      49      // 128 switch inputs
#define MTYP_CANMIO_SVO   50      // 16MHz 25k80 version of CANSERVO8c on CANMIO hardware
#define MTYP_CANMIO_INP   51      // 16MHz 25k80 version of CANACE8MIO on CANMIO hardware
#define MTYP_CANMIO_OUT   52      // 16MHz 25k80 version of CANACC8 on CANMIO hardware
#define MTYP_CANBIP_OUT   53      // 16MHz 25k80 version of CANACC5 on CANBIP hardware
#define MTYP_CANASTOP     54      // DCC stop generator
#define MTYP_CANCSB       55      // CANCMD with on board 3A booster
#define MTYP_CANMAG       56      // Magnet on Track detector
#define MTYP_CANACE16CMIO 57      // 16 input equivaent to CANACE8C
#define MTYP_CANPiNODE    58      // CBUS module based on Raspberry Pi
#define MTYP_CANDISP      59      // 25K80 version of CANLED64 (IHart and MB)
#define MTYP_CANCOMPUTE   60      // Compute Event processing engine
#define MTYP_CANRC522     61      // Read/Write from/to RC522 RFID tags
#define MTYP_CANINP       62      // 8 inputs module (2g version of CANACE8c) (Pete Brownlow)
#define MTYP_CANOUT       63      // 8 outputs module (2g version of CANACC8) (Pete Brownlow)
#define MTYP_CANEMIO      64      // Extended CANMIO (24 I/O ports) (Pete Brownlow)
#define MTYP_CANCABDC     65      // DC cab
#define MTYP_CANRCOM      66      // DC Railcom detector/reader
#define MTYP_CANMP3       67      // MP3 sound player in response to events (eg: station
announcements) (Duncan Greenwood)
#define MTYP_CANXMAS      68      // Addressed RGB LED driver (Duncan Greenwood)
#define MTYP_CANSVOSET    69      // Servo setting box (Duncan Greenwood)
#define MTYP_CANCMDDC     70      // DC Command station
#define MTYP_CANTEXT      71      // Text message display
#define MTYP_CANASIGNAL   72      // Signal controller
#define MTYP_CANSLIDER    73      // DCC cab with slider control (Dave Radcliffe)
#define MTYP_CANDCATC     74      // DC ATC module (Dave Harris)
#define MTYP_CANGATE      75      // Logic module using and/or gates (Phil Silver)
#define MTYP_CANSINP      76      // Q series PIC input module (Ian Hart)
#define MTYP_CANSOUT      77      // Q series PIC input module (Ian Hart)
#define MTYP_CANSBIP      78      // Q series PIC input module (Ian Hart)
#define MTYP_CANBUFFER    79      // Message buffer (Phil Silver)
//
//
//
// At the time of writing the list of defined MERG module types is maintained by Pete Brownlow
software@upsys.co.uk
// Please liaise with Pete before adding new module types,
// and/or create your own GitHub branch, add your proposed new module type(s) and then create a
Pull Request
//
#define MTYP_CAN_SW       0xFF    // Software nodes
#define MTYP_EMPTY        0xFE    // Empty module, bootloader only
#define MTYP_CANUSB       0xFD    // USB interface
//
// Sprog Module types
//
#define MTYP_CANPiSPRG3   1       // Pi-SPROG 3 programmer/command station
#define MTYP_CANSPROG3P   2       // SPROG 3 Plus programmer/command station
#define MTYP_CANSPROG     3       // CAN SPROG programmer/command station
#define MTYP_CANSBOOST    4       // System Booster
```

```
#define MTYP_CANPiSPRGP   5       // Pi-SPROG 3 Plus programmer/command station
#define MTYP_CANISB       6       // CAN ISB Isolated CAN USB Interface
#define MTYP_CANIO        7       // 8-channel I/O module
#define MTYP_CANSERVOIO   8       // 8-channel Servo I/O module
#define MTYP_CANSOLIO     9       // 8-channel (4-pairs) Solenoid I/O module
//
//
// Rocrail Module types
//
#define MTYP_CANGC1       1       // RS232 PC interface
#define MTYP_CANGC2       2       // 16 I/O
#define MTYP_CANGC3       3       // Command station (derived from cancmd)
#define MTYP_CANGC4       4       // 8 channel RFID reader
#define MTYP_CANGC5       5       // Cab for fixed panels (derived from cancab)
#define MTYP_CANGC6       6       // 4 channel servo controller
#define MTYP_CANGC7       7       // Fast clock module
#define MTYP_CANGC1e      11      // CAN<->Ethernet interface
//
// Spectrum Engineering Animated Modeller module types
//
#define MTYP_AMCTRLR      1       // Animation controller (firmware derived from cancmd)
#define MTYP_DUALCAB      2       // Dual cab based on cancab
//
//
// SysPixie Module types (Konrad Orlowski)
//
#define MTYP_CANPMSense   1       // Motorised point motor driver with current sense
//
//
//
// CBUS opcodes list
//
// Packets with no data bytes
//
#define OPC_ACK  0x00     // General ack
#define OPC_NAK  0x01     // General nak
#define OPC_HLT  0x02     // Bus Halt
#define OPC_BON  0x03     // Bus on
#define OPC_TOF  0x04     // Track off
#define OPC_TON  0x05     // Track on
#define OPC_ESTOP0x06     // Track stopped
#define OPC_ARST 0x07     // System reset
#define OPC_RTOF 0x08     // Request track off
#define OPC_RTON 0x09     // Request track on
#define OPC_RESTP0x0a     // Request emergency stop all
#define OPC_RSTAT0x0c     // Request node status
#define OPC_QNN  0x0d     // Query nodes
//
#define OPC_RQNP 0x10     // Read node parameters
#define OPC_RQMN 0x11     // Request name of module type
//
// Packets with 1 data byte
//
#define OPC_KLOC 0x21     // Release engine by handle
#define OPC_QLOC 0x22     // Query engine by handle
#define OPC_DKEEP0x23     // Keep alive for cab
//
#define OPC_DBG1 0x30     // Debug message with 1 status byte
#define OPC_EXTC 0x3F     // Extended opcode
//
// Packets with 2 data bytes
//
#define OPC_RLOC 0x40     // Request session for loco
#define OPC_QCON 0x41     // Query consist
#define OPC_SNN  0x42     // Set node number
#define OPC_ALOC 0X43     // Allocate loco (used to allocate to a shuttle in cancmd)
//
#define OPC_STMOD0x44     // Set Throttle mode
#define OPC_PCON 0x45     // Consist loco
#define OPC_KCON 0x46     // De-consist loco
#define OPC_DSPD 0x47     // Loco speed/dir
#define OPC_DFLG 0x48     // Set engine flags
#define OPC_DFNON0x49     // Loco function on
#define OPC_DFNOF0x4A     // Loco function off
#define OPC_SSTAT0x4C     // Service mode status
#define OPC_NNRSM0x4F     // Reset to manufacturer's defaults
//
#define OPC_RQNN 0x50     // Request Node number in setup mode
#define OPC_NNREL0x51     // Node number release
#define OPC_NNACK0x52     // Node number acknowledge
#define OPC_NNLRN0x53     // Set learn mode
#define OPC_NNULN0x54     // Release learn mode
#define OPC_NNCLR0x55     // Clear all events
#define OPC_NNEVN0x56     // Read available event slots
#define OPC_NERD 0x57     // Read all stored events
#define OPC_RQEVN0x58     // Read number of stored events
#define OPC_WRACK0x59     // Write acknowledge
#define OPC_RQDAT0x5A     // Request node data event
```

```
#define OPC_RQDDS 0x5B    // Request short data frame
#define OPC_BOOT  0x5C    // Put node into boot mode
#define OPC_ENUM  0x5D    // Force can_id self enumeration
#define OPC_NNRST 0x5E    // Reset node (as in restart)
#define OPC_EXTC1 0x5F    // Extended opcode with 1 data byte
//
// Packets with 3 data bytes
//
#define OPC_DFUN  0x60    // Set engine functions
#define OPC_GLOC  0x61    // Get loco (with support for steal/share)
#define OPC_ERR   0x63    // Command station error
#define OPC_CMDERR        0x6F    // Errors from nodes during config
//
#define OPC_EVNLF 0x70    // Event slots left response
#define OPC_NVRD  0x71    // Request read of node variable
#define OPC_NENRD 0x72    // Request read stored event by index
#define OPC_RQNPN 0x73    // Request read module parameters
#define OPC_NUMEV 0x74    // Number of events stored response
#define OPC_CANID 0x75    // Set canid
#define OPC_EXTC2 0x7F    // Extended opcode with 2 data bytes
//
// Packets with 4 data bytes
//
#define OPC_RDCC3 0x80    // 3 byte DCC packet
#define OPC_WCVO  0x82    // Write CV byte Ops mode by handle
#define OPC_WCVB  0x83    // Write CV bit Ops mode by handle
#define OPC_QCVS  0x84    // Read CV
#define OPC_PCVS  0x85    // Report CV
//
#define OPC_ACON  0x90    // on event
#define OPC_ACOF  0x91    // off event
#define OPC_AREQ  0x92    // Accessory Request event
#define OPC_ARON  0x93    // Accessory response event on
#define OPC_AROF  0x94    // Accessory response event off
#define OPC_EVULN 0x95    // Unlearn event
#define OPC_NVSET 0x96    // Set a node variable
#define OPC_NVANS 0x97    // Node variable value response
#define OPC_ASON  0x98    // Short event on
#define OPC_ASOF  0x99    // Short event off
#define OPC_ASRQ  0x9A    // Short Request event
#define OPC_PARAN 0x9B    // Single node parameter response
#define OPC_REVAL 0x9C    // Request read of event variable
#define OPC_ARSON 0x9D    // Accessory short response on event
#define OPC_ARSOF 0x9E    // Accessory short response off event
#define OPC_EXTC3 0x9F    // Extended opcode with 3 data bytes
//
// Packets with 5 data bytes
//
#define OPC_RDCC4 0xA0    // 4 byte DCC packet
#define OPC_WCVS  0xA2    // Write CV service mode
//
#define OPC_ACON1 0xB0    // On event with one data byte
#define OPC_ACOF1 0xB1    // Off event with one data byte
#define OPC_REQEV 0xB2    // Read event variable in learn mode
#define OPC_ARON1 0xB3    // Accessory on response (1 data byte)
#define OPC_AROF1 0xB4    // Accessory off response (1 data byte)
#define OPC_NEVAL 0xB5    // Event variable by index read response
#define OPC_PNN   0xB6    // Response to QNN
#define OPC_ASON1 0xB8    // Accessory short on with 1 data byte
#define OPC_ASOF1 0xB9    // Accessory short off with 1 data byte
#define OPC_ARSON1        0xBD    // Short response event on with one data byte
#define OPC_ARSOF1        0xBE    // Short response event off with one data byte
#define OPC_EXTC4 0xBF    // Extended opcode with 4 data bytes
//
// Packets with 6 data bytes
//
#define OPC_RDCC5 0xC0    // 5 byte DCC packet
#define OPC_WCVOA 0xC1    // Write CV ops mode by address
#define OPC_CABDAT        0xC2    // Cab data (cab signalling)
#define OPC_FCLK  0xCF    // Fast clock
//
#define OPC_ACON2 0xD0    // On event with two data bytes
#define OPC_ACOF2 0xD1    // Off event with two data bytes
#define OPC_EVLRN 0xd2    // Teach event
#define OPC_EVANS 0xd3    // Event variable read response in learn mode
#define OPC_ARON2 0xD4    // Accessory on response
#define OPC_AROF2 0xD5    // Accessory off response
#define OPC_ASON2 0xD8    // Accessory short on with 2 data bytes
#define OPC_ASOF2 0xD9    // Accessory short off with 2 data bytes
#define OPC_ARSON2        0xDD    // Short response event on with two data bytes
#define OPC_ARSOF2        0xDE    // Short response event off with two data bytes
#define OPC_EXTC5 0xDF    // Extended opcode with 5 data bytes
//
// Packets with 7 data bytes
//
#define OPC_RDCC6 0xE0    // 6 byte DCC packets
#define OPC_PLOC  0xE1    // Loco session report
#define OPC_NAME  0xE2    // Module name response
```

```
#define OPC_STAT 0xE3     // Command station status report
#define OPC_PARAMS        0xEF     // Node parameters response
//
#define OPC_ACON3 0xF0     // On event with 3 data bytes
#define OPC_ACOF3 0xF1     // Off event with 3 data bytes
#define OPC_ENRSP 0xF2     // Read node events response
#define OPC_ARON3 0xF3     // Accessory on response
#define OPC_AROF3 0xF4     // Accessory off response
#define OPC_EVLRNI        0xF5     // Teach event using event indexing
#define OPC_ACDAT 0xF6     // Accessory data event: 5 bytes of node data (eg: RFID)
#define OPC_ARDAT 0xF7     // Accessory data response
#define OPC_ASON3 0xF8     // Accessory short on with 3 data bytes
#define OPC_ASOF3 0xF9     // Accessory short off with 3 data bytes
#define OPC_DDES  0xFA     // Short data frame aka device data event (device id plus 5 data bytes)
#define OPC_DDRS  0xFB     // Short data frame response aka device data response
#define OPC_DDWS  0xFC     // Device Data Write Short
#define OPC_ARSON3        0xFD     // Short response event on with 3 data bytes
#define OPC_ARSOF3        0xFE     // Short response event off with 3 data bytes
#define OPC_EXTC6 0xFF     // Extended opcode with 6 data byes
//
// Opcodes that are proposed and/or agreed but not yet in the current published specification
//
#define OPC_VCVS  0xA4     // Verify CV service mode - used for CV read hints
#define OPC_DTXC  0xE9     // CBUS long message packet
//
//
// Modes for STMOD
//
#define TMOD_SPD_MASK     3        //
#define TMOD_SPD_128      0        //
#define TMOD_SPD_14       1        //
#define TMOD_SPD_28I      2        //
#define TMOD_SPD_28       3        //
//
// Error codes for OPC_ERR
//
#define ERR_LOCO_STACK_FULL       1        //
#define ERR_LOCO_ADDR_TAKEN       2        //
#define ERR_SESSION_NOT_PRESENT   3        //
#define ERR_CONSIST_EMPTY 4        //
#define ERR_LOCO_NOT_FOUND        5        //
#define ERR_CMD_RX_BUF_OFLOW      6        //
#define ERR_INVALID_REQUEST       7        //
#define ERR_SESSION_CANCELLED     8        //
//
// Status codes for OPC_SSTAT
//
#define SSTAT_NO_ACK      1        //
#define SSTAT_OVLD        2        //
#define SSTAT_WR_ACK      3        //
#define SSTAT_BUSY        4        //
#define SSTAT_CV_ERROR    5        //
//
// Error codes for OPC_CMDERR
//
#define CMDERR_INV_CMD    1        //
#define CMDERR_NOT_LRN    2        //
#define CMDERR_NOT_SETUP  3        //
#define CMDERR_TOO_MANY_EVENTS    4        //
#define CMDERR_NO_EV      5        //
#define CMDERR_INV_EV_IDX 6        //
#define CMDERR_INVALID_EVENT      7        //
#define CMDERR_INV_EN_IDX 8        // now reserved
#define CMDERR_INV_PARAM_IDX      9        //
#define CMDERR_INV_NV_IDX 10       //
#define CMDERR_INV_EV_VALUE       11       //
#define CMDERR_INV_NV_VALUE       12       //
//
// Additional error codes proposed and/or agreed but not yet in the current published
specification
//
#define CMDERR_LRN_OTHER  13       // Sent when module in learn mode sees NNLRN for different
module (also exits learn mode)
//
//
// Sub opcodes for OPC_CABDAT
//
#define CDAT_CABSIG       1        //
//
// Aspect codes for CDAT_CABSIG
//
// First aspect byte
//
#define SASP_DANGER       0        //
#define SASP_CAUTION      1        //
#define SASP_PRELIM_CAUTION       2        //
#define SASP_PROCEED      3        //
#define SASP_CALLON       4        // Set bit 2 for call-on - main aspect will usually be at danger
```

```
#define SASP_THEATRE      8       // Set bit 3 to 0 for upper nibble is feather lcoation, set 1
for upper nibble is theatre code
//
// Aspect codes for CDAT_CABSIG
//
// Second Aspect byte
//
#define SASP_LIT 0        // Set bit 0 to indicate lit
#define SASP_LUNAR        1       // Set bit 1 for lunar indication
//
// Remaining bits in second aspect byte yet to be defined - can be used for other signalling
systems
//
//
// Parameter index numbers (readable by OPC_RQNPN, returned in OPC_PARAN)
// Index numbers count from 1, subtract 1 for offset into parameter block
// Note that RQNPN with index 0 returns the parameter count
//
#define PAR_MANU 1        // Manufacturer id
#define PAR_MINVER        2       // Minor version letter
#define PAR_MTYP 3        // Module type code
#define PAR_EVTNUM        4       // Number of events supported
#define PAR_EVNUM5        // Event variables per event
#define PAR_NVNUM6        // Number of Node variables
#define PAR_MAJVER        7       // Major version number
#define PAR_FLAGS8        // Node flags
#define PAR_CPUID9        // Processor type
#define PAR_BUSTYPE       10      // Bus type
#define PAR_LOAD 11       // load address, 4 bytes
#define PAR_CPUMID        15      // CPU manufacturer's id as read from the chip config space, 4
bytes (note - read from cpu at runtime, so not included in checksum)
#define PAR_CPUMAN        19      // CPU manufacturer code
#define PAR_BETA 20       // Beta revision (numeric), or 0 if release
//
// Offsets to other values stored at the top of the parameter block.
// These are not returned by opcode PARAN, but are present in the hex
// file for FCU.
//
#define PAR_COUNT0x18     // Number of parameters implemented
#define PAR_NAME 0x1A     // 4 byte Address of Module type name, up to 8 characters null terminated
#define PAR_CKSUM0x1E     // Checksum word at end of parameters
//
// Flags in PAR_FLAGS
//
#define PF_NOEVENTS       0       // Module doesn't support events
#define PF_CONSUMER       1       // Module is a consumer of events
#define PF_PRODUCER       2       // Module is a producer of events
#define PF_COMBI 3        // Module is both a consumer and producer of events
#define PF_FLiM  4        // Module is in FLiM
#define PF_BOOT  8        // Module supports the FCU bootloader protocol
#define PF_COE   16       // Module can consume its own events
#define PF_LRN   32       // Module is in learn mode
//
// BUS type that module is connected to
//
#define PB_CAN   1        //
#define PB_ETH   2        //
#define PB_MIWI  3        //
#define PB_USB   4        //
//
// Processor manufacturer codes
//
#define CPUM_MICROCHIP    1       //
#define CPUM_ATMEL        2       //
#define CPUM_ARM 3        //
//
// Microchip Processor type codes (identifies to FCU for bootload compatibility)
//
#define P18F2480 1        //
#define P18F4480 2        //
#define P18F2580 3        //
#define P18F4580 4        //
#define P18F2585 5        //
#define P18F4585 6        //
#define P18F2680 7        //
#define P18F4680 8        //
#define P18F2682 9        //
#define P18F4682 10       //
#define P18F2685 11       //
#define P18F4685 12       //
//
#define P18F25K8013       //
#define P18F45K8014       //
#define P18F26K8015       //
#define P18F46K8016       //
#define P18F65K8017       //
#define P18F66K8018       //
#define P18F25K8319       //
```

```
#define P18F26K8320      //
#define P18F27Q8421      //
#define P18F47Q8422      //
#define P18F27Q8323      //
#define P18F14K2225      //
//
#define P32MX534F064     30      //
#define P32MX564F064     31      //
#define P32MX564F128     32      //
#define P32MX575F256     33      //
#define P32MX575F512     34      //
#define P32MX764F128     35      //
#define P32MX775F256     36      //
#define P32MX775F512     37      //
#define P32MX795F512     38      //
//
// ARM Processor type codes (identifies to FCU for bootload compatibility)
//
#define ARM1176JZF_S     1       // As used in Raspberry Pi
#define ARMCortex_A7     2       // As Used in Raspberry Pi 2
#define ARMCortex_A53    3       // As used in Raspberry Pi 3

#ifdef  __cplusplus
}
#endif

#endif // __CBUSDEFS
```